

Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили

А. В. Швед
Є. О. Давиденко

**ПРАКТИЧНИЙ WEB-ДИЗАЙН:
ПРОЕКТУВАННЯ, СТВОРЕННЯ
ТА СУПРОВОДЖЕННЯ WEB-ВУЗЛА**

Навчальний посібник



Миколаїв 2019

УДК 004.738-045.43+004.92]-027.21(075)

Ш 34

Рекомендовано до друку вченою радою Чорноморського національного університету імені Петра Могили (протокол № 2 від 11.10.2018).

Рецензенти:

Литвин В. В. – д-р техн. наук, професор, завідувач кафедри інформаційних систем та мереж Національного університету «Львівська політехніка»;

Яровий А. А. – д-р техн. наук, професор, завідувач кафедри комп'ютерних наук Вінницького національного технічного університету;

Бичков О. С. – канд. фіз.-мат. наук, доцент, завідувач кафедри програмних систем і технологій Київського національного університету імені Тараса Шевченка.

Ш 34

Швед А. В. Практичний web-дизайн : проектування, створення та супроводження web-вузла : навчальний посібник / А. В. Швед, Є. О. Давиденко. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2019. – 192 с.

ISBN 978-966-336-406-3

У навчальному посібнику викладені теоретичні основи розробки web-застосунків: створення та обробка зображень у графічному редакторі Photoshop, мова розмітки гіпертексту HTML, особлива увага приділена стандарту HTML5; каскадні таблиці стилів CSS, у тому числі стандарт CSS3; основи скриптової мови JavaScript.

Розглянуті можливості багатофункціональних бібліотек JavaScript: jQuery та Bootstrap. За допомогою яких реалізується навігація по HTML-документу, маніпуляція даними, обробка подій, анімація та ефекти, разом з тим забезпечується адаптивність web-застосунків.

УДК 004.55:004.4'234

ISBN 978-966-336-406-3

© Швед А. В., Давиденко Є. О., 2019

© ЧНУ ім. Петра Могили, 2019

З М І С Т

ВСТУП	7
РОЗДІЛ 1. СТВОРЕННЯ ТА ОБРОБКА ЗОБРАЖЕНЬ У ГРАФІЧНОМУ РЕДАКТОРІ PHOTOSHOP	8
1.1 Типи комп'ютерної графіки.....	8
1.1.1 Згладжування (Anti-alias).....	10
1.1.2 Інтерполяція (усереднення).....	11
1.2 Кольорові моделі.....	13
1.3 Інтерфейс Photoshop.....	17
1.3.1 Палітри.....	21
1.3.2 Панель інструментів.....	22
1.3.3 Режими відображення	24
1.3.4 Способи зміни масштабу перегляду.....	26
1.3.5 Швидке переміщення зображенням	27
1.4 Інструменти Photoshop	28
1.4.1 Інструменти виділення.....	28
1.4.2 Інструменти заливки.....	29
1.4.3 Шари і текст.....	30
1.4.4 Робота з текстом.....	32
1.4.5 Переміщення.....	35
1.4.6 Трансформація	42
1.4.7 Інструменти ретушування.....	46
1.4.8 Інструменти корекції зображень.....	53
РОЗДІЛ 2. МОВА РОЗМІТКИ ГІПЕРТЕКСТУ HTML.....	55
2.1 Основи HTML	55
2.1.1 Структура документа.....	55
2.1.2 Форматування тексту.....	57

2.1.3	Параграф.....	59
2.1.4	Зображення.....	59
2.1.5	Списки.....	61
2.1.6	Гіперпосилання.....	63
2.1.7	Таблиці.....	64
2.1.8	Рядкові та блокові елементи.....	67
2.1.9	Форми.....	68
2.1.10	Навігаційні карти.....	71
2.2	Основи Html5.....	72
2.2.1	Структурні елементи.....	72
2.2.2	Елементи рівня блока.....	73
2.2.3	Елементи рівня тексту.....	74
2.2.4	Інтерактивні елементи.....	74
2.2.5	Таблиці.....	74
2.2.6	iframe.....	75
2.2.7	Мультимедіа.....	75
2.2.8	Форми.....	77

РОЗДІЛ 3. КАСКАДНІ ТАБЛИЦІ

	СТИЛІВ CSS.....	79
3.1	Синтаксис CSS.....	80
3.2	Підключення CSS.....	80
3.3	Колір і фон.....	82
3.4	Шрифти.....	83
3.5	Форматування тексту.....	84
3.6	Псевдокласи.....	85
3.7	Псевдоелементи.....	86
3.8	Ідентифікатори та класи.....	87
3.9	Розмір елементів.....	89
3.10	Боксова модель.....	90
3.11	Плаваючі елементи.....	95
3.12	Позиціонування елементів.....	97
3.13	Керування шаром.....	99
3.14	@-правила CSS.....	101

РОЗДІЛ 4. ОСНОВИ JAVASCRIPT	104
4.1 Основні поняття мови JavaScript.....	104
4.1.1 Підключення JavaScript.....	104
4.1.2 Змінні. Правила побудови ідентифікаторів.....	104
4.1.3 Коментарі.....	105
4.1.4 Константи.....	105
4.1.5 Типи даних.....	105
4.2 Оператори	106
4.2.1 Оператор присвоєння.....	106
4.2.2 Арифметичні оператори	107
4.2.3 Оператори порівняння.....	109
4.2.4 Логічні оператори	110
4.3 Керуючі структури JavaScript.....	110
4.3.1 Оператор умови.....	110
4.3.2 Оператор ?.....	112
4.3.3 Оператор множинного вибору.....	112
4.3.4 Оператори циклу	113
4.4 Функції.....	115
4.4.1 Оголошення функції.....	115
4.4.2 Повернення значення	116
4.4.3 Локальні і глобальні змінні.....	117
4.4.4 Стандартні функції. Вікна повідомлень.....	118
4.5 Об'єкти	119
4.5.1 Об'єкт Date.....	119
4.5.2 Об'єкт Math.....	120
4.5.3 Об'єкт Array.....	121
4.6 Модель подій	123
4.7 Основи об'єктної моделі документа.....	124
РОЗДІЛ 5. JAVASCRIPT-БІБЛІОТЕКА JQUERY.....	128
5.1 Введення в jQuery.....	128
5.2 Робота з jQuery.....	128
5.3 Фільтри в jQuery	134
5.3.1 Базові фільтри.....	134

5.3.2	Фільтри контенту	137
5.3.3	Додаткові фільтри атрибутів	138
5.3.4	Фільтри елементів форм	138
5.3.5	Маніпуляція атрибутами в jQuery	139
5.3.6	Функції для маніпуляції класами	142
5.4.	Робота з HTML та CSS в jQuery	144
5.4.1	Методи jQuery для роботи з html	144
5.4.2	Методи jQuery для роботи з CSS	144
5.4.3	Маніпуляція елементами, що вставляються	145
5.5	Робота з DOM в jQuery	147
5.5.1	Методи фільтрації набору елементів	148
5.5.2	Функції пошуку по DOM в jQuery	150
5.5.3	Обробники подій в jQuery	156
5.6	Анімація та ефекти	159
5.6.1	Функція animate()	159
5.6.2	Ефекти в jQuery	161
5.6.3	Функції для роботи з чергою	166
5.6.4	Допоміжні функції jQuery	168
5.6.5	Метадані в jQuery	172
РОЗДІЛ 6. БІБЛІОТЕКА BOOTSTRAP		174
6.1	Знайомство з Bootstrap	174
6.2	Колонкова верстка	175
6.3	Bootstrap CSS	177
СПИСОК ЛІТЕРАТУРИ		190

В С Т У П

Згідно зі щорічними дослідженнями, найпопулярніша професія серед користувачів основних сервісів – це web-розробник. Саме до цієї категорії належать усі front-end-розробники. Для створення інтерактивних і компактних web-застосунків необхідно знати чимало сучасних web-стандартів. У цьому посібнику зібрано багато цікавого матеріалу, який послужить хорошим поштовхом для web-програмістів, що тільки починають свій тернистий професійний шлях.

Тематично навчальний посібник містить 6 розділів.

У першому розділі висвітлені принципи роботи з кольором, основи створення та обробки зображень у графічному редакторі Photoshop.

У другому розділі розглянуті основи мови розмітки гіпертексту HTML і стандарту HTML5. Розглянуто основні елементи мови, що необхідні для розробки статичних web-документів.

Третій розділ присвячено теоретичним основам CSS, у тому числі стандарту CSS3. Розглянуто основи технології верстки сайтів, основну увагу приділено блочній верстці.

У четвертому розділі викладені основи мови JavaScript, розглянуті основні положення моделі подій і основи об'єктної моделі документа.

У п'ятому розділі розглянуті особливості роботи з бібліотекою jQuery, базові селекторні вибірки, робота з DOM, анімація та ефекти, а також різноманітні методи для маніпуляції елементами.

Шостий розділ присвячено найпопулярнішій front-end бібліотеці Bootstrap, за допомогою якої можливо створити застосунок з використанням вбудованих змінних Sass, адаптивної системи сіток, потужних плагінів, побудованих на jQuery.

РОЗДІЛ 1

СТВОРЕННЯ ТА ОБРОБКА ЗОБРАЖЕНЬ У ГРАФІЧНОМУ РЕДАКТОРІ PHOTOSHOP

1.1 Типи комп'ютерної графіки

Розрізняють два основні типи комп'ютерної графіки – растрову і векторну (рис. 1.1). Знання про їх природу, відмінності, взаємодії є основою професійної роботи дизайнера.

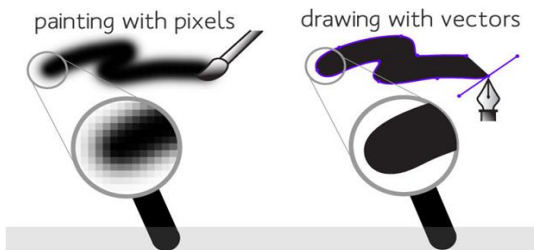


Рис. 1.1. Приклад растрового і векторного зображення

У растровій графіці зображення складається із найдрібніших точок – пікселів (*pixel, px*). Будь-яке растрове зображення має фіксовану кількість пікселів. Якщо збільшити графічне зображення, то можна побачити що пікселі – це різнокольорові квадрати, зазубрені краї ліній. При масштабуванні (збільшенні) растрового зображення дуже важко зрозуміти, що на ньому зображено.

Якість друку растрових зображень залежить від роздільної здатності. При масштабуванні, в силу своєї піксельної природи, растрові зображення завжди втрачають якість. Прикладом растрового зображення може служити будь-яка фотографія, відсканована або отримана шляхом цифрової зйомки.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

У векторній графіці якість зображення не залежить від роздільної здатності. Векторні об'єкти описуються математичними рівняннями, тому при масштабуванні вони не втрачають своєї якості. Але рівняння самі по собі нічого не значать, якщо не можна побачити результат їх застосування. Векторні об'єкти раструються на пристроях виводу, таких, як монітор або принтер. Як результат, векторна графіка застосовується для великих, чітких форм, наприклад, тексту, логотипів, плоских рисунків.

Основними характеристиками растрового зображення є висота і ширина, що задаються в момент його створення, і які можна змінити в процесі роботи.

Залежно від подальшого використання обирають різні одиниці виміру: якщо ви хочете використовувати зображення в поліграфії (друкований відбиток на папері, фотографія в рамочці і т. ін.) – обирають сантиметри (см); якщо для web-графіки – пікселі (px). Величина, що визначає якість друку растрових зображень – це роздільна здатність.

Роздільна здатність зображення – це кількість пікселів (точок) на одиницю довжини. Зазвичай її вимірюють у точках на дюйм (*Dots per Inch, dpi*) або в пікселях на дюйм (*Pixel per Inch, ppi*). Дюйм дорівнює 2,54 см.

Піксель (скорочення від *Picture Element*, елемент зображення) – найменший неподільний компонент растрового зображення, над яким виконується робота. Він має дві характеристики: положення і колір. Чим більша роздільна здатність, тим менший розмір пікселя і як результат – більша їх кількість на дюйм, і тим краща якість зображення (рис. 1.2).

Роздільна здатність підбирається для кожного зображення індивідуально і залежить від того, де воно буде застосовано. Наприклад, якщо ви плануєте використовувати фото в Інтернеті, то роздільна здатність обирається 72 ppi. Такий вибір обмежується монітором, з якого буде трансляватися це зображення. Основним критерієм для Інтернету є швидкість за-

вантаження зображень, а не їх дивовижна якість, тому обираються відповідні формати збереження файлів, де якість не є найважливішим критерієм.

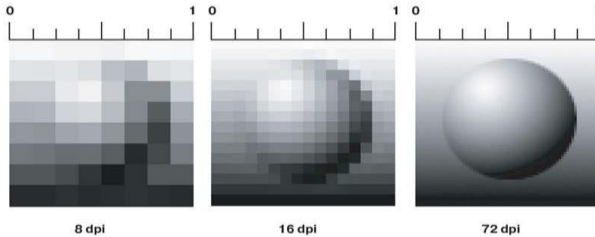


Рис. 1.2. Роздільна здатність рисунку

1.1.1 Згладжування (Anti-alias)

Криві і діагональні лінії зображення важко передати на моніторі через те, що він може відображати тільки прямокутні елементи. Для збереження плавного виду країв існує механізм згладжування (*Anti-alias*). Для різних інструментів і команд у редакторі існує опція *Anti-alias*, яка за замовчуванням є ввімкненою.

Згладжування (*Anti-alias*) – це механізм розташування пікселів різного ступеня прозорості уздовж країв («проблемних областей») кривих і діагональних ліній.

На рис. 1.3 показані дві діагональні лінії. У лівій згладжування ввімкнено – на краях видно пікселі різної прозорості, які «заповнюють» простір між різкими краями. Праворуч наведена ступінчаста лінія з різкими, зазубреними краями (опція *Anti-alias* вимкнена).

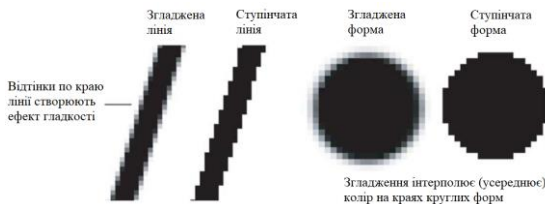


Рис. 1.3. Ефект згладжування

На рис. 1.3 представлено збільшене зображення круглої форми зі згладжуванням і без нього.

Коли ви обираєте інструмент **Pencil** (Олівець), параметр **Brush** (Пензель) у нього – жорсткий пензель, без згладжування.

Якщо ви створюєте виділення і опція **Anti-alias** ввімкнена, це призведе до згладжених форм майбутнього об'єкта.

Якщо сильно збільшити діагональну лінію, контури якої згладжені, можна побачити, що в міру віддалення від лінії в пікселях по її краю поступово зменшується вміст кольору лінії і посилюється інтенсивність кольору фону зображення.

Таким чином, згладжування полягає в утворенні плавного переходу між внутрішньою областю та її фоном у разі непрямолінійних об'єктів.

1.1.2 Інтерполяція (усереднення)

Припустимо, що ви маєте фото, яке має задані розміри в пікселях і ви бажаєте збільшити його в три рази за висотою та шириною. Виникає справедливе питання: звідки Photoshop візьме додаткові пікселі? І, найголовніше, в який колір він їх зафарбує?

Існують методи інтерполяції (*Interpolation Method*), за допомогою яких Photoshop може «додумати», як нові пікселі повинні бути вставлені в зображення:

- за допомогою створення пікселів, що мають найближчий до вихідного пікселя колір;
- за допомогою інтерполяції пікселів, що оточують по горизонталі і вертикалі, і подальшого створення кольорового усереднення загальних сум для нових пікселів;
- за допомогою інтерполяції пікселів по горизонталі, вертикалі і діагоналі, і використання усереднення підсумкових кольорів для кожного нового пікселя.

У редакторі Photoshop ці три методи мають назву **Nearest Neighbor** (інтерполяція за найближчими сусідами), **Bilinear**

1.2 Кольорові моделі

Зображення, яке ви підготували за допомогою Photoshop, можна роздрукувати на принтері або переглянути на іншому комп'ютері (або на екрані телевізора).

Але на папері «результат» може виглядати зовсім не так, як на екрані монітора. Кольори виявляються спотвореними: блакитне небо набуває лілового відтінку, а людина – неприродної малинової засмаги. Основною причиною спотворення екранних кольорів у процесі друку є діаметрально протилежні способи генерації кольору монітором і принтером.

RGB

Передусім необхідно усвідомити, що сприйняття кольору є результатом роботи мозку. Відчуття кольору створюється електромагнітними коливаннями з довжинами хвиль від 380 до 750 нм, що потрапляють в око людини. Експериментально (ще в 1852 р.) було встановлено, що будь-який колір можна отримати складанням трьох світлових потоків: червоного (**R**) – довжина хвилі приблизно 630 нм, зеленого (**G**) – приблизно 528 нм і синього (**B**) – приблизно 457 нм.

Саме на цьому принципі базується створення кольорового зображення на екрані монітора і телевізора. Поверхня монітора складається з найдрібніших точок (пікселів) червоного, зеленого і синього кольорів (тріада люмінофорів), форма цих пікселів залежить від типу електронно-променевої трубки екрана. При потраплянні електронного променя на піксель, останній забарвлюється в певний відтінок свого кольору залежно від сили сигналу. Оскільки пікселі маленькі, то навіть з невеликої відстані вони стають невиразними і створюють три світлові потоки, які, потрапляючи в око, сприймаються нами як колір. Цей колір може бути описаний за допомогою трьох складових – **R**, **G** і **B**. Ця кольорова модель отримала назву **RGB** і була прийнята в 1931 р.

Відповідно до цієї моделі суміш червоного і зеленого дає жовтий колір (*Yellow*), червоного і синього – пурпурний (*Magenta*), синього і зеленого – блакитний (*Cyan*), а червоного, зеленого і синього – білий. У системі **RGB** кожен колір на екрані монітора має 256 градацій яскравості (від 0 до 255). Таким чином, на екрані монітора може бути відображено більше 16 млн кольорів.

Якщо змішати червону, зелену і синю фарби, то біла, напевно, не вийде. Абсолютно вірно, тому що фарби не випромінюють світло на зразок сонця, лампочок або електронно-променевих трубок. Коли ми бачимо кольорове зображення в журналі, то в око надходить світловий потік, відбитий від паперу, покритого фарбою. Якщо ми бачимо червоний аркуш паперу при денному світлі, то це означає, що фарба поглинає всі світлові потоки і відображає тільки червоний. Освітить той самий аркуш паперу синім кольором, і він стане чорним, тому що фарба не відображає синій колір.

СМУК

Трьома основними кольорами в живописі є синій, червоний і жовтий. Змішуючи їх, художники отримують різні кольори на своїх полотнах.

Спадкоємцями цієї тріади кольорів при друці стали блакитний (*Cyan*), пурпурний (*Magenta*) і жовтий (*Yellow*) кольори. Однак якщо теоретично при змішуванні цих кольорів виходить чорний колір, то практично цей колір має коричневий відтінок. Це пов'язано з тим, що ідеальних фарб не існує. Тому під час друку додають як мінімум ще одну фарбу – чорну. Подібна кольорова модель називається **СМУК**.

СМУК – субтрактивна модель. Іншими словами, кольори створюються шляхом поглинання або «віднімання» з видимої частини спектру світлових хвиль певної довжини.

У результаті хвилі, що не поглинулися відбиваються, і відбите світло – це ті кольори, які ми бачимо.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

На відміну від **RGB**, кількість кожного кольору задається у відсотках від 0 до 100. Перетворення зображення з моделі **RGB** на модель **СМЬК** виконується командою **Image** → **Mode** → **СМЬК** (Зображення → Режим → СМЬК) (рис. 1.6).

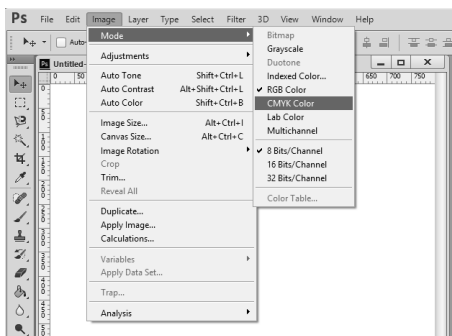


Рис. 1.6. Перетворення зображення на модель *СМЬК*

Як правило, після виконання цієї команди кольори зображення на екрані змінюються. Але майте на увазі, що діапазон відтінків **СМЬК** значно менший, ніж **RGB**, і коли ви переглядаєте на екрані зображення в режимі **СМЬК**, це лише імітація на екрані друкованих кольорів. Не всі кольори **СМЬК** можна відтворити на моніторі, а також не всі кольори **RGB** можна відтворити в **СМЬК**.

HSB

Ця модель вважається найбільш зрозумілою тому, що в ній присутня координата **Hue** (Тон). Ви можете легко зрозуміти, про який колір йдеться, якщо у вас перед очима, а краще в голові, знаходиться круг з координатами кольорів (рис. 1.7).

Hue задається в градусах і приймає значення від 0 до 360.

Друга координата – **Saturation** (Насиченість) – це радіус кола. Найбільш насичені кольори лежать на межі кола і мають координати 100. Білий колір має координату 0. На радіусі кола лежать відтінки кольорів.

Третя координата – **Brightness** (Яскравість) – приймає значення від 0 до 100. Якщо яскравість дорівнює 0, то колір чорний.

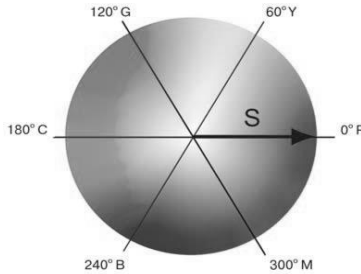


Рис. 1.7. Модель HSB

На панелі інструментів у Photoshop задаються **Foreground Color** (Основний колір) і **Background Color** (Фоновий колір).

Foreground Color (колір переднього плану, або основний) у Photoshop використовується для малювання, заливки документа або виділеної області й у якості початкового кольору інструмента **Gradient** (Градiєнт).

Background Color (колір заднього плану, або фоновий) з'являється при видаленні пікселів, при відсутності прозорості, а також завершує градієнт.

Клікання по піктограмі **Foreground** або **Background** викликає діалогове вікно **Color Picker** (Палітра кольорів). У діалоговому вікні можна задавати колір, клацаючи мишею у великому квадраті, а також при введенні в поля відповідних значень. Праворуч від квадрата добору кольору знаходиться шкала параметрів. На рис. 1.8 увімкнено функцію **H** (Тон).

Установивши повзунок шкали активного параметра на позицію 230 градусів, ви обрали синій колір.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

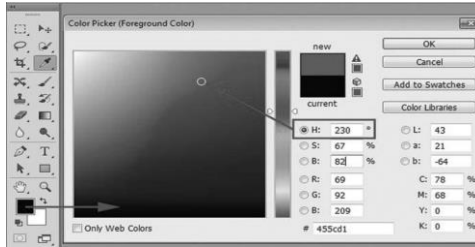


Рис. 1.8. Вікно налаштування кольору в Photoshop

У квадраті добору кольору виберіть найяскравіший синій колір і перемкніть активний параметр на *S* (Насиченість) (рис. 1.9).

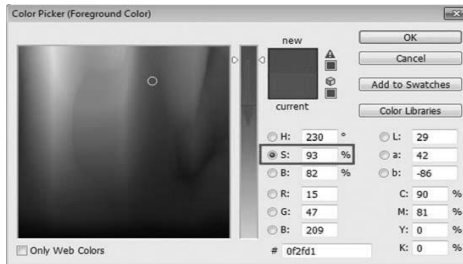


Рис. 1.9. Вибір відтінку кольору в Photoshop

Натиснувши на кнопку ОК, ви перенесете цей колір у програму. Вибір кольору за моделлю **HSB** вважається інтуїтивно зрозумілим. Якщо у вас задані координати кольору в колірній моделі, просто введіть їх у відповідні поля.

1.3 Інтерфейс Photoshop

Після завантаження редактора Photoshop на екрані з'являється вікно програми, в якому можна відкрити вже готове зображення або створити нове.

Для створення нового документа необхідно виконати команду **File** → **New** (Файл → Створити). У діалоговому вікні можна задати параметри нового документа.

У поле **Name** (Ім'я) за замовчуванням встановлено значення *Untitled-1* (Без назви-1).

На початку роботи не варто витратити час на вигадування назви, назву можна змінити під час збереження файлу.

Поле **Preset** (Набір) являє собою випадаючий список розділів, що містить різні варіанти використання документів (рис. 1.10). Обравши потрібний розділ із цього списку, ви можете в подальшому з розташованого нижче списку *Size* (Розмір) обрати документ з попередньо встановленими для даних цілей розмірами і якістю друкованого відбитка.

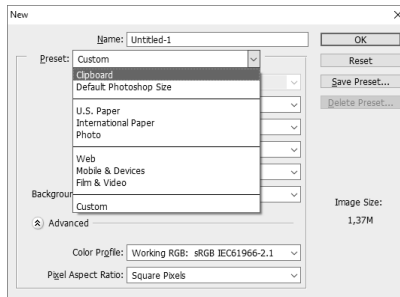


Рис. 1.10. Варіанти використання документів у Photoshop

Наприклад, набір **International Paper** (Міжнар. формат паперу) задається в міліметрах, **Web** – у пікселях і т. ін. (рис. 1.11).

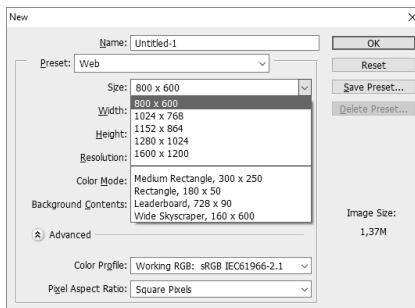


Рис. 1.11. Варіант використання набору *Web*

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Параметри документа – це його висота, ширина, роздільна здатність і кольорова модель.

Для прикладу скористаємося вже готовими розмірами, обравши зі списку **Preset** (Набір) розділ **Web**, а **Size** (Розмір) візьмемо 800x600 (роздільна здатність екрана деяких моніторів). Зауважте, при цьому автоматично встановлюється роздільна здатність 72 ppi.

У поле **Background Contents** (Вміст фону) встановимо значення **White** (Білий), щоб поле документа було білим.

У версії CS6 розробники надали користувачам можливість вибору кольору інтерфейсу. Більш звичний для Photoshop інтерфейс був сірого кольору. Для встановлення сірого кольору необхідно виконати команду **Edit** → **Preferences** (Редагувати → Параметри), обравши розділ **Interface** (Інтерфейс) – колір «сірий квадратик».

Розглянемо робочий простір програми з відкритими файлами (рис. 1.12). Для цього відкриємо файл з будь-яким рисунком командою **File** → **Open** (Файл → Відкрити).

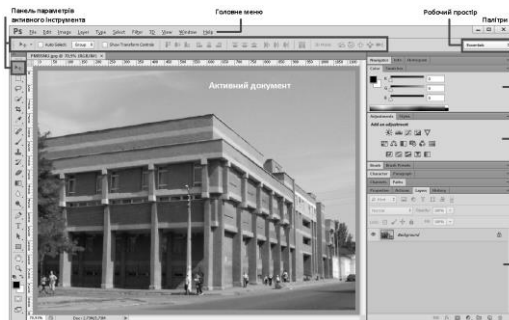


Рис. 1.12. Вікно програми Photoshop

Рядок заголовка показує назву і піктограму програми Adobe Photoshop. Кнопки управління у правій частині рядка використовуються для згортання/розгортання, зміни розмірів і закриття вікна програми.

Головне меню містить основні команди управління, об'єднані за загальним призначенням і містить такі розділи.

File (Файл). Операції роботи з файлами, такі, як відкриття і закриття, імпорт і експорт. У цьому пункті знаходяться команди для пакетної автоматичної обробки файлів, отримання зображень із пристроїв введення, виведення документа на друк.

Edit (Редагувати). Команди редагування – скасування та повернення дій, операції з буфером обміну, команди заливки і обведення, трансформації, задання пензлів, візерунків і т. ін., а також установки програми.

Image (Зображення). Команди, призначені для зміни зображення – кольорової моделі, розміру, а також команди кольорової і тонової корекції.

Layer (Шар). Команди роботи з шарами.

Type (Тип). Команди роботи з текстом.

Select (Виділити). Команди створення, модифікації, збереження, виділення.

Filter (Фільтр). Різні фільтри редактора.

View (Перегляд). У цьому пункті зосереджено все, що ви можете бачити на екрані. Команди зміни масштабу перегляду документа, а також відображення різних допоміжних елементів інтерфейсу.

Window (Вікно). Команди організації робочого простору, відображення палітри і вікон документів.

Help (Довідка). Виклик довідкової інформації і допомоги.

Панель параметрів активного інструмента (**Options**). Вміст цієї панелі залежить від обраного інструмента в палітрі інструментів.

Рядок стану активного документа (у попередніх версіях він знаходився в нижній частині вікна програми). Призначений для відображення інформації про документ, робочі диски, активний інструмент тощо.

1.3.1 Палітри

Палітри містять набори або налаштування, необхідні в роботі (рис. 1.12). Досить рідко необхідно бачити всі палітри відразу. Тому більшість палітр виводяться в сітці, а також можуть бути представлені в повністю розгорнутому вигляді. Повне представлення незручне тому, що займає більшу частину робочої області програми.

Варіанти відображення стандартного набору палітр **Essential** (Основне) представлені на рис. 1.13.

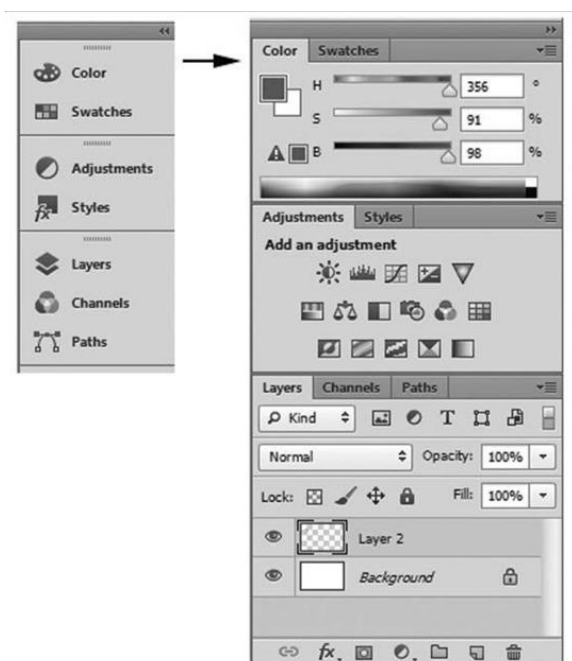


Рис. 1.13. Варіанти відображення стандартного набору палітр

Клікаючи по піктограмі «подвійної стрілки» вгорі палітр, ви можете розкривати палітри, повторне клікання – знову згортає їх, залишаючи тільки значки.

1.3.2 Панель інструментів

На панелі інструментів знаходяться всі інструменти, кнопки вибору кольору переднього і заднього плану, а також засоби перегляду зображення (рис. 1.14).

Стрілка, що розташована в правому нижньому кутку піктограми із зображенням інструмента, свідчить про наявність панелі, що розкривається, яка містить додаткові інструменти.

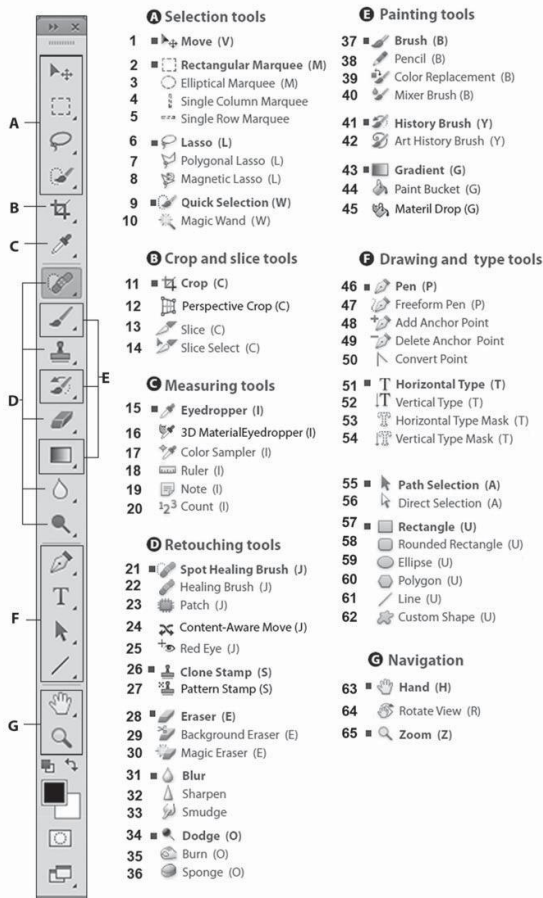


Рис. 1.14. Розгорнутий вигляд панелі інструментів

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Не натискаючи кнопки миші, встановіть покажчик над одним з інструментів, і ви побачите підказку програми – напис з назвою інструмента і клавішу, натиснувши яку, ви викличете цей інструмент (рис. 1.15).

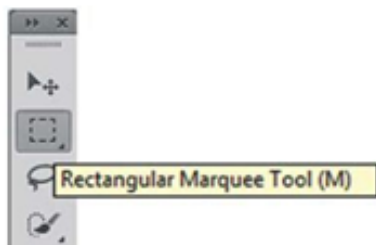


Рис. 1.15. Підказка

Інструмент активізується кліканням по його піктограмі або натисканням швидкої клавіші (її відображає підказка) (рис. 1.16).

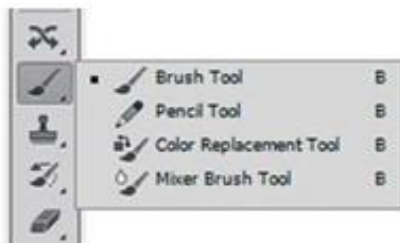


Рис. 1.16. Вибір інструмента

Обраний інструмент підсвічений сірим кольором, і під головним меню програми розташовується панель його параметрів. Інструмент активний до тих пір, поки ви не оберете інший.

У правому нижньому куті піктограм деяких інструментів стоїть маленький трикутник. Це означає, що за цим інструментом «заховані» інші, додаткові.

Вибрати «захований» інструмент можна декількома способами:

- установіть покажчик на піктограму, де є трикутник, натисніть кнопку миші і, дочекавшись появи додаткових інструментів, виділіть один з них та відпустіть кнопку миші;
- натиснувши клавішу <Alt>, клацніть на піктограмі інструмента. З кожним клацанням у комірці з'являється черговий додатковий інструмент;
- для перемикання між інструментами в одній групі з відповідною «гарячою» клавішею необхідно натиснути клавішу <Shift>.

1.3.3 Режими відображення

У програмі використовуються три режими відображення вікна програми.

У нижній частині панелі інструментів знаходиться піктограма перемикання режимів відображення (рис. 1.17).

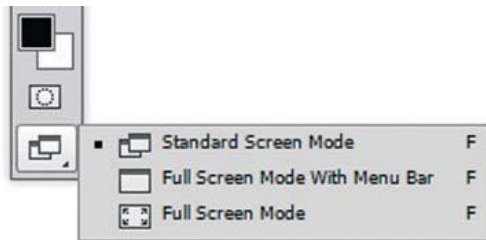


Рис. 1.17. Піктограми перемикання режимів відображення

Перемикатися між режимами також можливо «гарячою» клавішею <F>.

За замовчуванням включений перший режим – **Standard Screen Mode** (Стандартний режим екрана) (рис. 1.18).

Практичний web-дизайн: проектування, створення та супроводження web-вузла

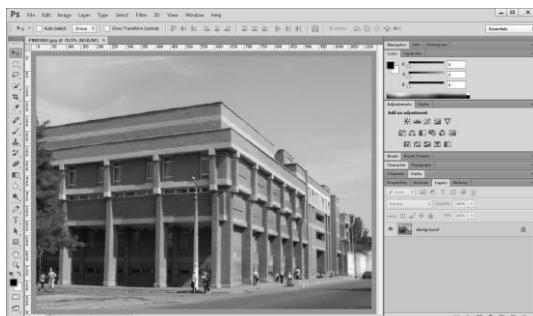


Рис. 1.18. Стандартний режим відображення

Другий режим називається **Full Screen With Menu Bar** (Повноекранний режим з рядком меню) (рис. 1.19). Однак у цьому режимі немає ні рядка стану документа, ні смуг прокрутки.

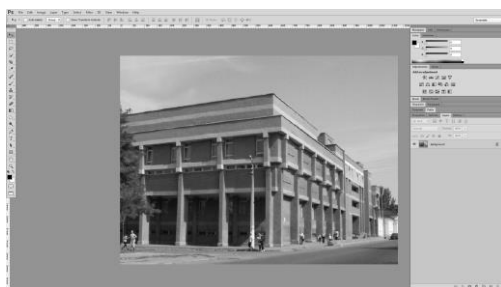


Рис. 1.19. Відображення на весь екран з головним меню

Третій режим – **Full Screen Mode** (Повноекранний режим) (рис. 1.20).



Рис. 1.20. Відображення на весь екран

Активний документ відображається на чорному фоні, що рекомендується при остаточному композиційному перегляді.

1.3.4 Способи зміни масштабу перегляду

Переміщення зображенням і швидка зміна масштабу дуже часто виявляються важливими прийомами роботи. Найбільш часто при роботі використовується масштаб 100 і 200 %, це означає, що ви бачите тільки невелику частину цілого зображення на моніторі. Через це вам необхідно вміти швидко змінювати масштаб зображення, щоб побачити, як ваша робоча частина поєднується з цілим зображенням (рис. 1.21). Крім того, ви повинні зосередитися на виконанні дизайнерських завдань, а не витратити час на вибір інструментів.



Рис. 1.21. Різні масштаби перегляду зображення:
25 %; 50 %; 100 %; 400 %

Працюючи із зображенням, ви можете використовувати такі прийоми:

- збільшення масштабу (коли активним є будь-який інструмент) – «гарячі» клавіші <Ctrl>+<+>.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

- зменшення масштабу (коли активним є будь-який інструмент) – «гарячі» клавіші **<Ctrl>+<->**.

- збільшення певної частини зображення – однією рукою натисніть комбінацію клавіш **<Ctrl> + <Space>**, на екрані з'явиться лупа; іншою рукою (за допомогою миші) обведіть лупою в рамочку область, яку необхідно збільшити.

Перехід до масштабу 100 %:

- двічі клікніть по значку інструмента **Zoom** (Масштаб) (№ 65 на рис. 1.14);

- у головному меню виберіть команду **View** → 100 % (Перегляд → 100 %) або натисніть комбінацію клавіш **<Ctrl> + <1>**.

Побачити зображення повністю:

- двічі клікніть по значку інструмента **Hand** (Рука) (№ 63 на рис. 1.14);

- натисніть комбінацію клавіш **<Ctrl> + <0>**;

- у головному меню оберіть команду **View** → **Fit on Screen** (Перегляд → За розміром екрана).

1.3.5 Швидке переміщення зображенням

Якщо ви працюєте у великому масштабі і розмір зображення більший, ніж може поміститися на екрані монітора, у документа з'являються смуги прокрутки.

Для прокрутки використовується інструмент **Hand** (Рука) (№ 63 на рис. 1.14).

Його легко викликати, утримуючи клавішу **<Space>** при будь-якому активному інструменті (за винятком інструмента **Type** (Текст) при введенні тексту).

1.4 Інструменти Photoshop

1.4.1 Інструменти виділення

Клацніть по цьому інструменту (рис. 1.22) і утримуйте ліву кнопку миші, розкриється меню вибору: прямокутна область, овальна область, горизонтальний рядок, вертикальний рядок.

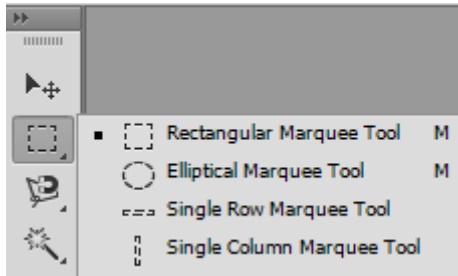


Рис. 1.22. Інструмент *Виділення*

У кожного інструмента є панель параметрів, вона знаходиться під рядком меню і змінюється залежно від обраного інструмента (рис. 1.23).

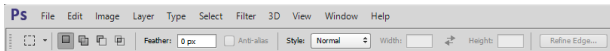


Рис. 1.23. Панель параметрів інструмента *Виділення*

Отже, для прямокутного виділення маємо такі параметри:

- нове виділення. Створюється нове виділення, при цьому наявні виділені області (якщо вони є) зникають;
- додати до виділення. Позначена область додається до вже існуючого виділення;
- видалити з виділення. Виключає виділену область з виділення;
- виділити перетин. З двох виділень (старого і нового) виділить область перетину.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Feather (Розмиття) – по краях виділення з’являється область частково виділених пікселів. Дозволяє зробити плавний перехід об’єкта в фон.

Style (Стиль) – задає спосіб виділення: *normal* (нормальний) – вільне виділення, *fixed aspect ratio* (стале співвідношення) – пропорційне виділення, *fixed size* (сталий розмір) – область виділення заданого розміру.

Для всіх інструментів виділення параметри аналогічні.

1.4.2 Інструменти заливки

Клацніть по цьому інструменту (рис. 1.24) і утримуйте ліву кнопку миші, розкриється меню вибору: градієнт, заливка та 3D-накладання матеріалу.

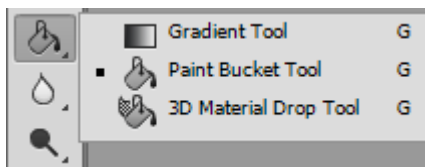


Рис. 1.24. Інструменти Заливка

Розглянемо панель параметрів (рис. 1.25).

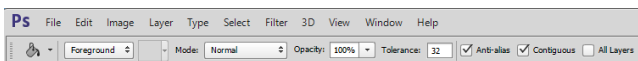


Рис. 1.25. Панель параметрів інструмента Заливка

Для заливки існують такі параметри:

- **Fill** (Заливка) – визначає, що використовувати в якості «фарби»: *Foreground* (Передній план) залле основним кольором, шаблон (*Pattern* – Візерунок) залле деяким візерунком;
- **Pattern** (Шаблон) – візерунок для заливки;
- **Mode** (Режим) – спосіб накладення пікселів;
- **Opacity** (Непрозорість) – визначає ступінь прозорості заливки;

– **Tolerance** (Допуск) – діапазон кольорів для заповнення.

Галочка згладжування означає пом'якшити межі переходу.

Гradient створює заливку з плавним переходом між двома або кількома кольорами.

Розглянемо панель параметрів (рис. 1.26).

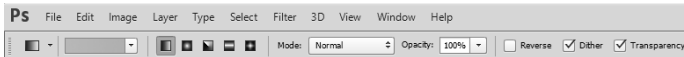


Рис. 1.25. Панель параметрів інструмента *Gradient*

Якщо клацнути по стрілочці справа другої піктограми, то відкриється палітра з варіантами gradienta, при подвійному натисканні перед стрілкою відкриється вікно редагування gradienta (тут можна налаштувати свої варіанти gradienta).

Третя група піктограм – визначає вид gradienta (зліва направо – лінійний, радіальний, кутовий, відбитий, ромбоподібний).

Reverse (Обернути) – галочка вказує на геометричний переворот gradienta.

Dither (Тремтіння) – галочка вказує на згладження для зменшення сегментації.

Transparency (Прозорість) – галочка вказує на використання прозорості.

1.4.3 Шари і текст

Основним поняттям програми Photoshop є шар. Уявіть собі безліч тонких скляних пластинок, на яких намальовані різні фігури. Якщо накласти ці пластини одна на одну, то отримаємо нове зображення.

Аналогічно влаштована багат шарова картинка. На різних шарах ми поміщаємо різні об'єкти (фон, окремі фігури, текст і т. ін.), коли цілісне зображення готове – об'єднуємо всі шари в один.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Створіть новий файл, розміром 400x200 пікселів. Розширте вікно, щоб навколо білого полотна було видно сірий фон.

Для зручності роботи в Photoshop існує кілька основних палітр, які зазвичай відкриті в головному вікні програми і розташовуються праворуч. Знайдіть серед них палітру **Layers** (Шари) (рис. 1.26). Якщо палітра не відображається в головному вікні, то оберіть у меню пункт **Window** → **Layers** (Вікно → Шари).

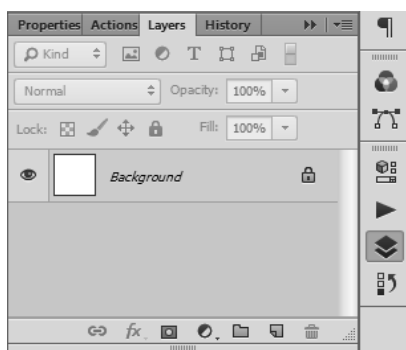


Рис. 1.26. Палітра *Шари*

На цій палітрі передбачено дуже багато налаштувань. Зараз ознайомимося з такими:

- створити новий шар;
- видалити шар;
- шар увімкнений (якщо клікнути на око, то шар вимкнеться – око зникне).

Синім кольором виділено активний шар, тобто той, з яким ми зараз працюємо. Щоб зробити шар активним, досить по ньому клікнути мишкою.

Зараз на нашій палітрі шарів відображено тільки один шар – *Background* (Фон). Це фон зображення, яке ми створили.

Клікніть по іконці «створити новий шар», з'явиться новий шар з прозорим фоном (сірі та білі квадратики говорять про те, що шар прозорий) (рис. 1.27).

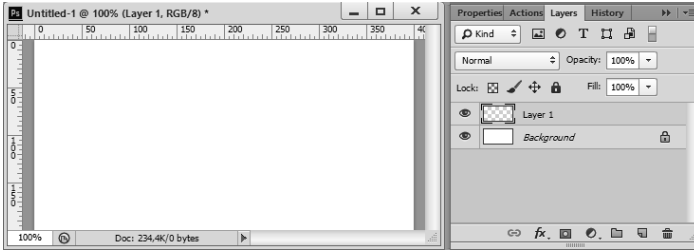


Рис. 1.27. Створення нового шару

Переконайтеся, що цей шар активний (тобто виділений синім).

Візьміть інструмент **Відро** і залийте цей шар, наприклад, жовтою фарбою.

Подивіться на палітру шарів, наш *Background*, як і раніше, білий, а *Layer 1* – жовтий.

Вимкніть *Layer 1* (клацнувши по оку) і на нашому зображенні ви побачите білий фон. Увімкніть шар (клацніть ще раз) і побачите жовту заливку.

Переконайтеся, що активним зараз є *Layer 1* (виділено синім) і створіть ще один шар (натисніть на «створити новий шар»). З'явиться прозорий *Layer 2*. Чому треба було переконаватися, що активним є *Layer 1*? Тому що новий шар завжди створюється над активним.

Порядок шарів можна встановити простим перетягуванням – клацаєте лівою клавішею миші по шару, який хочете перетягнути і, утримуючи її натиснутою, тягнете шар угору або вниз до необхідного рівня, відпускаєте кнопку миші і бачите, що шар тепер знаходиться на новому місці.

1.4.4 Робота з текстом

Для роботи з текстом знову звернемося до панелі інструментів (рис. 1.28). Натисніть і утримуйте ліву кнопку миші на цьому інструменті, розкриється меню вибору.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

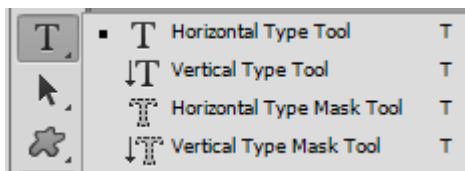


Рис. 1.28. Інструмент *Текст*

Відразу з'являється інтуїтивно зрозуміла панель параметрів інструмента **Текст** (рис. 1.29).

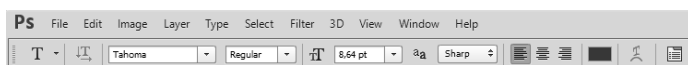


Рис. 1.29. Панель параметрів інструмента *Текст*

При натисканні на іконку «панель символ» відкриється панель, на якій можна задати ширину, висоту і відстань між символами у словах (рис. 1.30).



Рис. 1.30. Панель *Символ*

Візьміть інструмент **Текст** (клікніть по ньому), тепер клікніть по квадратику кольору на панелі параметрів і оберіть червоний колір, установіть великий розмір шрифту (60pt), по-

тім клікніть по зображенню (з'явиться вертикальний курсор) і напишіть яке-небудь слово (рис. 1.31).



Рис. 1.31. Введення тексту

Зверніть увагу на палітру шарів, шар **Текст** позначений буквою **T**, а ім'ям шару є введений текст.

Тепер клікніть по **Warp text** (Деформувати текст) на панелі параметрів, відкриється вікно (рис. 1.31).

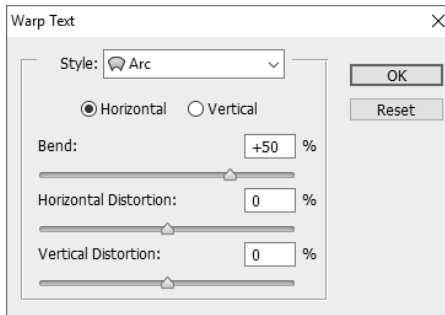


Рис. 1.31. Деформація тексту

У цьому меню зібрані різні варіанти деформації тексту. Виберіть в списку **Style** (Стиль) будь-який варіант деформації (наприклад, *Wave* (Хвиля)). Задайте параметри:

<i>Bend</i> (Згін)	+ 50 %
<i>Horizontal Distortion</i> (Горизонтальне викривлення)	-35 %
<i>Vertical Distortion</i> (Вертикальне викривлення)	-45 %

Натисніть ОК і подивіться на результат (рис. 1.32).

Практичний web-дизайн: проектування, створення та супроводження web-вузла



Рис. 1.32. Приклад деформації тексту

1.4.5 Переміщення

Для переміщення різних об'єктів на панелі інструментів існує спеціальний інструмент (рис. 1.33).



Рис. 1.33. Інструмент *Переміщення*

Клікніть по цьому інструменту, підведіть курсор миші до тексту (переконайтеся, що шар з текстом активний), натисніть ліву кнопку миші і, не відпускаючи її, перетягніть текст у будь-яке інше місце. Не забувайте, що всі інструменти працюють саме з активним шаром і його об'єктами.

Відкриття зображення

Спочатку необхідно відкрити саме зображення у програмі Photoshop. Для цього оберіть команду меню **File** → **Open** (Файл → Відкрити). У вікні, що відкрилося, оберіть потрібне зображення.

Зверніть увагу, в пункті **Тип файлу** є величезний список підтримуваних файлів. Якщо ви виберете певний тип файлу (наприклад, *gif*), то у вікні вибору будуть відображені тільки фай-

ли з цим розширенням. Тому, якщо ви точно знаєте, що конкретний файл лежить в цій папці, а у вікні вибору зображення ви його не бачите, це означає, що ви вибрали не той тип файлу.

Припустимо, у нас є дві фотографії (рис. 1.34).



Рис. 1.34. Фотографії для об'єднання

Ми хочемо, щоб кінь скакав морським узбережжям. Що нам необхідно зробити:

- створити новий файл для нового зображення (кінь на морському узбережжі);
- розмістити на різних шарах нового файлу зображення коня і моря;
- вирізати зображення коня. Для цього його необхідно спочатку виділити;
- додати природності новому зображенню.

Розберемо всі пункти покроково.

Створення нового файлу

Отже, створимо новий файл, розміром 500x375 (такі розміри фотографії з морем, хоча ми могли б зробити і менше зображення).

Створимо два нових шари: один назвемо *Horse*, інший – *Sea*, причому шар *Sea* повинен знаходитися під шаром *Horse*. Щоб шару дати назву, клікніть правою кнопкою миші по шару, у контекстному меню оберіть **Layer Properties** (Параметри шару). Відкриється вікно параметрів, де і потрібно задати ім'я.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Зараз у редакторі відкрито три файли, тобто три вікна з різними файлами (з морем, з конем і щойно створений, який будемо називати робочим вікном). Причому активним вікном (з яким працюємо) є останнє.

У ньому, на шарі *Sea* ми розташуємо наше море. Для цього:

- зробимо активним вікно з морем (просто клікніть по ньому);

- виділимо всі зображення, для цього в меню оберемо команду **Select** → **All** (Виділити → Усе) або натисніть комбінацію клавіш **<Ctrl> + <A>**. Навколо зображення з'явиться рамка виділення у вигляді мурашок, що біжать;

- скопіюємо це зображення (**<Ctrl> + <C>**). Тепер зробимо активним наше робоче вікно (клацнемо по ньому), переконаємося, що активним є шар *Sea* (якщо ні, то клацніть по ньому) і вставимо зображення (**<Ctrl> + <V>**). Тепер на шарі *Sea* зображено море.

Виконайте все те ж саме, щоб розташувати на шарі *Horse* зображення коня. Таким чином, у нас є обидві необхідні складові для майбутнього зображення. Тому файли з фотографіями моря і коня можна закрити.

Виділення зображення

У Photoshop передбачено кілька інструментів виділення зображення, один з них (**Прямокутна область**) ми розглядали раніше. Але цей спосіб не підійде, оскільки кінь має складну форму.

Для його виділення ми скористаємося двома інструментами на панелі інструментів (рис. 1.35).

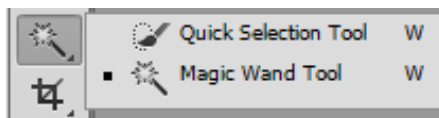


Рис. 1.35. Інструмент *Чарівна паличка*

Виберіть інструмент **Чарівна паличка**. Якщо клікнути по будь-якому пікселю шару цим інструментом, то будуть виділені прилеглі пікселів одного відтінку. На панелі параметрів цього інструменту оберіть наступні значення (рис. 1.36).

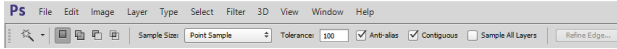


Рис. 1.36. Налаштування панелі параметрів для виділення

У поле **Tolerance** (Допуск) можна вводити значення від 0 до 255. Якщо встановити значення 0, то буде виділений тільки один відтінок, якщо встановити 32, то інструмент буде виділяти пікселі в діапазоні від кольору на 16 одиниць темніше і до кольору на 16 одиниць світліше того, за яким ви клацнули. У нашому випадку ми встановили значення 100.

Ми також установимо прапорець **Anti-alias** (Згладжування), щоб зробити більш гладкими межі виділення.

Прапорець **Contiguous** (Суміжний) виділяє тільки ті пікселі, які знаходяться в сусідніх з обраним областях.

Приберіть прапорець **Sample All Layers** (Зразки всіх шарів), тому що він поширює свою дію на всі шари, а ми працюємо тільки з шаром *Horse*.

Тепер клацніть інструментом по зображенню коня, він виділиться (рис. 1.37).



Рис. 1.37. Виділення об'єкта

Власне виділення не буде ідеальним (ми його доопрацюємо пізніше), на цьому етапі головне виділити контури коня. Якщо вам не подобається те, як виділилося, клацніть ще раз інструментом і виділення пропаде (або <Ctrl> + <Z>) і спробуйте знову.

Тепер, коли коня виділено, нам потрібно видалити все зайве. Для цього в меню виберемо команду **Select** → **Inverse** (Виділити → Інверсія). Тепер буде виділена вся область навколо коня. Натискаємо <Delete> на клавіатурі і отримуємо вирізаного коня на тлі моря (рис. 1.38).



Рис. 1.38. Інверсія виділення та видалення

Приберемо виділення (у меню оберемо команду **Select** → **Deselect** (Виділити → Зняти виділення)). Залишилось прибрати траву. Для цього спочатку збільшимо нижню частину зображення (там, де трава).

Отже, ви збільшили ту частину, де трава. Натисніть лівою клавішею миші на інструмент **Ласо**, відкриється вікно вибору інструмента (рис. 1.39).

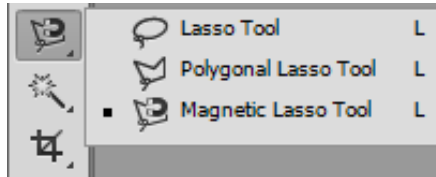


Рис. 1.39. Інструмент Ласо

Оберіть інструмент **Магнітне ласо**. Цей інструмент автоматично створює виділену область у той час, як ви переміщуєте курсор. Край області «прилипає» до найближчого відмінного відтінку кольору або тіні, який визначає межу фігури.

Клікніть по зображенню, щоб створити першу проміжну точку. Переміщайте курсор уздовж межі фігури, яку ви хочете виділити (у нашому випадку кінцівки коня). У той час як ви перемістили курсор, лінія виділення буде «прилипати» до межі фігури. Тимчасові точки, що з'являються в процесі переміщення курсора зникнуть, як тільки ви замкнете контур (рис. 1.40).



Рис. 1.40. Виділення за допомогою інструмента Ласо

Знову <Delete> на клавіатурі, прибрати виділення, зменшити розмір до нормального.

Для більшої переконливості додамо піску на копита коня. Для цього створіть новий шар, розмістіть його вгорі. Тепер візьміть інструмент **Піпетка** на панелі інструментів (рис. 1.41).

Практичний web-дизайн: проектування, створення та супроводження web-вузла



Рис. 1.41. Інструмент *Пінетка*

Цей інструмент вибирає колір з активного зображення або з будь-якої іншої області екрана і призначає його в якості основного або фонового. Нам потрібен колір піску, тому клікаємо по зображенню піска. Подивіться, основний колір змінився.

Тепер оберіть інструмент **Пензель** на панелі інструментів (рис. 1.42).

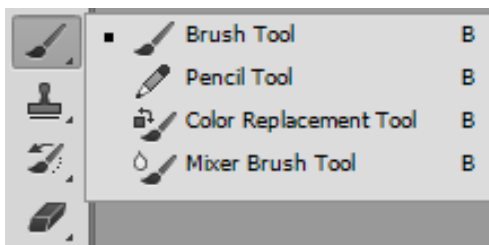


Рис. 1.42. Інструмент *Пензель*

На панелі параметрів цього інструмента виберіть **Пензель**, що нагадує пісок (тобто з нерівними краями) (рис. 1.43).

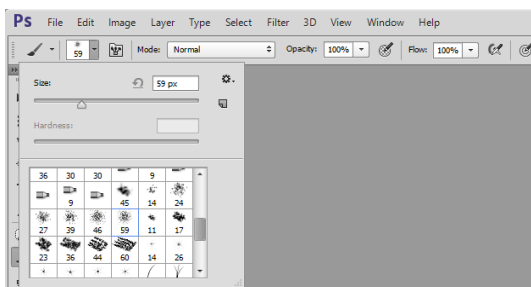


Рис. 1.43. Вибір форми пензля

Зробіть ним по одному кліку миші на кожному копиті (рис. 1.44).



Рис. 1.44. Кінцевий результат обробки фотографії

1.4.6 Трансформація

Отже, розглянемо докладніше можливості трансформації зображень. Трансформацію можна застосовувати до цілого зображення, до окремого шару цього зображення або до частини шару. Для початку потрібно виділити об'єкт, який будемо трансформувати:

- щоб трансформувати весь шар, зробіть його активним (клікніть по ньому на панелі **Шари**) і переконайтеся, що на зображенні немає виділених областей;

- щоб трансформувати частину шару, виділіть на ньому потрібну частину зображення;

- щоб трансформувати кілька шарів, виберіть їх, утримуючи клавішу <Ctrl>. На панелі **Шари** вони стануть активними (тобто синіми).

- щоб трансформувати все зображення, виконайте спочатку його зведення (**Layer** → **Flatten Image** (Шар → Виконати зведення зображення)).

Визначившись із об'єктом трансформації, виберіть команду **Edit** → **Transform** (Редагувати → Трансформувати) (рис. 1.45).

Практичний web-дизайн: проектування, створення та супроводження web-вузла

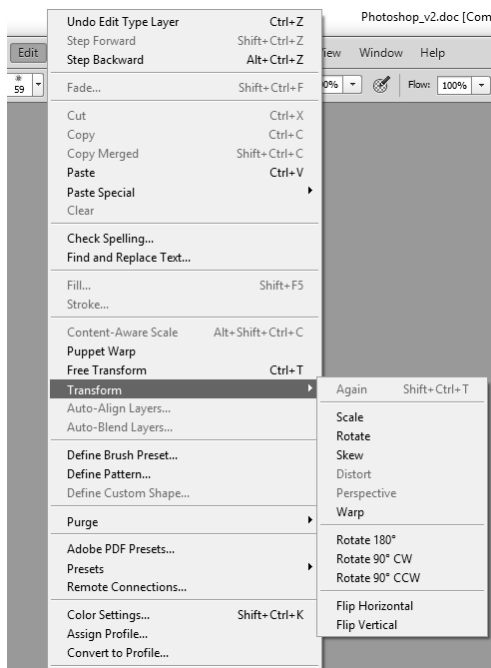


Рис. 1.45. Трансформація

Розглянемо пункти підменю:

Scale (Масштаб) – збільшення або зменшення розміру об'єкта. Масштабувати можна по горизонталі або по вертикалі, а також по горизонталі і по вертикалі одночасно (рис. 1.46).

Щоб зробити горизонтальне масштабування, необхідно підвести курсор миші до квадратиків на правій або лівій сторонах рамки. Курсор набуде вигляду горизонтальної стрілки. Утримуючи ліву кнопку миші, рухайте рамку праворуч або ліворуч (рис. 1.46, 2).

Щоб зробити вертикальне масштабування, необхідно підвести курсор миші до квадратиків на верхній або нижній стороні рамки. Курсор набуде вигляду вертикальної стрілки. Утримуючи ліву кнопку миші, рухайте рамку вгору або вниз (рис. 1.46, 3).

Щоб зробити одночасно вертикальне та горизонтальне масштабування, необхідно підвести курсор миші до квадратика на будь-якій вершині рамки. Курсор набуде вигляду діагональної двобічної стрілки. Утримуючи ліву кнопку миші, рухайте рамку по діагоналі вгору або вниз. Якщо хочете, щоб збереглися пропорції зображення, утримуйте при цьому клавішу <Shift> (рис. 1.46, 4).



Рис. 1.46. Різновиди масштабування

Rotate (Повернути) – поворот об’єкта навколо контрольної точки (рис. 1.47, 2). За замовчуванням ця точка знаходиться в центрі об’єкта. Підведіть курсор миші до квадратика на будь-якій вершині рамки. Курсор набуде вигляду зігнутої двобічної стрілки. Утримуючи ліву кнопку миші, повертайте рамку по колу вгору або вниз.

Skew (Нахил) – нахил об’єкта по вертикалі або по горизонталі (рис. 1.47, 3). Підведіть курсор миші до квадратика на будь-якій вершині рамки. Курсор набуде вигляду штрихованого трикутника. Утримуючи ліву кнопку миші, нахиліть рамку.



Рис. 1.47. Поворот та нахил

Distort (Викривлення) – розтягування об’єкта в усіх напрямках (рис. 1.48, 2).

Perspective (Перспектива) – до обраного об’єкта застосовується перспектива сходження в одній точці (рис. 1.48, 3).

Warp (Деформація) – зміна форми об’єкта. На панелі параметрів можна вибрати стиль деформації (рис. 1.48, 4).

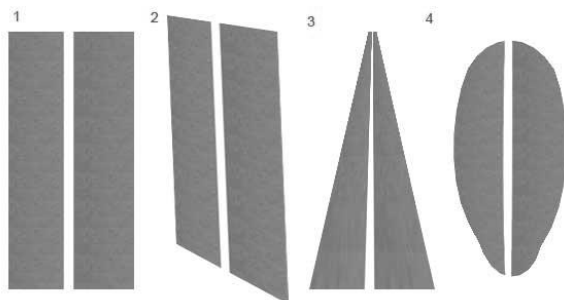


Рис. 1.48. Спотворення, перспектива, деформація

Rotate (Повернути на 180 °, на 90 ° за годинниковою стрілкою, на 90 ° проти годинникової стрілки) – повертає об’єкт відповідно на 90 ° або 180.

Flip Horizontal (Віддзеркалити за горизонталлю) – відображає об’єкт у горизонтальній площині (рис. 1.49, 2).

Flip Vertical (Віддзеркалити за вертикаллю) – відображає об’єкт у вертикальній площині (рис. 1.49, 3).

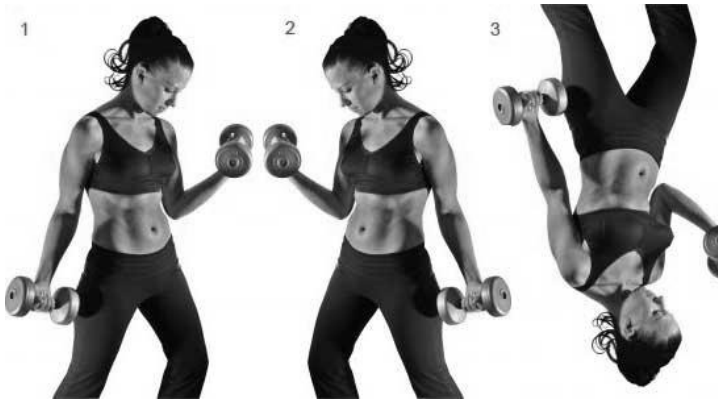


Рис. 1.49. Відображення рисунка

1.4.7 Інструменти ретушування

Пензель відновлення

Healing Brush Tool (Пензель відновлення) – дозволяє відновлювати ділянки фотографій за допомогою взятих за зразок збережених фрагментів (рис. 1.50). Він використовує пікселі за зразком зображення або візерунка і зіставляє їх текстуру, освітлення, прозорість і затінення з відповідними параметрами пікселів, що виправляються.

Обробимо зображення, що наведено на рис. 1.51.



Рис. 1.50. Інструмент Пензель відновлення



Рис. 1.51. Початкове зображення

Ми хочемо прибрати ластовиння (рис. 1.52). Для цього виконаємо такі кроки:

- візьмемо інструмент **Пензель відновлення**;
- підберемо на палітрі параметрів відповідний розмір пензля (наприклад, 10 пікселів);
- підведемо курсор миші до ділянки шкіри, де немає ластовинок (наприклад, на шиї);
- натиснемо клавішу **<Alt>** (курсор стане схожим на мішень) і клікнемо, щоб визначити область-джерело;
- відпустимо кнопку **<Alt>** і почнемо малювати по ластовинках. Пікселі з «джерела» будуть передаватися і підганяти за кольором ті, де ви малюєте пензлем.



Рис. 1.52. Оброблене фото

Так можна прибрати родимки, подряпини та інші дефекти.

Точковий пензель відновлення – дозволяє абсолютно автоматично ретушувати деякі недоліки фотографій. Дії точкового відновлювального пензля аналогічні діям відновлюва-

льного пензля, але, на відміну від нього, не потрібно вказувати точку, яка використовується в якості зразка. Точковий пензель відновлення автоматично вибирає зразки пікселів з області навколо фрагмента, що ретушується. Обробимо зображення, що наведене на рис. 1.53.



Рис. 1.53. Початкове зображення для
Точкового пензля відновлення

Ми хочемо прибрати родимки (рис. 1.54). Для цього виконаємо такі кроки:

- візьмемо інструмент **Точковий пензель відновлення**;

- підберемо на палітрі параметрів відповідний розмір пензля. Рекомендується обирати діаметр пензля дещо більше відновлюваної області, щоб її можна було покрити одним кліканням;

- виберемо режим **Замінити**, задамо тип пензля:

1. **Відповідність наближення** – по пікселях навколо кордону виділення знаходиться область зображення, яка підходить як латка для виділеної області. Якщо ця дія не привела до потрібного результату, скасуйте це виправлення і спробуйте виконати **Створення текстури**;

2. **Створення текстури** – по пікселях виділеної області створюється текстура для виправлення області;

- клікнемо по області, яку потрібно виправити.



Рис. 1.54. Оброблене фото інструментом
Точковий пензель відновлення

Латка

Латка – дозволяє відновлювати виділену область за допомогою пікселів іншої області (поєднання довільного виділення і заливки). Обробимо зображення, що наведене на рис. 1.55.



Рис. 1.55. Початкове зображення для *Латки*

Ми хочемо прибрати пліт із зображення (рис. 1.56). Для цього виконаємо такі кроки:

- візьмемо інструмент **Латка**;
- курсором миші виділимо область навколо плоту, потім на панелі параметрів виберемо «джерело»;

– перетягнемо межу виділеної області в ту область, з якої потрібно отримати зразок (тобто нижче і лівіше). При відпусканні кнопки миші спочатку виділена область буде заповнена пікселями зразка, тобто водою.



Рис. 1.56. Оброблене фото інструментом *Латка*

Якщо в кроці 2 на панелі параметрів вибрати **Призначення**, то перетягніть межу виділеної області в ту область, на яку слід поставити латку. При відпусканні кнопки миші знову виділена область буде заповнена пікселями зразка.

Латку можна поставити і у вигляді візерунка, якщо вибрати його на панелі параметрів як джерело.

Червоні очі – видаляє ефект «червоних очей», а також білі і зелені відблиски на фотографіях, знятих зі спалахом.

Для того щоб прибрати червоні очі, виконаємо такі кроки:

- візьмемо інструмент **Червоні очі**;
- клікнемо по області, де проявився ефект червоних очей;

– якщо не вдалося домогтися потрібного ефекту, скасуйте корекцію, потім установіть на панелі параметрів значення **Розмір зіниці** (збільшення або зменшення області, яка буде оброблятися інструментом) і **Величина затемнення** (налаштування справляє затемнення) і знову клацніть.

Штамп

Штамп – призначений для нанесення однієї частини зображення поверх іншої частини зображення (рис. 1.57). Обробимо зображення, що наведено на рис. 1.58.

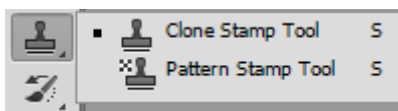


Рис. 1.57. Інструмент *Штамп*



Рис. 1.58. Початкове зображення для *Штампа*

Ми хочемо додати ще одного метелика (рис. 1.59). Для цього виконаємо такі кроки:

- візьмемо інструмент **Штамп**;
- на панелі параметрів виберемо потрібний пензель (у цьому випадку круглу, діаметром 150);
- підведемо курсор миші так, щоб метелику умістився в слід пензля, натиснемо **<Alt>** на клавіатурі і клацнемо мишкою. Зразок для клонування взято;
- перемістимо мишу туди, де буде клон метелика і клікнемо.



Рис. 1.59. Фото, оброблене інструментом *Штамп*

Гумка

Гумка – стирає надлишки нашої творчості із зображення. Просто виберіть цей інструмент і зітріть ним усе зайве.

Фонова гумка – призначений для відділення об'єкта від фону. Дуже зручний для контрастних зображень.

Чарівна гумка – гібрид **Гумки** і **Чарівної палички**. Головна відмінність інструмента **Чарівна гумка**, це швидкість роботи.

Якщо з **Фоновією гумкою** нам доводилося водити курсором миші для стирання фону, то в цьому випадку використовується лише клацання. Стирає інструмент все пікселі, колір яких схожий на колір точки, по якій ви клікнули, при цьому існують параметри, якими можна обмежувати його роботу.

Цим інструментом можна зробити вибраний колір пікселів частково прозорими, задавши числове значення параметра **Прозорість** нижче 100 %.

Параметр **Допуск** задає діапазон кольорів, що входять у спектр дії інструменту, чим вище числове значення, тим більше кольорів може стерти гумка. Щоб видалити пікселі тільки

одного кольору, потрібно задати числове значення, що дорівнює нулю.

Поставте прапорець у **Згладжування**, щоб зробити переходи між стертими областями і цілим зображенням плавними.

Поставте прапорець у **Суміжні**, щоб у спектр дії входили тільки близько розташовані точки.

1.4.8 Інструменти корекції зображень

Інструмент **Розмиття** (рис. 1.60)

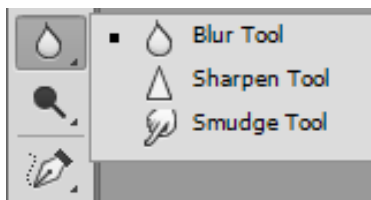


Рис. 1.60. Інструмент *Розмиття*

Blur Tool (Розмиття) – пом'якшує чіткі краї і зменшує деталізацію зображення. Чим більше цей інструмент застосовується до області, тим більш розмитою вона стає.

Sharpen Tool (Різкість) – підсилює контрастність уздовж країв, підвищуючи видиму різкість. Чим більше зазначений інструмент застосовується до області, тим більш різкою вона стає. Будьте обережні, якщо цей інструмент кілька разів поспіль застосувати до однієї ділянки зображення, можуть з'явитися просто кольорові точки, ніяк не відповідають вмісту зображення.

Узагалі різкість і розмиття зображень або їх частин краще регулювати відповідними фільтрами в меню **Фільтр**.

Smudge Tool (Палець) – імітує ефект розмазування свіжої фарби пальцем. Інструмент визначає колір у точці, де починається рух пальця, і «розмазує» його за напрямком руху.

Інструмент **Освітлювання** (рис. 1.61)

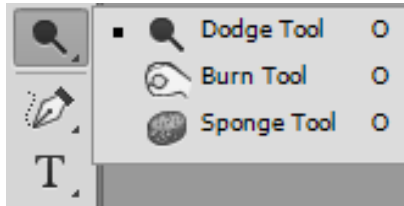


Рис. 1.61. Інструмент *Освітлювання*

Dodge Tool (Освітлювач) – освітлює область зображення.

Burn Tool (Випалювання) – затемнює область зображення.

Sponge Tool (Губка) – змінює насиченість кольору оброблюваної області. На панелі параметрів цього інструмента необхідно вибрати режим: збільшити насиченість або зменшити насиченість.

Принцип роботи в усіх інструментів однаковий: вибираємо інструмент, задаємо параметри на панелі параметрів і проводимо мишкою по необхідних областях.

РОЗДІЛ 2

МОВА РОЗМІТКИ ГІПЕРТЕКСТУ HTML

2.1 Основи HTML

HTML (*HyperText Markup Language*) – мова розмітки гіпертекстових документів.

Гіпертекст – розширений текст, що може містити в собі текст, ілюстрації, різноманітні вбудовані об’єкти і посилання на інші ресурси або на інші web-сторінки.

HTML-документ являє собою звичайний текстовий документ з керуючими конструкціями мови HTML – тегами, які і виконують «форматування» текстових блоків і здійснюють додавання різноманітних об’єктів у документ.

При передачі інформації через Інтернет використовуються HTML-сторінки – файли *.htm і *.html, оскільки вони містять у собі гіпертекст, то мають розмір менший за звичайні текстові або інші файли і для їх перегляду потрібен тільки браузер.

2.1.1 Структура документа

<HTML>	тег-контейнер документа *.html
<HEAD>	тег, який призначений для зберігання інших елементів, що необхідні браузеру для роботи з даними
<TITLE>	установлює заголовок вікна web-сторінки
<BODY>	призначений для зберігання вмісту (контенту) web-сторінки

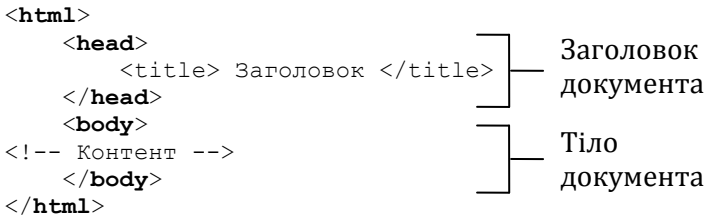


Рис. 2.1. Структура html-документа

Табл. 2.1

Елементи контейнера <HEAD>

Назва	Пояснення
<link>	встановлює зв'язок із зовнішнім документом <link rel="stylesheet" href="style.css">
<basefont>	встановлює фурнітуру шрифту за замовчуванням <basefont face="Arial" size="7" color="green">
<meta>	визначає метатеги, які використовуються для зберігання інформації, призначеної для браузерів та пошукових систем <meta charset="utf-8">
<script>	застосовується для опису скриптів <script> код </script>
<style>	застосовується для опису стилів <style> стилі </style>
<title>	визначає заголовок документа <title>Заголовок</title>

Атрибути контейнера <BODY>:

Синтаксис:

<body атрибут="значення">...</body>

Атрибути контейнера <BODY>

Назва	Пояснення
<i>alink</i>	встановлює колір активного посилання
<i>link</i>	встановлює колір посилання
<i>vlink</i>	встановлює колір відвіданого посилання
<i>background</i>	встановлює фоновий рисунок
<i>bgcolor</i>	встановлює колір фону
<i>text</i>	встановлює колір тексту
<i>leftmargin</i>	задає відступ по горизонталі від лівої границі вікна браузера до контенту
<i>topmargin</i>	задає відступ від верхньої границі вікна браузера до контенту
<i>scroll</i>	відображає полоси прокрутки

2.1.2 Форматування тексту

1. Теги логічного форматування

Тег <cite> – виділяє цитати, висловлювання і т. ін.

Тег <code> – форматує програмний код.

Тег – позначення видаленого тексту.

Тег – інтонаційне виділення тексту (візуально текст відображається так само, як форматування тегом <i>).

2. Теги фізичного форматування

2.1 Одинарні теги

 — перехід на новий рядок;

<hr/> — горизонтальна лінія.

Атрибути:

align – вирівнювання лінії;

color – колір лінії;

size – товщина лінії;

width – ширина лінії.

Приклад:

```
<hr align="center" width="300px" color="Red"/>
```

2.2 Функціональні блочні елементи

Теги **<h1>** ... **<h6>** задають розмір заголовку. Найменший розмір має заголовок **<h6>**, найбільший – заголовок **<h1>**.

```
<h1>Заголовок h1</h1>  
<h2>Заголовок h2</h2>  
<h3>Заголовок h3</h3>  
<h4>Заголовок h4</h4>  
<h5>Заголовок h5</h5>  
<h6>Заголовок h6</h6>
```

2.3 Параметри шрифту

Тег . Синтаксис:

```
<font size="значення" color="значення" face="значення">  
Текст </font>
```

Табл. 2.3

**Атрибути **

Назва	Пояснення
<i>face</i>	задає гарнітуру шрифтів
<i>size</i>	задає розмір шрифту в умовних одиницях від 1 до 7 (за замовчуванням 3)
<i>color</i>	задає колір тексту

Приклад:

```
<font size="5" color="red" face="Verdana, sans-serif">  
Текст</font>
```

Колір тексту може бути заданий у форматах:

Формат	Приклад
#hex – шістнадцяткова константа	#F0FFFF0
name – символічне ім'я	Honeydew
rgb	rgb(240, 255, 240)
rgba	rgba(240, 255, 240, 0.1)

Практичний web-дизайн: проектування, створення та супроводження web-вузла

У функції *rgba* останній параметр – ступінь прозорості вказаного кольору.

2.4 Форматування тексту

Табл. 2.4

Назва	Пояснення
<code>Текст</code>	жирний текст
<code><i>Текст</i></code>	курсивний текст
<code><u>Текст</u></code>	підкреслений текст
<code><sup>Текст</sup></code>	верхній індекс
<code><sub>Текст</sub></code>	нижній індекс
<code><s>Текст</s></code>	закреслений текст

Приклад:

```
<b>Формула води:</b>  
<i>Н<sub><small>2</small></sub>О</i>
```

2.1.3 Параграф

Тег <p>. Синтаксис:

```
<p align="значення"> ... </p>
```

Додає новий параграф, за замовчуванням вирівняний за лівою границею. Перед та після кожного параграфа автоматично додаються відступи.

Атрибут *align* – задає вирівнювання та має значення: *left*, *center*, *right*, *justify*.

Приклад:

```
<p>Параграф 1</p>  
<p align="left">Параграф 2</p>  
<p align="right">Параграф 3</p>
```

2.1.4 Зображення

Тег . Синтаксис:

```

```

Призначений для відображення рисунків у графічних форматах GIF, JPEG або PNG.

Табл. 2.5

**Атрибути **

Назва	Пояснення
<i>src</i>	шлях до графічного файлу
<i>align</i>	задає вирівнювання і спосіб обтікання текстом
<i>alt</i>	альтернативний текст для зображення
<i>border</i>	товщина рамки навколо зображення
<i>hspace</i>	горизонтальний відступ від рисунка до оточуючого контенту
<i>vspace</i>	вертикальний відступ від рисунка до оточуючого контенту
<i>width</i>	ширина рисунка
<i>height</i>	висота рисунка

Адресація:

1. Абсолютна адреса рисунка

```

```

2. Адреса рисунка відносно кореня сайту

```

```

3. Адреса рисунка відносно поточного HTML-документа

```

```

Табл. 2.6

Вирівнювання зображень

Значення атрибуту align	Пояснення
<i>top</i>	верхня границя рисунка вирівнюється за найвищим елементом поточного рядка
<i>middle</i>	вирівнювання середини рисунка за базовою лінією поточного рядка
<i>absmiddle</i>	вирівнювання середини рисунка відносно середини поточного рядка
<i>baseline</i>	вирівнювання рисунка за базовою лінією батьківського елемента
<i>bottom</i>	вирівнювання нижньої границі рисунка за текстом, що його оточує
<i>left</i>	вирівнювання рисунка за лівою границею
<i>right</i>	вирівнювання рисунка за правою границею

Приклад:

```
<h1 align="center">Заголовок</h1>  
<p>Текст абзацу </p>
```

2.1.5 Списки

Нумеровані списки призначені для перелічення елементів, що слідуєть у чітко визначеному порядку.

Марковані списки призначені для перелічення елементів, що слідуєть у довільному порядку.

Списки визначень призначені для перелічення елементів без маркерів, але з відступами.

1. Нумеровані списки

Тег . Синтаксис:

```
<ol>  
  <li>Рядок списку </li>  
</ol>  
ol – Ordered Lists
```

Табл. 2.7

**Атрибути **

Назва	Пояснення
<i>type</i>	задає вид маркера списку
<i>reversed</i>	нумерація здійснюється у зворотному напрямку
<i>start</i>	задає число, з якого починається нумерація елементів списку

Приклад:

```
<ol>  
  <li>пункт 1</li>  
  <li>пункт 2</li>  
  <li>пункт 3</li>  
</ol>  
  
<ol type="I">  
  <li>пункт 1</li>  
  <li>пункт 2</li>  
  <li>пункт 3</li>  
</ol>  
  
<ol start="15">  
  <li>пункт 1</li>  
  <li>пункт 2</li>  
  <li>пункт 3</li>  
</ol>
```

2. Марковані списки

Тег . Синтаксис

 Рядок списку

ul – UnOrdered Lists

Значення атрибута **type**: *disc* – зафарбоване коло; *circle* – незафарбоване коло; *square* – зафарбований квадрат.

Приклад:

```
<ul type="circle">
  <li>пункт 1</li>
  <li type="disc">пункт 2</li>
  <li type="square">пункт 3</li>
</ul>
```

3. Багаторівневий список

Багаторівневий список формується шляхом вкладання одного списку в тіло іншого.

Приклад:

```
<ol>
  <li>пункт 1</li>
  <li>пункт 2
    <ul>
      <li>Підпункт 2.1</li>
      <li>Підпункт 2.2
        <ul>
          <li>Підпункт 2.2.1</li>
          <li>Підпункт 2.2.2</li>
        </ul>
      </li>
      <li>Підпункт 2.3</li>
    </ul>
  </li>
  <li>пункт 3</li>
</ol>
```

4. Списки визначень

Тег <dl>. Синтаксис:

```
<dl>  
  <dt>Термін 1</dt>  
  <dd>Визначення терміна 1</dd>  
  <dt>Термін 2</dt>  
  <dd>Визначення терміна 2</dd>  
</dl>
```

Табл. 2.8

Список визначень

Назва	Пояснення
<dl>	початок/кінець списку
<dt>	початок/кінець терміна
<dd>	початок/кінець пояснювального тексту терміна

Теги <dt> та <dd> не обов'язково чергувати.

2.1.6 Гіперпосилання

Посилання HTML дозволяють зв'язати елемент сторінки (текст, рисунок і т. ін.) з гіпертекстовими документами або іншими ресурсами.

Тег <a>. Синтаксис:

```
<a href="URL" target="Вікно" title="Підказка">Текст по-  
силання</a>
```

Табл. 2.9

Атрибути <a>

Назва	Пояснення
<i>href</i>	адреса документа
<i>target</i>	визначає, в якому вікні буде завантажено файл. (<i>_top</i> – у всьому вікні браузера; <i>_blank</i> – в новому вікні браузера; <i>_self</i> – у вікні, яке містить посилання)
<i>title</i>	текст підказки

Приклад 1. Абсолютні посилання

```
<a href="https://www.google.ua"> Google </a>
```

Приклад 2. Відносні посилання

```
<a href="images/pic.jpg">Картинка</a>
```

Приклад 3. Графічний файл як посилання

```
<a href="page1.htm">  
    
</a>
```

Посилання всередині сторінки

1. Створити закладку в місці, куди необхідно здійснити перехід:

```
<a name="Ім'я закладки"></a>
```

2. Розмістити посилання на закладку

2.1 у тому ж документі:

```
<a href="#Ім'я закладки">Перейти на початок докумен-  
та</a>
```

Ім'я посилання на закладку починається із символу #, після якого вказуємо ім'я закладки.

2.2 в іншому документі:

```
<a href="Ім'я документа/#Ім'я закладки"> Перейти до  
розділу 1 </a>
```

Приклад внутрішнього посилання:

```
<a name="top"></a> Початок документа  
...  
<a href="#top">Нагору</a> Кінець документа
```

2.1.7 Таблиці

Тег <table>. Синтаксис:

```
<table>  
  <tr>  
    <td>...</td>  
  </tr>  
</table>
```

Додавання рядків

```
<tr> ..... </tr>
```

Додавання комірок у рядок

Практичний web-дизайн: проектування, створення та супроводження web-вузла

`<td>... </td>`

Додавання комірок-заголовків

`<th>... </th>`

Приклад:

```
<table>
  <tr>
    <th>Заголовок 1</th> <th >Заголовок 2</th>
  </tr>
  <tr>
    <td>Комірка 1</td> <td> Комірка 2</td>
  </tr>
</table>
```

Табл. 2.10

Атрибути <table>

Атрибут	Пояснення
<i>align</i>	вирівнювання таблиці (<i>left, right, center</i>)
<i>background</i>	фоновий рисунок
<i>bgcolor</i>	колір фону таблиці
<i>border</i>	товщина рамки в пікселях
<i>bordercolor</i>	колір рамки
<i>bordercol-ordark</i>	тінь рамки
<i>cellpadding</i>	відстань від границі комірки до її вмісту
<i>cellspacing</i>	відстань між комірками
<i>nowrap</i>	заборона переносу рядків у тексті
<i>frame</i>	установлює тип рамки таблиці

Якщо комірка не містить даних, то вона не буде мати границь. Якщо необхідно, щоб комірка мала границі, то вона має щось у собі містити, хоча б тег `
` або ` `.

Приклад:

```
<table align="center" cellpadding="5px" nowrap
frame="hsides">
  <tr>
    <td>Комірка 1</td> <td>Комірка 2</td>
  </tr>
</table>
```

Заголовок таблиці

Тег <caption>. Синтаксис

```
<table>
  <caption>Заголовок таблиці</caption>
  <tr>
    <td>...</td>
  </tr>
</table>
```

Табл. 2.11

Теги <td> , <th> . Атрибути

Атрибут	Пояснення
<i>align</i>	задає горизонтальне вирівнювання вмісту комірки (<i>left, center, right</i>)
<i>valign</i>	задає вертикальне вирівнювання вмісту комірки (<i>top, middle, bottom</i>)
<i>height</i>	висота комірки
<i>colspan</i>	об'єднання комірок у рядку
<i>rowspan</i>	об'єднання комірок у стовпчику
<i>width</i>	ширина комірки

Приклад:

```
<table border width="100%">
  <tr height="75px">
    <td align="left" valign="top"> комірка1</td>
    <td align="right" valign="bottom"> комірка 2</td>
  </tr>
</table>
```

Об'єднання стовпчиків

Синтаксис:

```
<th colspan="число">...</th>
    або
<td colspan="число">...</td>
```

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Приклад:

```
<table width="200px" border="1px" align="center"
cellspacing="5px">
  <tr bgcolor="#F0FFFF">
    <th colspan="2">Комірка 1</th>
  </tr>
  <tr bgcolor="#FFF5EE">
    <td>Комірка 2</td> <td>Комірка 3</td>
  </tr>
</table>
```

Об'єднання рядків

Синтаксис:

```
<th rowspan="число">...</th>
```

Приклад:

```
<table>
  <tr>
    <th rowspan="2">Комірка 1 </th> <td>Комірка 2</td>
  </tr>
  <tr>
    <td>Комірка 3</td>
  </tr>
</table>
```

2.1.8 Рядкові та блокові елементи

Блокові елементи відображаються на web-сторінці у вигляді прямокутника і займають всю доступну ширину батьківського елемента (*div, form, ol, ul, table, p* і т. ін.). Для групування блокових елементів застосовується тег `<div>`.

Рядкові елементи є частиною іншого елемента (*a, font, span* і т. ін.). Для групування рядкових елементів застосовується тег ``.

Рядкові	Блокові
можуть містити лише дані або інші рядкові елементи;	можуть містити як рядкові так і блокові елементи;
можуть починатись у будь-якому місці рядка;	завжди починаються з нового рядка;
ширина визначається вмістом елемента плюс значення відступів та границь	займають всю доступну ширину контейнера

2.1.9 Форми

Елемент `<form>` встановлює форму на web-сторінці.

Тег `<form>`. Синтаксис:

`<form > ... </form>`

Табл. 2.12

Атрибути `<form >`

Атрибут	Пояснення
<i>name</i>	ім'я форми
<i>action</i>	визначає URL, за яким буде відправлено вміст форми
<i>method</i>	визначає спосіб відправки вмісту форми
<i>enctype</i>	визначає спосіб кодування вмісту форми
<i>accept-charset</i>	визначає кодування, в якому сервер може приймати та обробляти дані форми
<i>autocomplete</i>	задає або відключає автозаповнення форми
<i>novalidate</i>	скасовує валідацію даних форми

Елементи форми

1. Елемент `<Textarea>`

Створює багаторядкове поле введення даних (тексту).

Синтаксис:

`<textarea атрибут="значення">`

текст

`</textarea>`

Табл. 2.13

Атрибути `<Textarea>`

Атрибут	Пояснення
<i>name</i>	ім'я елемента
<i>rows</i>	кількість рядків тексту
<i>cols</i>	ширина текстового поля
<i>readonly</i>	поле доступне тільки для читання
<i>disabled</i>	поле є неактивним

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Приклад:

```
<form>  
  <textarea name="txt" cols="40" rows="3" readonly> текст  
</textarea>  
</form>
```

2. Елемент <INPUT>

Поле призначено для введення інформації користувачем.

Синтаксис:

<input атрибут="значення">

Табл. 2.14

Атрибути <INPUT>

Атрибут	Пояснення
<i>maxlength</i>	визначає максимальну кількість символів, які вводяться в поле
<i>name</i>	ім'я поля
<i>size</i>	ширина текстового поля
<i>type</i>	задає тип елемента форми (за замовчуванням "text")
<i>value</i>	зберігає значення елемента

Табл. 2.15

Значення атрибута *type*

Значення	Пояснення
<i>button</i>	кнопка
<i>checkbox</i>	прапори; дозволяють обирати декілька з множини варіантів

Закінчення табл 2.15

<i>radio</i>	перемикачі; дозволяють обирати один із множини варіантів
<i>reset</i>	кнопка, що повертає дані форми в первісний стан
<i>submit</i>	кнопка для відправлення даних форми на сервер
<i>text</i>	поле для введення тексту

Приклад:

```
<form>
  <input type="checkbox" name="c1" checked/> Зелений <br/>
  <input type="checkbox" name="c1"/> Синій <br/>
  <input type="checkbox" name="c1" checked/> Червоний <br/>
</form>
```

3. Кнопки

Види кнопок:

- **submit** – кнопка відправки вмісту форми на web-сервер;
- **reset** – кнопка, що відновлює всі значення за замовчуванням на формі;
- **button** – довільна кнопка.

Синтаксис:

```
<button атрибут="значення"> .... </button>
```

Атрибути <button>:

type – тип кнопки (*reset, submit, button*);

name – ім'я кнопки;

value – напис на кнопці.

Приклад:

```
<form>
  Поле 1: <input name="p1"/> <BR/>
  Поле 2: <input name="p2" size="8" maxlength="10"/><BR/>
  <input type="reset"/>
  <input type="submit"/>
</form>
```

Приклад:

```
<form>
  <button name="but">ОК</button>
</form>
```

4. Елемент <SELECT>

Елемент <SELECT> створює меню вибору одного з декількох пунктів із переліку можливих.

Синтаксис

```
<select атрибут="значення">  
  <option атрибут="значення"/>Пункт 1  
  <option атрибут="значення"/>Пункт 2  
</select>
```

Табл. 2.16

Атрибути <Select>

Атрибут	Пояснення
<i>multiple</i>	дозволяє вибір декількох пунктів меню
<i>name</i>	ім'я меню
<i>size</i>	задає кількість видимих пунктів меню
<i>option</i>	описує пункти меню

Табл. 2.17

Атрибути <Option>

Атрибут	Пояснення
<i>selected</i>	визначає обраний пункт меню
<i>value</i>	задає обраному пункту значення

Приклад:

```
<form>  
  <select name="menu" size="1">  
    <option selected value="1"/>Перший пункт  
    <option value="2"/>Другий пункт  
    <option value="3"/>Третій пункт  
    <option value="4"/>Четвертий пункт  
  </select>  
</form>
```

2.1.10 Навігаційні карти

1. Розмістити зображення:

```

```

2. Додати карту зображення:

```
<map name="panel"> ... </map>
```

Тег <area>. Атрибути:

href – шлях до документа;

shape – форма області рисунка, яка є посиланням:

rect – прямокутна область;

poly – багатокутник;

circle – область задана колом;

coords – координати форми.

Приклад:

```

<map name="карта">
<area shape="rect" coords="0,0,100,100" href="1.html"
alt="Yellow" title="Yellow zone"/>
<area shape="rect" coords="101,0,201,100" href="2.html"
alt="Red" title="Red zone"/>
<area shape="rect" coords="201,0,301,100" href="3.html"
alt="Blue" title="Blue zone"/>
<area shape="rect" coords="201,101,301,200" href="4.html"
alt="Green" title="Green zone"/>
```

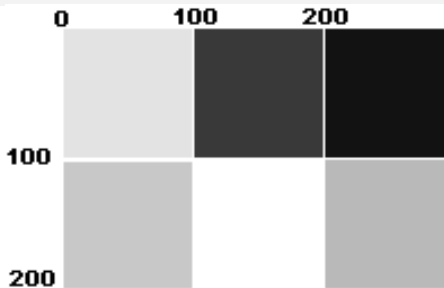


Рис. 2.2. Файл map_example.png

2.2 Основи Html5

2.2.1 Структурні елементи

У HTML5 для кожної частини сторінки визначено власний елемент:

Тег **<header>** – визначення верхньої секції на сторінці.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Тег **<nav>** – блок навігації.

Тег **<section>** – визначення секцій (опис визначеного блоку тексту).

Тег **<article>** – виділення визначеної частини тексту.

Тег **<footer>** – визначення нижньої секції на сторінці.

Приклад:

```
<section>
  <nav>
    <a href="URL шлях">Посилання 1</a>
    <a href="URL шлях">Посилання 2</a>
  </nav>
</section>
<section>
  <article>
    <h1>Заголовок статті 1</h1>
    <p>Текст статті 1</p>
  </article>
  <article>
    <h1>Заголовок статті 2</h1>
    <p>Текст статті 2</p>
  </article>
</section>
<footer>
  <!--Контент-->
</footer>
```

2.2.2 Елементи рівня блока

HTML5 визначає три рівня блоку:

Тег **<aside>** – виділяє неосновний текст всередині основного.

Тег **<figure>** – встановлює зв'язок між підписом та відповідним зображенням.

Приклад:

```
<figure>
  <figcaption>Підпис рисунка</figcaption>
  
</figure>
```

```
<figure>
  <legend> Підпис рисунка</legend>
  
</figure>
```

2.2.3 Елементи рівня тексту

Тег **<mark>** – виділення визначеної частини тексту.

Тег **<meter>** – представлення чисел у визначеному форматі.

Тег **<progress>** – представлення стану.

Тег **<time>** – виділення часу та дати.

Приклад:

```
<div>Завантажено:
  <progress value="50" max="100">50%</progress>
</div>
```

2.2.4 Інтерактивні елементи

Елемент **<details>** – містить інформацію, що є необов'язковою для відображення.

Приклад:

```
<details>
  <summary>Заголовок</summary>
  <ul>
    <li>Пункт 1</li>
    <li>Пункт 2</li>
    <li>Пункт 3</li>
  </ul>
</details>
```

2.2.5 Таблиці

У HTML5 введено розділення заголовка, футера і тіла таблиці – елементи **<thead>**, **<tbody>**, **<tfoot>**.

Приклад:

```
<table>
  <caption>Заголовок таблиці</caption>
```

Практичний web-дизайн: проектування, створення та супроводження web-вузла

```
<thead>
  <tr>
    <th>Заголовок 1</th> <th>Заголовок 2</th>
  </tr>
</thead>
<tbody>
  <tr>
    <td>Комірка 1</td> <td>Комірка 2</td> </tr>
</tr>
</tfoot>
</table>
```

2.2.6 iframe

Тег **<iframe>** створює плаваючий фрейм, який знаходиться всередині звичайного документа. Він дозволяє завантажувати на сторінку будь-які інші документи.

Тег <iframe>. Синтаксис:

```
<iframe src="URL шлях">...</iframe>
```

Приклад:

```
<iframe src="page1.html" width="400px" height="400px">
</iframe>
```

2.2.7 Мультимедіа

Елементи мультимедіа:

Тег **<audio>** – додає аудіофайли.

Тег **<video>** – додає відеофайли.

Тег <audio>. Синтаксис:

```
<audio src="URL"> </audio>
```

або

```
<audio>
  <source src="URL шлях"/>
</audio>
```

Табл. 2.18

Тег <AUDIO>. Атрибути

Атрибут	Пояснення
<i>src</i>	шлях до файлу
<i>controls</i>	панель управління
<i>autoplay</i>	автоматичне відтворення файлу після завантаження web-сторінки
<i>loop</i>	кількість повторів відтворення файлу

Приклад:

```
<audio controls>
  <source src="file1.mp3" type="audio/mpeg"/>
</audio>
<audio src="file2.mp3" autoplay="autoplay" loop="3">
</audio>
```

Тег <video>. Синтаксис:

```
<video>
  <source src="URL шлях">
</video>
```

Табл. 2.19

Тег <VIDEO>. Атрибути

Атрибут	Пояснення
<i>autoplay</i>	автоматичне відтворення файлу після завантаження web-сторінки
<i>controls</i>	панель управління
<i>height</i>	висота області для відтворення відеофайлу
<i>loop</i>	повторення відтворення відео
<i>poster</i>	адреса зображення, яке буде відображатися до завантаження відео
<i>preload</i>	завантаження відео при завантаженні web-сторінки
<i>src</i>	шлях до відеофайлу
<i>width</i>	ширина області для відтворення файлу

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Приклад:

```
<video width="400px" height="300px" controls  
poster="pic.jpg">  
    <source src="file.mp4" type='video/mp4; codecs =  
"avc1.42E01E, mp4a.40.2"'>  
</video>
```

2.2.8 Форми

У HTML5 введені нові атрибути елемента <INPUT>

Табл. 2.20

Тег <INPUT>. Атрибути

Атрибут	Пояснення
<i>autocomplete</i>	вмикає або вимикає автозаповнення
<i>autofocus</i>	встановлює фокус в поле форми
<i>list</i>	вказує на список варіантів, які можна вибирати при введенні даних
<i>placeholder</i>	виводить текст-підказку
<i>required</i>	обов'язкове поле для заповнення

Табл. 2.21

Значення атрибута *type*

Значення	Пояснення
<i>file</i>	поле для вибору файлу
<i>hidden</i>	приховане поле
<i>image</i>	кнопка із зображенням
<i>password</i>	поле для введення пароля
<i>color</i>	віджет для вибору кольору
<i>date</i>	поле для вибору календарної дати (у форматі дд.мм.рррр)
<i>date-time</i>	введення дати та часу (час у форматі гг:хх)
<i>email</i>	введення адреси електронної пошти
<i>number</i>	поле введення чисел
<i>range</i>	повзунок для вибору чисел у вказаному діапазоні

<i>search</i>	поле для пошуку
<i>tel</i>	поле введення телефонних номерів
<i>time</i>	поле введення часу
<i>url</i>	поле введення web-адреси
<i>month</i>	вибір місяця (місяць у форматі місяць-рік)
<i>week</i>	вибір тижня (тиждень у форматі тиждень-рік)

Приклад:

```
<input type="color"/>
<input type="text" placeholder="введіть текст"/>
<input type="text" required/>
<input type="tel" pattern="8[0-9]"/>
<input type="file" name="file" multiple/>
```

Mitku label

Тег `<label>` встановлює зв'язок між визначеною міткою, у якості якої зазвичай виступає текст, і елементом форми (`<input>`, `<select>`, `<textarea>`).

Усі елементи форми можна зв'язати з відповідними текстовими підписами за допомогою елемента `<label>`.

Тег `<label>`. Синтаксис:

```
<input id="ідентифікатор"/>
<label for="ідентифікатор">Текст</label>
```

Для зв'язку із полем введення мітка має атрибут **for**, який вказує на **id** поля введення

Приклад:

```
<label for="load">Файл: </label>
<input type="file" size="40"/>
```

РОЗДІЛ 3

КАСКАДНІ ТАБЛИЦІ СТИЛІВ CSS

CSS (*Cascading Style Sheets*) – каскадні таблиці стилів – це набір стильових описів для різних елементів HTML (теги, ідентифікатори, класи). CSS дозволяє суттєво розширити можливості HTML.

Стилі зберігаються в таблицях стилів, які визначені або в HTML-документі, або в окремому файлі з розширенням .css.

Можливості CSS 1	Фурнітура шрифту Кольори Атрибути тексту Вирівнювання тексту, зображень, таблиць та інших елементів Властивості блоків
Можливості CSS 2	Блокова верстка Можливість задавати різні стилі для різних носіїв Звукові таблиці стилів Вказівники
Можливості CSS 3	Додавання тіні елементам (<i>Box-shadows</i>) Додавання тіні тексту (<i>Text shadows</i>) Фони із декількох зображень (<i>Multiple backgrounds</i>) Рамка із зображень (<i>Border images</i>) Рівні прозорості (<i>Opacity levels</i>) RGBA coloring Закруглені кути (<i>Rounded corners</i>) Анімація Трансформація Уведення змінних (<i>variables</i>)

3.1 Синтаксис CSS

У загальному вигляді базовий синтаксис CSS є таким:

селектор {властивість: значення;}

Якщо в якості селектора вказано символ * – правила стилю поширюються на всі елементи.

У якості селектора можуть виступати різноманітні елементи HTML – теги, ідентифікатори, класи, групи елементів.

Приклад:

```
p {color: red;} /*задано колір абзацу*/
p a{color: red;} /*задано колір тексту для посилань, якщо вони
розташовані в середині абзацу*/
h1, h2, h3 {background-color: red;}
/*задано колір фону заголовків h1, h2 та h3*/
```

Колір можна задавати у форматах:

rgb

```
h4 {background-color: rgb(255, 248, 220);}
```

rgba

```
h4 {background-color: rgba(255, 248, 220, 0.5);}
```

шістнадцятковою константою

```
h4 {background-color: #FFF8DC;}
```

символьним ім'ям

```
h4 {background-color: Cornsilk;}
```

3.2 Підключення CSS

Існує декілька способів підключення таблиць стилів.

1. Атрибут **style**.

Приклад:

```
<head>
<title>Приклад</title>
</head>
```


Практичний web-дизайн: проектування, створення та супроводження web-вузла

```
<body>
  <p style="color:#FF0000;">Текст абзацу - червоний</p>
</body>
```

2. Тер style.

Приклад:

```
<head>
  <title>Приклад</title>
  <style>
    p {color:#FF0000;}
  </style>
</head>
<body>
  <p>Текст абзацу - червоний</p>
</body>
```

3. Підключити зовнішній файл із розширенням *css*.

Припустимо, що коренева директорія сайту має таку структуру:



Рис. 3.1. Приклад кореневої директорії сайту

Тоді підключення зовнішньої таблиці стилів продемонстровано таким прикладом.

Приклад. Текст файлу index.html

```
<head>
  <title> Приклад </title>
  <link rel = "stylesheet" href= "css/style.css">
</head>
<body>
  <p> Текст абзацу - червоний </p>
</body>
```

Приклад. Текст файлу style.css

```
p {color:#FF0000;}
```

Декілька HTML-документів можуть посилатись на одну й ту саму таблицю стилів. Один HTML-документ може містити підключення декількох таблиць стилів, для підключення кожної таблиці стилів використовується окремий тег <link>.

3.3 Колір і фон

Табл. 3.1

Властивості кольору та фону

Властивість	Пояснення
<i>color</i>	колір переднього плану елемента
<i>background-color</i>	колір фону елемента
<i>background-image</i>	фонове зображення
Повторення фонового зображення	
<i>background-repeat: repeat-x</i>	по горизонталі
<i>background-repeat: repeat-y</i>	по вертикалі
<i>background-repeat: repeat</i>	по горизонталі та вертикалі
<i>background-repeat: no-repeat</i>	не повторюється
Блокування фонового зображення	
<i>background-attachment: scroll</i>	зображення прокручується разом зі сторінкою
<i>background-attachment: fixed</i>	зображення фіксоване, прокручується тільки контент
Розташування фонового зображення	
<i>background-position: h v</i>	h – положення по горизонталі (<i>left, center, right</i>); v – положення по вертикалі (<i>top, center, bottom</i>). Значення можуть бути вказані в абсолютних або відносних одиницях

Порядок властивостей:

background-color | *background-image* | *background-repeat* |
background-attachment | *background-position*

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Приклад:

```
body {
background-image: url("fon.jpg");           /*фонова картинка*/
background-repeat: no-repeat;              /*не повторюється*/
background-attachment: fixed;              /*зафіксована*/
background-position: 100px 200px;         /*відступ від лівої грани-
ці 100px, зверху 200px */}
```

Скорочений запис:

```
body {background-image: url("fon.jpg") no-repeat fixed
100px 200px;}
```

3.4 Шрифти

Табл. 3.2

Форматування шрифтів

Властивість	Пояснення
<i>font-style</i>	стиль шрифту (<i>normal</i> , <i>italic</i> або <i>oblique</i>)
<i>font-variant</i>	вибір між значеннями <i>normal</i> та <i>small-caps</i>
<i>font-weight</i>	вага шрифту; властивість описує ступінь товщини шрифту (<i>normal</i> або <i>bold</i>); підтримуються числові значення
<i>font-size</i>	розмір шрифту
<i>font-family</i>	вказує пріоритетний список шрифтів

При визначенні значення властивості **font-family** рекомендовано вказувати список пріоритетних шрифтів. Якщо перший зі списку шрифт відсутній, то застосовується наступний з переліку, доки шрифт не буде застосовано. Останнім рекомендовано вказувати родове сімейство шрифтів, наприклад, sans-serif.

Порядок властивостей:

font-style | *font-variant* | *font-weight* | *font-size* | *font-family*

Приклад:

```
h3 {font-size: 120 %;} /*розмір шрифту збільшено на 20 % */
p {
```

```
font-family: "Times New Roman", serif;  
font-style: italic;  
font-weight: bold;  
font-size: 12pt;  
}
```

Скорочений запис:

```
p {font: italic bold 12pt "Times New Roman", serif; }
```

3.5 Форматування тексту

Табл. 3.3

Форматування тексту

Властивість	Пояснення
<i>text-indent</i>	встановлює абзацний відступ
<i>text-align</i>	встановлює вирівнювання тексту (<i>left, right, center, justify</i>)
<i>text-decoration</i>	декорування тексту (<i>underline, overline, line-through</i>)
<i>letter-spacing</i>	указує інтервал між літерами (значення >0 – текст розряджений; значення <0 – текст ущільнений (стиснутий))
<i>text-transform</i>	задає регістр символів (<i>capitalize, uppercase, lowercase</i>)

Приклад:

```
h1 {  
text-align: center; /*текст вирівняно по центру*/  
text-decoration: underline; /*текст підкреслений*/  
letter-spacing: 6px; /*текст розряджений на 6px */  
text-transform: capitalize; /*кожне слово з великої літери*/  
}  
p {  
text-indent: 30px; /*задано абзацний відступ на 30px */  
}
```

3.6 Псевдокласи

Псевдокласи – властивості, що дозволяють змінювати стиль елемента залежно від дій користувача, або положення елемента.

Синтаксис

селектор: псевдоклас {властивість : значення;}

Табл. 3.4

Псевдокласи

Псевдоклас	Пояснення
<i>hover</i>	задає стиль елемента, на який наведено курсор миші
<i>link</i>	задає стиль посилань на ресурси, які користувач ще не відвідував
<i>active</i>	задає стиль активного посилання
<i>visited</i>	задає стиль посилань на ресурси, які користувач вже відвідував
<i>first-child</i>	задає стиль першого дочірнього елемента в батьківському контейнері
<i>lang</i>	визначає мову
<i>focus</i>	визначає стиль елемента, що перебуває у фокусі

Приклад:

```
tr {  
    background-color: #CFCFCF;  
    color: #CDB5CD;  
    font-size: 12pt;  
}  
tr:hover {  
    background-color: #EE9A49;  
    color: #FFFAFA;  
    font-size: 150%;  
    text-align: center;  
}
```

```
a {color: #F0FFF0;}
a:link {color: #2F4F4F;}
a:visited {color: #66CDAA;}
a:active {background-color: #E0FFFF;}
a:hover { letter-spacing: 5px;}
```

3.7 Псевдоелементи

Псевдоелементи – властивості, що дозволяють встановлювати/змінювати стилі для окремої частини елемента.

Табл. 3.5

Псевдоелементи

Псевдоелемент	Пояснення
<i>first-letter</i>	задає стиль першої літери текстового блоку
<i>first-line</i>	задає стиль першого рядка текстового блоку
<i>after</i>	додає вміст після елемента
<i>before</i>	додає вміст до елемента

Приклад:

```
p {font: 24px Arial;}
/*задає стилі для першої літери абзацу*/
p:first-letter {
    font: bold 32px Verdana;
    color:#6495ED;
    text-align: justify;
}
/*задає стилі для першого рядка абзацу*/
p:first-line {
    font: 32px "Book Antiqua";
    color: #5F9EA0;
}
```

Псевдоелементи *after* і *before* дозволяють додавати (вставляти) контент перед (*:before*) або після (*:after*) елемента за допомогою властивості content.

Синтаксис:

селектор:after {content: значення;}
селектор:before {content: значення;}

Правила стилю поширюються на всі елементи:
*:псевдоелемент {властивість: значення; }

Табл. 3.6

Значення властивості *content*

Значення	Пояснення
"текст"	текст
url(шлях до об'єкта)	адреса об'єкта
open-quote	відкриваючі лапки
close-quote	закриваючі лапки
no-open-quote	скасовує відкриваючі лапки
no-close-quote	скасовує закриваючі лапки

Приклад:

```
li :before {content: url("pic.gif"); }  
/*перед кожним елементом списку додає картинку*/  
p :after {content: "Кінець абзацу"; }  
/*після кожного абзацу додає зазначений текст*/
```

3.8 Ідентифікатори та класи

Атрибут *id* застосовується для ідентифікації унікального елемента. У документі не може бути більше ніж один елемент з одним і тим самим *id*.

Синтаксис

#ім'я_ідентифікатора {властивість: значення;}

Приклад:

```
#id1 {color: #FFDAB9;}
```

Визначення ідентифікатора для тегу

Тег#ім'я_ідентифікатора {властивість: значення;}

Приклад:

```
p {background: #E6E6FA;}
p#id2 {
    background: #DCDCDC;
}
```

Звернення до ідентифікатора

<Тег id = "ім'я_ідентифікатора">

Приклад:

```
<p id = "id2"> Колір фона абзацу #DCDCDC
<p> Колір фона абзацу #E6E6FA
```

Атрибут **class** дозволяє визначити властивість для групи елементів.

Синтаксис

.ім'я_класу {властивість: значення;}

Приклад:

```
.c11 {color: #0000FF;}
```

Визначення класу для тегу

Тег.ім'я_класу { властивість: значення;}

Приклад:

```
p {text-align: justify;}
/*задає стиль для абзацу, до якого підключено клас c12*/
p.c12 {
    color: #6B8E23;
    font-style: italic;
}
```

Звернення до класу

< Тег class = "ім'я_класу">

Приклад:

```
<p class = "c12"> До абзацу підключено клас c12
<p> До абзацу не підключено жодного класу
```

Застосування декількох класів одночасно

```
.p_font {font-size: 14pt;}
.p_color {color: #6B8E23;}
<p class = "p_font p_color"> Текст абзацу </p>
```


3.9 Розмір елементів

Для визначення розміру блокових елементів застосовуються властивості *width* та *height*.

1. Висота елемента

Синтаксис:

селектор {**height** : значення; }

Додаткові властивості:

min-height – мінімальна висота елемента.

max-height – максимальна висота елемента.

2. Ширина елемента

Синтаксис

селектор {**width** : значення; }

Додаткові властивості:

max-width – максимальна ширина елемента.

min-width – мінімальна ширина елемента.

Ці властивості CSS можуть задаватися в:

px

%

none – без обмежень.

Приклад:

```
.b11 {
  background: #66CDAA;
  width:50px;
  height:50px;}
.b12 {
  background: #66CDAA;
  width: 50%;           /*50 % ширини батьківського елемента*/
  height:50px;}

<div class="b11"></div>
<div class="b12"></div>
```

3.10 Боксова модель

Боксова модель передбачає, що кожен елемент розташовується в окремому боксі, таким чином, для кожного елемента можна задавати його позицію, розміри, встановлювати внутрішні та зовнішні відступи, визначати тип боксу (рядковий, блоковий), або взагалі не відображати (приховувати) елемент і т. ін.

У загальному вигляді, концепція боксової моделі наведена на рисунку 3.2.

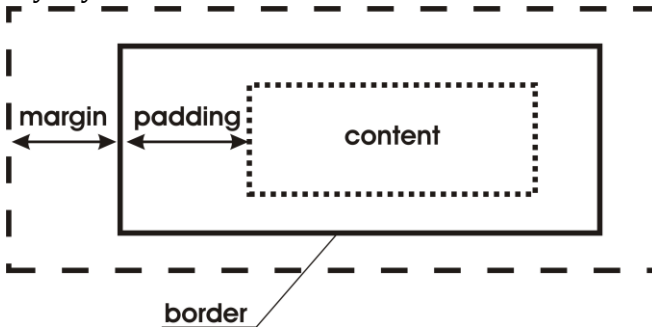


Рис. 3.2. Концепція боксової моделі

Властивість **margin** – задає відстань від зовнішньої границі (рамки) елемента до границі елемента-батька або іншого елемента, що знаходиться на одному з ним рівні вкладеності.

Синтаксис margin:

```
селектор {  
    margin-top: значення;  
    /* Відступ від верхньої границі елемента */  
    margin-right: значення;  
    /* Відступ від правої границі елемента */  
    margin-bottom: значення;  
    /* Відступ від нижньої границі елемента */  
    margin-left: значення;  
    /* Відступ від лівої границі елемента */  
}
```

Скорочений запис:

margin: margin-top | margin-right | margin-bottom | margin-left

Приклад:

```
h1 {  
    margin-top: 100px;  
    margin-right: 40px;  
    margin-bottom: 10px;  
    margin-left: 70px;  
}
```

АБО

```
h1 {margin: 100px 40px 10px 70px;}
```

Способи запису значень властивості **margin**

```
h1 {margin: 100px 40px 10px;}
```

У прикладі задано верхній відступ=100px, правий відступ = 40px, нижній відступ = 10px та лівий відступ = 40px.

```
h1 {margin: 100px 40px;}
```

У прикладі задано верхній та нижній відступи=100px, правий та лівий відступи = 40px.

```
h1 {margin: 100px;}
```

У прикладі задано всі відступи = 100px.

Властивість **padding** – задає відстань (відступ) від внутрішньої границі елемента (рамки) до його контенту.

Синтаксис padding:

селектор {

padding-top: значення;

/ Відступ від верхньої границі елемента до контенту*/*

padding-right: значення;

/ Відступ від правої границі елемента до контенту*/*

padding-bottom: значення;

/ Відступ від нижньої границі елемента до контенту*/*

padding-left: значення;

/ Відступ від лівої границі елемента до контенту*/*

}

Приклад:

```
h2 {  
    padding-top: 10px;  
    padding-right: 20px;  
    padding-bottom: 30px;  
    padding-left: 80px;  
}
```

АБО

```
h2 {padding: 10px 20px 30px 80px;}
```

Відступи **margin** та **padding** задаються в:
px – у пікселях або інших допустимих CSS одиницях вимірювання;

% – у відсотках;

auto – розмір полів та відступів визначається браузером автоматично.

Для визначення границь елементів (рамки) застосовується властивість **border**

Табл. 3.7

Властивості рамки

Властивість	Пояснення
<i>border-width</i>	визначає товщину рамки (<i>thin, medium</i> та <i>thick</i> , або числове значення)
<i>border-color</i>	визначає колір рамки
<i>border-style</i>	визначає тип рамки (<i>dotted, dashed, solid, double</i> і т. ін.)

Приклад:

```
o1 {  
    border-width: thick;  
    border-style: double;  
    border-color: lime;  
}
```

АБО

```
o1 {border: thick double lime;}
```

Закруглені кути елементів

Скруглення кутів на основі
кола

Синтаксис border-radius:

border-radius: радіус;

Приклад:

```
border-radius: 10px;
```

Способи визначення значень властивості border-radius:

```
div {border-radius: 10px;}
```

Задано радіус скруглення для всіх кутів = 10px;

```
div {border-radius: 20px 10px;}
```

Задано радіус скруглення для лівого верхнього та правого нижнього кутів = 20px, і правого верхнього та лівого нижнього кутів = 10px;

```
div {border-radius: 50px 0 10px;}
```

Задано радіус скруглення для лівого верхнього = 50px, правого верхнього і лівого нижнього кутів = 0, для правого нижнього = 10px.

```
div {border-radius: 50px 20px 10px 0;}
```

Задано радіус скруглення для лівого верхнього = 50px, правого верхнього = 20px, лівого нижнього = 10px, для правого нижнього кута = 0px.

Тип боксу задається властивістю **display**. Ця властивість дозволяє перетворювати блокові елементи в рядкові, і навпаки.

Синтаксис display:

селектор {display: значення;}

Скруглення кутів на основі
еліпса

Синтаксис border-radius:

border-radius: R1 / R2;

R1 – горизонтальний радіус;

R2 – вертикальний радіус.

```
border-radius: 10px/6px;
```

Табл. 3.8

Значення властивості *display*

Значення	Пояснення
block	відображає елемент як блоковий
inline	відображає елемент як рядковий
none	елемент не відображається ("видаляється" із загального потоку)

Приклад:

```
.block_style{display: inline;}
.cl1 {background: #66CDAA;}
.cl2 {background: #BDB76B;}
.cl3 {background: #A0522D;}

<div class="block_style cl1">Блок 1</div>
<div class="block_style cl2">Блок 2</div>
<div class="block_style cl3">Блок 3</div>
```

Властивість *overflow* управляє відображенням вмісту блокового елемента.

Синтаксис overflow:

селектор {overflow : значення;}

Табл. 3.9

Значення властивості *overflow*

Значення	Пояснення
<i>visible</i>	елемент розтягується до необхідних розмірів
<i>hidden</i>	вміст елемента "обрізається" до розмірів елемента
<i>scroll</i>	додаються полоси прокрутки за замовчуванням
<i>auto</i>	полоси прокрутки додаються за необхідністю

Приклад:

```
.over{
    overflow: auto;
    width: 20px;
    height: 50px;
}
<div class="over"> Текст </div>
```

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Властивість **visibility** призначена для керування видимістю елемента.

Синтаксис visibility:

селектор {visibility : значення;}

Табл. 3.10

Значення властивості **visibility**

Значення	Пояснення
<i>inherit</i>	елемент відображається за правилами, що вказані в батьківському елементі
<i>visible</i>	візуалізує елемент
<i>hidden</i>	ховає елемент
<i>collapse</i>	ховає стовпчики та рядки таблиці (як display: none); для решти елементів працює як hidden

Приклад

```
.block_v{visibility: hidden;}
.c11 {background: #66CDAA;}
.c12 {background: #BDB76B;}
.c13 {background: #A0522D;}

/*Блок 1 не візуалізується*/
<div class="block_v c11">Блок 1</div>
<div class="c12">Блок 2</div>
<div class="c13">Блок 3</div>
```

3.11 Плаваючі елементи

Властивість **float** задає обтікання елемента. Бокс, до якого застосована властивість **float**, включаючи його контент, спливає праворуч або ліворуч у вікні документа, або в межах батьківського елемента.

Синтаксис float:

селектор {float : значення;}

Табл. 3.11

Значення властивості *float*

Значення	Пояснення
<i>left</i>	бокс та його контент може спливати ліворуч (обтікатися з правої сторони)
<i>right</i>	бокс та його контент може спливати праворуч (притискатися до правої границі, обтікатися з лівого боку)
<i>none</i>	бокс та його контент не обтікається

Приклад

```
.r{
    width:50%;
    height:50px;
}
.f1{
    float:right;
    background: #66CDAA;
}
.f2{
    float:left;
    background: #778899;
}
<div class="f1 r"></div>
<div class="f2 r"></div>
```

Властивість ***clear*** вказує, з якого боку заборонено обтікання елемента.

Синтаксис clear:

селектор {clear : значення;}

Табл. 3.12

Значення властивості *clear*

Значення	Пояснення
<i>left</i>	обтікання заборонено з лівої сторони
<i>right</i>	обтікання заборонено з правої сторони
<i>both</i>	обтікання заборонено з обох боків
<i>none</i>	обтікання дозволено з обох боків. Значення за замовчуванням

Приклад

```
<div style="clear: both;"> </div>
```

3.12 Позиціонування елементів

Для вказання позиції (місця розташування) елемента застосовується властивість **position**:

Синтаксис position:

```
селектор {  
    position: значення;  
    bottom: значення;  
    left: значення;  
    right: значення;  
    top: значення;  
}
```

Табл. 3.13

Значення властивості **position**

Значення	Пояснення
<i>absolute</i>	елемент виривається із загального потоку та втрачає своє місце (простір) у документі, його координати обраховуються відносно границь вікна документа або батьківського елемента
<i>relative</i>	позиція елемента, що обраховується відносно його оригінальної позиції – елемент зміщується праворуч, ліворуч, угору або вниз відносно своєї оригінальної позиції
<i>fixed</i>	працює так само як <i>absolute</i> , але при прокручуванні елемент залишається на своєму місці
<i>static</i>	елемент відображається як зазвичай
<i>inherit</i>	наслідує значення від батьківського елемента

Для вказання позиції елемента на екрані застосовуються властивості:

bottom – відстань від нижньої границі вікна браузера.

left – відстань від лівої границі вікна браузера.

right – відстань від правої границі вікна браузера.

top – відстань від верхньої границі вікна браузера.

1. Абсолютне позиціонування

Оскільки елемент виривається із загального потоку, він звільняє простір, який займав до цього. Його місце займають інші елементи.

Синтаксис:

селектор {position : absolute;}

Для позиціонування елемента відносно границь вікна браузера застосовують властивості: *bottom*, *left*, *right*, *top*.

Приклад:

```
#pos1 {
  position: absolute;
  background: #1E90FF;
  top: 100px;
  left: 100px;
  width: 150px;
  height: 150px;
}
<div id="pos1"></div>
```

2. Відносне позиціонування

Елемент із відносним позиціонуванням змінює своє розташування, зміщуючись праворуч, ліворуч, вгору або вниз на вказане значення відносно своєї оригінальної позиції.

Синтаксис:

селектор {position : relative;}

Приклад:

```
#pos2 {
  position: relative;
  background: #1E90FF;
  top: 100px;
  left: 100px;
  width: 150px;
  height: 150px;
}
<div id="pos2"></div>
```

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Для позиціонування вкладених блоків, у блока батька задають **position: relative**, а у вкладеного **position: absolute**. Таким чином, система координат для вкладеного блока визначається батьківським (відступи **bottom, left, right, top** обраховуються відносно границь елемента батька).

3. Фіксоване позиціонування

Елемент із фіксованим позиціонуванням виривається із загального потоку та втрачає своє місце в документі, його координати обраховуються відносно границь вікна документа. При прокручуванні елемент не змінює свого місцезрештування відносно границь вікна документа.

Синтаксис:

селектор {**position : fixed;**}

Приклад:

```
#pos3 {
  position: fixed;
  background: #1E90FF;
  top: 50%;
  left: 50%;
  margin: -75px 0 0 -75px; /*розташовуємо по центру*/
  width: 150px;
  height: 150px;
}

<div id="pos2"></div>
```

3.13 Керування шаром

Властивість **z-index** керує порядком накладання позиційованих елементів.

У елемента, розташованого на задньому плані **z-index** менший, ніж у елемента, що розташований на передньому плані.

Синтаксис:

селектор {**z-index: значення;**}

Значення властивості z-index

Значення	Пояснення
auto	елементи накладаються один на одного в тому порядку, в якому вони вказані в кодї HTML (за замовчуванням)
ціле число	чим більше число, тим більш високу позицію займає елемент

Приклад:

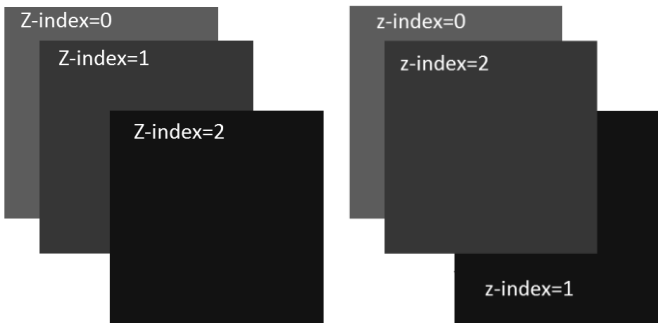


Рис. 3.3. Приклад застосування властивості z-index

Варіант 1	Варіант 2
<pre> .b11 { background: #2E8B57; top:0; left:0; } .b12 { background: #B22222; top:20px; left:20px; } .b13 { background: #1E90FF; top:40px; left:40px; } .box{ position: absolute; width:200px; </pre>	<pre> .b11 { background: #2E8B57; top:0; left:0; } .b12 { background: #B22222; top:20px; left:20px; z-index:2; } .b13 { background: #1E90FF; top:40px; left:40px; z-index:1; } .box{ </pre>

Практичний web-дизайн: проектування, створення та супроводження web-вузла

<pre>height:200px; }</pre>	<pre>position: absolute; width:200px; height:200px; }</pre>
<pre><div class="box b11">Z-index 0</div> <div class="box b12">Z-index 1</div> <div class="box b13">Z-index 2</div></pre>	

За замовчуванням, контент розташований на початку документа має **z-index** менший, ніж контент, що розташований у кінці документа.

3.14 @-правила CSS

Табл. 3.15

@-правила CSS

@-правила	Пояснення
<i>@charset</i>	застосовується для вказання кодування зовнішнього CSS-файлу
<i>@font-face</i>	визначає налаштування шрифтів
<i>@import</i>	імпортує вміст CSS-файлу в поточну таблицю стилів
<i>@media</i>	указує тип носія, для якого буде застосовано вказаний стиль
<i>@page</i>	задає значення полів документа, що друкується

Media-запит (@media)

Синтаксис:

```
@media not | only mediatype and (media function)
{ код CSS }
```

Табл. 3.16

Типи пристроїв

Тип	Пояснення
all	застосовується для всіх типів пристроїв
handheld	застосовується для невеликих або портативних пристроїв
print	застосовується для принтерів
screen	застосовується для екранів комп'ютерів, планшетів, смартфонів і т. ін.
tv	застосовується для телеекранів

Табл. 3.17

Мультимедійні функції

Функція	Пояснення
aspect-ratio	задає співвідношення ширини та висоти області перегляду
color	визначає кількість біт на колір для пристрою
color-index	визначає кількість кольорів, яку пристрій може відобразити
device-aspect-ratio	визначає співвідношення ширини та висоти пристрою виведення
device-height	задає висоту пристрою
device-width	задає ширину пристрою
height	задає висоту області перегляду
orientation	задає режим перегляду дисплея: ландшафтний або портретний
resolution	задає роздільну здатність
width	задає ширину області перегляду

Табл. 3.18

Логічні оператори

Оператор	Пояснення
and	логічне і, об'єднує декілька медіа-запитів в один (запит виконується, якщо всі медіа-запити, що складають умову, виконуються)
not	логічне заперечення, інвертує значення результату
,	логічне або, об'єднує декілька медіа-запитів в один (запит виконується, якщо хоча б один медіа-запит, що складає умову, виконується)

Приклад:

```
p {color: #FFDAB9;}
@media (max-width: 1200px) and (min-width: 900px) {
  p {color: green;}
}
```

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Колір тексту стає зеленим, якщо ширина екрана буде знаходитись у межах між 900px та 1200px (включно).

Правило @font-face

Синтаксис:

```
@font-face{
  font-family: name;           /* Обов'язкове значення */
  src: URL;                   /* Обов'язкове значення */
  font-stretch: normal | condensed | ultra-condensed | extra-
condensed | semi-condensed | expanded | semi-expanded | extra-
expanded | ultra-expanded;
  font-style: normal | italic | oblique;
  font-weight: normal | bold | 100 | 200 | 300 | 400 | 500 | 600 |
700 | 800 | 900;
  unicode-range: unicode-range; }
```

Табл. 3.19

Властивості правила @font-face

Властивість	Пояснення
<i>font-family</i>	задає ім'я шрифту
<i>src</i>	задає URL-адресу для завантаження шрифту
<i>font-stretch</i>	задає, як шрифт має бути розтягнутий
<i>font-style</i>	стиль шрифту
<i>font-weight</i>	встановлює, наскільки жирним буде текст
<i>unicode-range</i>	задає діапазон символів Unicode, які підтримує шрифт

Приклад:

```
@font-face {
  font-family: "myFont";           /* Назва */
  src: url("beer_money.ttf");     /* Шлях до файлу */
}
P{
  font-family : "myFont", arial;
  color: #E6E6FA;
}
```

РОЗДІЛ 4

ОСНОВИ JavaScript

4.1 Основні поняття мови JavaScript

4.1.1 Підключення JavaScript

Підключити скрипти JavaScript можна двома шляхами:

1. Тег `<script>`

```
<script>  
    КОД  
</script>
```

2. Підключення зовнішнього файлу *.js

```
<script src="js/script1.js"></script>
```

Підключення декількох скриптів:

```
<script src="js/script1.js"></script>  
<script src="js/script2.js"></script>
```

4.1.2 Змінні. Правила побудови ідентифікаторів

Змінна складається з імені та виділеної області пам'яті, що їй відповідає.

Оголошення змінної. Синтаксис:

`var ім'я_змінної;`

Приклад:

```
var msg;  
msg = 'Привіт';
```

АБО

```
var msg = 'Привіт';
```

Правила побудови ідентифікаторів:

1. Ідентифікатор (ім'я змінної, функції, об'єкта і т. ін.) може складатися із: цифр, літер, символів \$ та _
2. Першим символом не може бути цифра.

Приклад:

Правильний ідентифікатор		Помилка
<code>var myName;</code>	<code>var \$ = 5;</code>	<code>var my-name;</code>
<code>var test123;</code>	<code>var _ = 15;</code>	<code>var 1a;</code>
<code>var Name = "Вася";</code>		

4.1.3 Коментарі

```
// Приклад однорядкового коментаря
/* Приклад
багаторядкового
коментаря
*/
```

4.1.4 Константи

Константа – це змінна, значення якої не змінюється (є постійною).

Приклад:

```
var CL_RED = "#F00";
var CL_GREEN = "#0F0";
```

4.1.5 Типи даних

1. Примітивні типи даних

1.1. Числа

```
var n = 123;
n = 12.345;
```

Змінні числового типу можуть приймати значення: ціле число, дійсне число, *Infinity* (нескінченність) та *NaN* (помилка обчислень).

Нескінченність *Infinity* отримується при діленні на нуль:

```
alert( 10 / 0 ); // Infinity
```

Помилка обчислень *NaN* отримується при виконанні некоректної математичної операції:

```
alert( "2" * 2 ); // NaN, помилка
```

1.2. Рядок

Рядок розташовується в одинарних або подвійних лапках.

```
var str = "рядок 1";  
str = 'рядок 2';
```

1.3. Логічний тип

Змінні логічного типу можуть приймати значення - *true* (істина) або *false* (хибність).

```
var checked = true;  
checked = false;
```

1.4. Значення *null* – значення невідоме

```
var a = null;
```

1.5. Значення *undefined* – значення не задано

Якщо змінна оголошена та їй не присвоєно жодного значення, то вона має значення *undefined*:

```
var a;  
alert(a); // undefined
```

2. Об'єкти

До цього типу даних відносяться, наприклад, об'єкти *Date*, *Math*, *Array*, які використовуються для колекцій даних і т. ін.

4.2 Оператори

4.2.1 Оператор присвоєння

Синтаксис:

ім'я_змінної = вираз;

Оператор обраховує значення виразу, який розташовується праворуч від знака =, і присвоює результат змінній, що розташована ліворуч від знака =. Праворуч від знака = може бути інша змінна, константа, вираз, виклик функції, що повертає значення, об'єкт і т. ін.

Приклад:

```
var i = 10 + 2;  
alert(i); // 12  
var a = 7;  
var b = 3;  
a = b + a;  
alert(a); // 10  
var a, b, c;  
a = b = c = 5 + 2;  
alert(a); // 7  
alert(b); // 7  
alert(c); // 7
```

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Скорочена форма оператора присвоєння:

$a = a + b$	$a += b$
$a = a - b$	$a -= b$
$a = a * b$	$a *= b$

4.2.2 Арифметичні оператори

Табл. 4.1

Унарні оператори

Оператор	Назва	Пояснення
-	унарний мінус	змінює знак числа на протилежний
+	унарний плюс	приводить операнд до числового типу
++	інкремент	збільшення операнда на 1
--	декремент	зменшення операнда на 1

Приклад: оператор унарний мінус «-»

```
var a = 10;
alert(-a);           // -10
alert(-(-10));      // 10
```

Приклад додавання двох чисел, у формі рядків

```
var a = "5";
var b = "5";
alert(a + b);       // "55", рядок
alert(+a + b);     // "55", рядок
alert(+a + +b);    // 10, число
```

Інкремент збільшує значення операнда на 1:

```
var i = 1;
i++;
alert(i);           // 2
5++;               // помилка
```

Декремент зменшує значення операнда на 1:

```
var i = 1;
i--;
alert(i);           // 0
5--;               // помилка
```

Оператор є бінарним, якщо він застосовується до двох операндів. Операнд – це те, над чим виконується дія (операція).

Табл. 4.2

Арифметичні бінарні оператори

Оператор	Пояснення
+	додавання
-	віднімання
*	множення
/	ділення
%	остача від ділення

Приклад:

```
alert(15 % 3); // 0, без остачі  
alert(17 % 3); // 2
```

Додавання рядків

Над рядками можна виконувати тільки одну арифметичну операцію – операцію додавання (конкатенація рядків).

Приклад:

```
var a = "Це" + " все" + " один" + " рядок";
```

Якщо хоча б один із аргументів – рядок, то другий аргумент перетворюється в рядок (тільки для бінарної операції додавання).

Приклад:

```
alert('10'+ 2 ); // "102"
```

Інші арифметичні оператори завжди приводять аргументи до числа.

Приклад:

```
alert('10' - 2); // 8
```

4.2.3 Оператори порівняння

Табл. 4.3

Оператори порівняння

Оператор	Пояснення
>	більше
<	менше
>=	більше або дорівнює
<=	менше або дорівнює
==	дорівнює
!=	не дорівнює

Приклад:

```
var a = 5, b = 10;  
alert(a == b);           // false, неправильно  
alert((a + 5) == b);    // true, правильно  
alert(a != b);          // true, правильно  
alert(b == 10);         // true, правильно
```

Порівняння рядків

JavaScript використовує кодування Unicode. При порівнянні рядків – посимвольно порівнюються числові коди відповідних символів.

Приклад:

```
alert('Я' > 'А');       // true  
alert("7" > "10");      // true  
alert(+ "10" < + "11"); // true, порівнюються числа
```

Порівняння різнотипних даних

При порівнянні значення числового типу із значенням іншого типу, відбудеться спроба перетворення нечислового значення в числове, і порівняння відбудеться над числами.

Приклад:

```
alert('10' > 1);        // true
```

4.2.4 Логічні оператори

Табл. 4.4

Оператор	Пояснення
&&	логічне І
	логічне АБО
!	логічне заперечення

Табл. 4.5

A	B	A&&B	A B	!A
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Приклад: А більше або дорівнює В, або С не дорівнює 0

```
((A >= B) || (C != 0))
```

4.3 Керуючі структури JavaScript

4.3.1 Оператор умови

Оператор **if** застосовується, якщо існує вибір між двома напрямками виконання програми.

Синтаксис:

if (умова) оператор;
 або

if (умова) { /*Блок операторів */};

Умова – це вираз логічного типу.

Алгоритм роботи:

1. Обрахувати значення виразу в умові.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

2. Якщо результат виразу = *true*, то виконати оператор, або блок операторів, що розташований після умови; інакше нічого не виконувати.

Приклад:

```
var a = +prompt('Введіть число a', '');  
var d = +prompt('Введіть число d', '');  
if (a == d) alert('Числа рівні');
```

Повна форма оператора умови

Синтаксис:

if (*умова*) оператор1; **else** оператор2;

Якщо *умова* = *true*, то виконати оператор1 (або блок операторів, що розташований після умови); інакше виконати оператор2 (або блок операторів, що розташований після ключового слова **else**).

Приклад:

```
var a = +prompt('Введіть число a', '');  
var d = +prompt('Введіть число d', '');  
if (a == d)  
    alert('Числа рівні');  
    else alert('Числа не рівні');
```

Приклад:

$$Y = \begin{cases} x^2, & \text{якщо } x > 4; \\ \frac{1}{x}, & \text{інакше.} \end{cases}$$

```
var x = prompt('Введіть x', '');  
x = +x;  
var rez = 'Помилка';  
if (x > 4)  
{  
    y = x*x;  
    rez = 'Y='+y;  
}  
else if (x != 0)  
{  
    y = 1/x;  
    rez = 'Y='+y;  
}  
alert(rez);
```

4.3.2 Оператор ?

Тернарний оператор. Синтаксис:

результат = (умова) ? значення1 : значення2;

Перевіряється умова, якщо *умова=true*, то в змінну *результат* повертається значення *значення1*, якщо *умова=false*, то в змінну *результат* повертається значення *значення2*.

Приклад:

```
var a = +prompt('Введіть число a', '');  
var d = +prompt('Введіть число d', '');  
var max = (a > d) ? a : d;  
alert(max);
```

4.3.3 Оператор множинного вибору

Оператор *switch* застосовується, якщо існує вибір між декількома (більше двох) напрямками виконання програми.

Синтаксис:

switch (вираз)

```
{  
  case константа_1: оператор_1; break;  
  case константа_2: оператор_2; break;  
  ...  
  case константа_n: оператор_n; break;  
  default: оператор;  
}
```

Алгоритм роботи оператора *switch*:

1. Обраховується значення виразу.
2. Результат по чергово порівнюється з кожною константою (замість константи може бути логічний вираз).
3. Якщо результат обрахунку виразу дорівнює константі, то виконується оператор або блок операторів, вказаний після відповідної константи до першого зустрічного оператора *break*;

Практичний web-дизайн: проектування, створення та супроводження web-вузла

4. Якщо результат не дорівнює жодній з констант, то виконується оператор, розташований після ключового слова *default*.

Приклад. Користувач задає арифметичну операцію – обрана арифметична операція виконується над числовими значеннями

```
var x = prompt('Введіть x', ''); // арифметична операція
var a = 4, b = 8, m;
switch(x)
{
    case "+": m = a + b; break;
    case "-": m = a - b; break;
    case "*": m = a * b; break;
    default: m = "Операція не задана"; break;
}
alert(m);
```

Приклад. Групування case

Користувач задає номер місяця – виводиться назва сезону, до якого належить місяць

```
var x = prompt('Введіть x', ''); // номер місяця
var m;
switch(x)
{
    case 1: case 2: case 12: m = "зима"; break;
    case 3: case 4: case 5: m = "весна"; break;
    case 6: case 7: case 8: m = "літо"; break;
    case 9: case 10: case 11: m = "осінь"; break;
    default : m = "Не існує"; break;
}
alert(m);
```

4.3.4 Оператори циклу

Цикл – це спеціальна конструкція мови, яка дозволяє запрограмувати багаторазове виконання команди або групи команд. Кожен прохід циклу називається ітерацією.

Цикл **while**. Цикл з передумовою

Синтаксис:

```
while (умова)
{
  /*Тіло циклу - блок операторів */
}
```

Алгоритм роботи циклу:

1. Обрахувати значення виразу в умові.
2. Якщо умова = *true*, виконати тіло циклу (блок операторів).
3. Якщо умова = *false* – завершити роботу циклу.

Приклад:

```
var i = 0;
while (i < 3)
{
  alert(i);
  i++;
}
```

Результат:

0
1
2

Цикл **do..while**. Цикл з післяумовою

Синтаксис:

```
do
{
  /*Тіло циклу - блок операторів */
} while (умова)
```

Алгоритм роботи циклу:

1. Обрахувати значення виразу в умові.
2. Якщо умова = *true*, виконати тіло циклу (блок операторів).
3. Якщо умова = *false* – завершити роботу циклу.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

На відміну від циклу *while*, цикл *do..while* виконується як мінімум 1 раз.

Приклад:

```
var i = 0;
do {
  alert(i);
  i++;
} while (i < 3)
```

Цикл for. Цикл з параметром

Синтаксис:

```
for (ініціалізація; умова; крок)
{
  /*Тіло циклу - блок операторів */
}
```

ініціалізація – початкове значення лічильника;

умова – умова виконання циклу;

крок – спосіб змінення лічильника.

Приклад:

```
var i;
for (i=0; i<3; i++) { alert(i); }
```

4.4 Функції

4.4.1 Оголошення функції

Синтаксис:

```
function ім'я_функції (список параметрів)
{
  /*Тіло функції – блок операторів*/
}
```

У дужках після імені функції перераховуються через кому формальні параметри функції – її вхідні аргументи.

Тіло функції являє собою блок операторів, які виконуються при виклику функції.

Приклад:

```
function hello() {  
  alert('Привіт!');  
}
```

Виклик функції

```
hello();
```

Приклад:

```
function sum(a, b) {  
  alert(a + b);  
}
```

Виклик функції

```
sum(5, 6); //11
```

АБО

```
var c=10, d = 7;  
sum(c, d);
```

При виклику функції вказуються фактичні параметри, тобто змінні, що мають конкретні значення при виклику функції (a, b – формальні параметри; c, d – фактичні параметри). Імена фактичних і формальних параметрів можуть збігатися.

4.4.2 Повернення значення

Функція може повертати результат, для цього застосовується оператор *return*.

Приклад:

```
function sum(a, b)  
{  
  return a + b;  
}  
var c=4, d = 7;  
var rez = sum(c, d);  
alert(rez);
```

4.4.3 Локальні і глобальні змінні

Локальні змінні

Функція може містити локальні змінні. Ці змінні оголошуються через службове слово *var* у тілі функції. Такі змінні є доступними (видимими) тільки всередині функції, у якій вони оголошені:

Приклад:

```
function sum()
{
  var a = +prompt('Введіть число a', '');
  var b = +prompt('Введіть число b', '');
  /* a і b локальні змінні. Їх область видимості обмежується функцією sum */
  return a + b;
}
```

Виклик:

```
sum();
alert(a);      // помилка (змінна a = undefined)
```

Зовнішні (глобальні) змінні

Глобальні (зовнішні) змінні – це змінні, що оголошені на рівні всього скрипту.

Приклад:

```
var a, b; /* a і b глобальні змінні. Область їх видимості розповсюджується на весь скрипт */

function sum()
{
  a = +prompt('Введіть число a', '');
  b = +prompt('Введіть число b', '');
  return a + b;
}
```

Виклик:

```
sum();
alert(a);      // буде виведено значення змінної a
```

4.4.4 Стандартні функції. Вікна повідомлень

1. Функція **alert**

Виводить модальне вікно (зупиняє виконання скрипту) з повідомленням, доки користувач не закриє вікно (не натисне кнопку ОК).

Синтаксис:

`alert('текст повідомлення');`

Приклад:

```
alert("Привіт");
```

2. Функція **prompt**

Виводить модальне вікно із заголовком *title*, полем для введення тексту, заповненим рядком за замовчуванням *default* і кнопками ОК/CANCEL.

Рядок, введений користувачем, зберігається в параметр *result*. Користувач може відмовитись від введення рядка, натиснув кнопку CANCEL або клавішу ESC, у цьому випадку параметр *result* прийме значення *null*.

Синтаксис:

`result = prompt(title, default);`

Приклад:

```
var str = prompt('Введіть ваше прізвище', '');
```

3. Функція **confirm**

Виводить модальне вікно з повідомленням і двома кнопками: ОК і CANCEL. Якщо користувач натискає кнопку ОК – функція повертає значення *true*, при натисканні кнопки CANCEL – функція повертає значення *false*.

Синтаксис:

`result = confirm('текст повідомлення');`

Приклад:

```
var rez = confirm("2 + 2 = 4 ?");  
alert(rez);
```

4.5 Об'єкти

4.5.1 Об'єкт Date

Об'єкт *Date* забезпечує доступ до різних функцій і методів для роботи із датою та часом.

Табл. 4.6

Методи об'єкта *Date*

Метод	Пояснення
<i>getDate()</i>	повертає день місяця в діапазоні від 1 до 31
<i>getDay()</i>	повертає день тижня: 0 – нд, 1 – пн, 2 – вт, 3 – ср, 4 – чт, 5 – пт, 6 – сб.
<i>getHours()</i>	повертає час в діапазоні від 0 до 23
<i>getMinutes()</i>	повертає значення хвилин у діапазоні від 0 до 59
<i>getMonth()</i>	повертає значення місяця в діапазоні від 0 до 11
<i>getSeconds()</i>	повертає значення секунд у діапазоні від 0 до 59
<i>getFullYear()</i>	повертає значення року
<i>setDate(day)</i>	задає день місяця в об'єкті від 1 до 31
<i>setHours(hours)</i>	задає години в об'єкті від 0 до 23
<i>setMinutes(minutes)</i>	задає хвилини в об'єкті від 0 до 59
<i>setMonth(month)</i>	задає місяць в об'єкті від 1 до 12
<i>setSeconds(seconds)</i>	задає секунди в об'єкті від 0 до 59
<i>setYear(year)</i>	задає рік в об'єкті, year має бути більше за 1900
<i>toString()</i>	перетворює вміст об'єкта <i>Date</i> в рядок

Приклад. Вивести поточний час

```
function H() {  
  var d = new Date();  
  return d.getHours();  
}
```

```
function M() {
    var d = new Date();
    return d.getMinutes();
}
function S() {
    var d = new Date();
    return d.getSeconds();
}
var time_ = H() + ":" + M() + ":" + S();
alert(time_);
```

4.5.2 Об'єкт Math

Об'єкт *Math* забезпечує доступ до різноманітних математичних функцій.

Табл. 4.7

Методи об'єкта *Math*

Метод	Пояснення
<i>abs(x)</i>	повертає абсолютне значення аргумента
<i>acos(x)</i>	повертає арккосинус аргументу
<i>asin(x)</i>	повертає арксинус аргументу
<i>atan(x)</i>	повертає арктангенс аргументу
<i>sin(x)</i>	повертає синус аргументу
<i>cos(x)</i>	повертає косинус аргументу
<i>tan(x)</i>	повертає тангенс аргументу
<i>exp(x)</i>	повертає експоненту аргументу
<i>sqrt(x)</i>	повертає квадратний корінь із аргумента
<i>ceil(x)</i>	повертає найменше ціле число, більше або рівне значенню аргумента
<i>floor(x)</i>	повертає найбільше ціле число, менше або рівне значенню аргумента
<i>round(x)</i>	округляє аргумент до найближчого цілого
<i>log(x)</i>	повертає натуральний логарифм аргументу
<i>max(x,y)</i>	повертає найбільший із аргументів
<i>min(x,y)</i>	повертає найменший із аргументів
<i>pow(x,y)</i>	підносить x у степінь y
<i>random()</i>	генерує випадкове число в діапазоні від 0 до 1

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Приклад:

```
div = Math.floor(15/2); //7,  
div = (15/2); //7,5,
```

Метод **abs**

```
var x = Math.abs(5); //5  
var x = Math.abs(-5); //5
```

Метод **ceil**

```
var x = Math.ceil(-2.8); //-2  
var x = Math.ceil(2.8); //3
```

Метод **pow**

```
var x;  
x = Math.pow(3, 2); //9
```

Метод **round**

```
var x;  
with (Math) {  
  x = round(2.49); //2  
  x = round(2.5); //3  
}
```

Метод **sin**

```
var x;  
with (Math) {  
  x = sin(0); //0  
  x = sin(PI/2); //1  
}
```

Метод **sqrt**

```
var x = Math.sqrt(4); //2  
var x = Math.sqrt(-1); //NaN
```

4.5.3 Об'єкт **Array**

Масив – це сукупність елементів, що мають спільне ім'я і розташовані в послідовних (сусідніх) комірках пам'яті.

Індекс – номер елемента масиву. Індексція елементів масиву починається з 0.

Масив може містити різнотипні елементи.

Приклад:

```
var a = ["рядок", 25, true];  
var b = [17, 55];
```

Кількість елементів масиву (довжина масиву)
ім'я_масиву.*length*

Приклад:

```
alert(a.length) // довжина масиву = 3
```

Звернення до елемента масиву

ім'я_масиву[індекс];

Приклад:

```
alert(a[0]); // "рядок"  
alert(b[0]); // 17  
a[0] = 10;  
alert(a[0]); // 10
```

Методи *push* і *pop*

Додають або видаляють значення з кінця масиву

Приклад:

```
var A = [3, 5, 7];  
A.push(9);  
alert(A.pop()); // 9  
var last = A.pop(); // = 7  
alert(A.length) // = 2
```

Ініціалізація елементів масиву

1. Спосіб ініціалізації масиву

Приклад:

```
var A = new Array();  
A[0] = 25;  
A[1] = 105;  
A[2] = -7;
```

2. Спосіб ініціалізації масиву

Приклад:

```
var B = [];
```

Додати елемент

```
B.push(108); // B[0] = 108
```

3. Спосіб. Уведення елементів масиву

Приклад:

```
var C = [];  
var count = 7;
```

Практичний web-дизайн: проектування, створення та супроводження web-вузла

```
for(var i = 0; i < count; i++)  
{  
    C[i] = + prompt('введіть' + i + ' елемент масиву', '');  
}
```

Приклад. Вивід елементів масиву в таблицю

```
for(var i = 0; i < C.length; i++)  
{  
    document.writeln("<Table border> <TR align = center><TH  
width = 10>"+(i+1)+ " </TH><TD width = 100>"+C[i]+  
"</TD></TR></Table>");  
}
```

Приклад. Підрахувати суму і кількість елементів масиву кратних 6

```
var D = [24, -6, 7, 18]  
var k = 0, s = 0;  
for(var i = 0; i < D.length; i++)  
    if (D[i] % 6 == 0)  
    {  
        k++;  
        s += D[i];  
    }  
alert('Кількість елементів, кратних 6' +':\n' + k);  
alert('Сума елементів, кратних 6' +':\n' + s);
```

4.6 Модель подій

Модель подій підтримує події, що генеруються користувачем при виконанні певних дій на сторінці з використанням миші, клавіатури, елементів інтерфейсу і т. ін.

Табл. 4.8

Події

Обробники подій	Пояснення
<i>onClick</i>	генерується під час кліку лівою кнопкою миші по елементу
<i>onContextmenu</i>	генерується під час кліку правою кнопкою миші по елементу
<i>ondblclick</i>	генерується під час подвійного кліку кнопкою миші по елементу

Закінчення табл. 4.8

<i>onHelp</i>	генерується після натискання клавіши F1
<i>onKeyDown</i>	генерується при натисканні клавіши клавіатури (в процесі натискання)
<i>onKeyPress</i>	генерується при натисканні клавіши буквено-цифрового ряду (коли клавіша клавіатури повністю натиснута)
<i>onKeyUp</i>	генерується, коли користувач відпускає клавішу клавіатури
<i>onMouseDown</i>	генерується при натисканні кнопки миші на елементі
<i>onMouseMove</i>	генерується при пересуванні миші по елементу
<i>onMouseout</i>	генерується, коли курсор миші залишає границі елемента
<i>onMouseover</i>	генерується при наведенні курсору миші на елемент
<i>onMouseup</i>	генерується, коли користувач відпускає кнопку миші

Приклад. Подія onmousemove

```
function p_over()
{
    alert("Курсор наведено на абзац");
}
<p onmouseover="p_over()"> Текст абзацу </p>
```

4.7 Основи об'єктної моделі документа

Об'єктна модель документа (*Document Object Model – DOM*) забезпечує програмний інтерфейс (*API*) для HTML документів. Вона визначає логічну структуру документів і способи взаємодії з ними.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

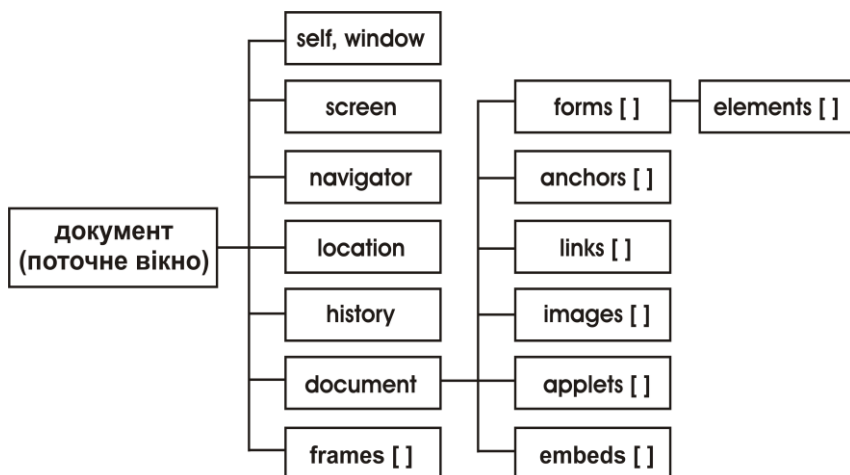


Рис. 4.1. Ієрархія об'єктів у браузерях

Об'єкт *Screen* дозволяє отримувати роздільну здатність екрана та глибину кольору.

Об'єкт *Navigator* надає інформацію про версію браузера.

Об'єкт *Location* надає доступ до URL-документа.

Об'єкт *History* містить кількість переглянутих у поточному сеансі документів, і дозволяє пересуватися історією переглядів.

Табл. 4.9

Властивість *document*

Властивість	Пояснення
all	властивість для доступу до будь-якого об'єкта сторінки
styleSheets	посилається на пов'язані із документом таблиці стилів
scripts	список вбудованих у документ сценаріїв
frames	список елементів IFRAME
embeds	список елементів EMBED (відео, флеш, аудіо)
links	список посилань (елементів A з атрибутом HREF)

Закінчення табл. 4.9

anchors	список якорів (елементів А з атрибутом NAME)
images	список зображень (елементів IMG)
forms	список форм (елементів FORM)

Доступ до об'єктів документа. Пошук елементів у DOM

Табл. 4.10

Методи доступу до об'єктів документа

Метод	Пояснення
<i>getElementById</i>	пошук елемента за id
<i>getElementsByTagName</i>	пошук елемента за тегом
<i>getElementsByName</i>	пошук елемента за name

Пошук за id

Для цього застосовується виклик:

document.getElementById("ідентифікатор_елемента");

Приклад. Змінити колір та ширину

```

p {
  background:darkred;
  color: yellow; }

<p id="p_1">Текст абзацу </p>

document.getElementById('p_1').style.background = 'blue';
document.getElementById("p_1").style.width="50%";
    
```

Пошук за тегом

Для цього застосовується виклик:

document.getElementsByTagName("назва_тегу");

Повертається масив елементів, що мають указаний тег.

Пошук за іменем

Для цього застосовується виклик:

document.getElementsByName("ім'я_елемента");

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Повертається масив елементів, у яких ім'я (атрибут *name*) дорівнює значенню *ім'я_елемента*. Виклик може бути застосований тільки для елементів, в яких у специфікації явно передбачено атрибут *name*: *form*, *input*, *a*, *select*, *textarea* і т. ін.

Властивість `innerHTML`

Властивість `innerHTML` дозволяє отримати доступ до вмісту елемента, який зберігається в ньому у вигляді рядка.

Синтаксис:

елемент.`innerHTML`;

Приклад. Отримати текст абзацу

```
<p id="p_2">Текст абзацу</p>
```

```
var p_ = document.getElementById('p_2');  
alert(p_.innerHTML);
```

РОЗДІЛ 5

JAVASCRIPT-БІБЛІОТЕКА JQUERY

5.1 Введення в jQuery

jQuery – швидка, компактна, потужна JavaScript-бібліотека.

З її допомогою можна взаємодіяти з HTML-документом, обробляти події, реалізовувати анімаційні ефекти і працювати з AJAX набагато простіше завдяки зручному API, який до того ж ще й кросбраузерний.

5.2 Робота з jQuery

Підключити бібліотеку jQuery можна двома способами:

1. Розмістити посилання на бібліотеку через jQuery CDN:

```
<script src="https://code.jquery.com/jquery-3.3.1.min.js">
</script>
```

2. Розмістити посилання на локальну копію бібліотеки:

```
<script src="jquery-3.3.1.js"></script>
```

Офіційний сайт бібліотеки: <http://jquery.com/>

Функція **jQuery()** – основна функція в бібліотеці, що дозволяє здійснювати пошук елементів на web-сторінці, створювати нові елементи, маніпулювати ними і багато іншого. Повертає об'єкт jQuery.

Для спрощення написання коду використовують скорочений запис функції *jQuery()* – функцію **\$()**.

У якості аргументів вона бере селектори (елементи), над якими слід виконати якісь дії.

```
$( "p" ); // вибір усіх параграфів на web-сторінці
```

Перш ніж починати виконувати JavaScript-код, потрібно бути впевненим у завантаженні всіх елементів сторінки (завантаженні DOM-дерева). У класичному JavaScript-кодi найчастіше відслідковується повне завантаження сторінки подією `onload`:

Практичний web-дизайн: проектування, створення та супроводження web-вузла

```
window.onload() = function() {  
  // тіло функції  
};
```

У jQuery використовується подія ready, яка відбувається на момент готовності DOM-дерева:

```
$(document).ready(function() {  
  // тіло функції  
});
```

Базові селектори

Селекторами називають рядкові вирази, за допомогою яких задаються умови пошуку елементів DOM-дерева на web-сторінці.

Селектори дозволяють знаходити елементи за різними ознаками: значенням атрибутів, вмісту елементів, батьківським елементом, дочірнім елементом, порядковими номерами, і звичайно, по іменах класів, ідентифікаторів та/або тегів.

Наприклад, вираз `$("div")` здійснює пошук усіх `div`-елементів на сторінці.

Вибірка за іменем тегу:

```
<script>  
...  
$("a")... інші дії  
...  
</script>  
<a href="#">...</a>
```

`$("a")` – відповідає всім елементам з ім'ям тегу `"a"`.

Вибірка за іменем класу:

```
<script>  
...  
$(".yellow")... інші дії  
...  
</script>  
<div class="yellow">...</div>  
<div class="green">...</div>  
<div class="yellow">...</div>
```

`$(".yellow")` – вибір елементів, з ім'ям класу `"yellow"`.

Вибірка за ідентифікатором:

```
<script>
```

```
...
$("#reg_form")... інші дії
...
</script>
<div id="reg_form">...</div>
```

`$("#reg_form")` – вибір елемента, з ім'ям ідентифікатора `reg_form`.

Як аргумент селекторної функції можна передавати будь-які селектори з CSS.

Табл. 5.1

Таблиця вибірок елементів

Конструкція	Пояснення	Приклад
<code>*</code>	вибір всіх елементів на сторінці	<code>\$("*")</code>
<code>\$("EF")</code>	вибір всіх елементів з ім'ям тегу F, вкладених в елемент з ім'ям тегу E	<code>\$("div a")</code>
<code>\$("E,G,H")</code>	вибір всіх елементів з іменами тегів E, G, H	<code>\$("div,span,.post")</code>
<code>\$("E>F")</code>	вибір всіх елементів з ім'ям тегу F, які є прямими нащадками елементів з ім'ям тегу E	<code>\$("div>a")</code>
<code>\$("E+F")</code>	вибір усіх елементів з ім'ям тегу F, яким безпосередньо передуює елемент E на тому ж рівні вкладеності	<code>\$("p+a")</code>
<code>\$("E~F")</code>	вибір всіх елементів з ім'ям тегу F, яким передуює елемент E на тому ж рівні вкладеності	<code>\$("span~p")</code>

Селектор `$("*")`

```
$( "div *" ).css ( "background-color", "red" );
```

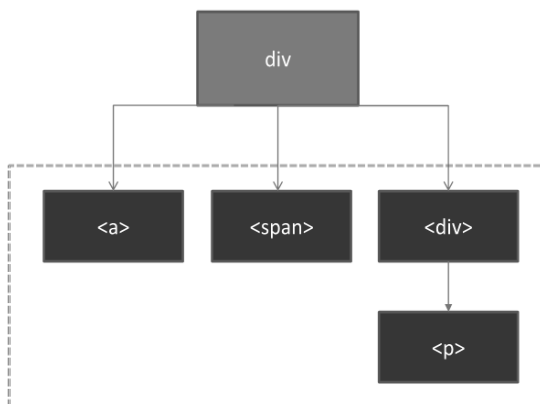


Рис. 5.1. Використання селектора \$("*")

Селектор \$("E F")

```
$("#main span").css("background-color", "red");
```

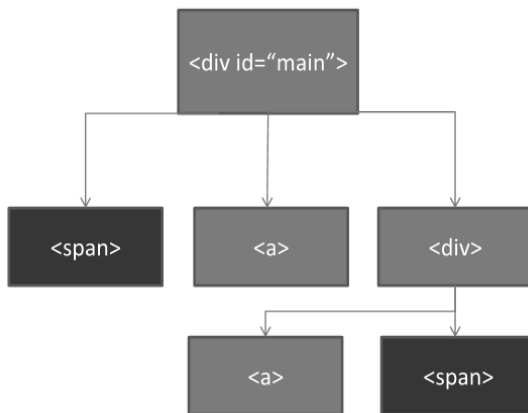


Рис. 5.2. Використання селектора \$("E F")

Будуть вибрані всі теги "span", які знаходяться в тегу з ідентифікатором "main" на будь-якому рівні вкладеності.

Селектор \$("E,G")

```
$("#span, a").css("background-color", "red");
```

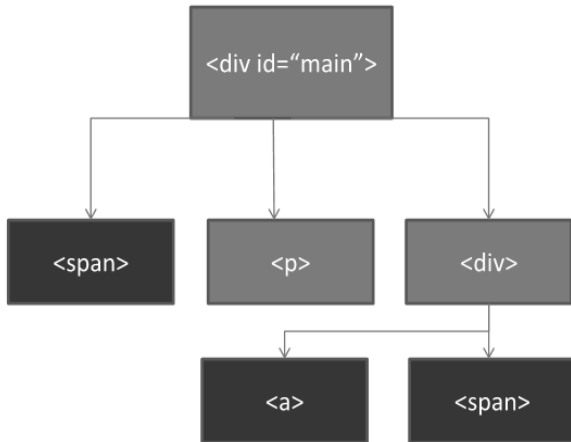


Рис. 5.3. Використання селектора $$(\"E,G\")$

Будуть вибрані всі теги-елементи, які є або тегами *"span"*, або *"a"*.

Селектор $$(\"E>F\")$

```
$(\"#main>span\").css(\"background-color\", \"red\");
```

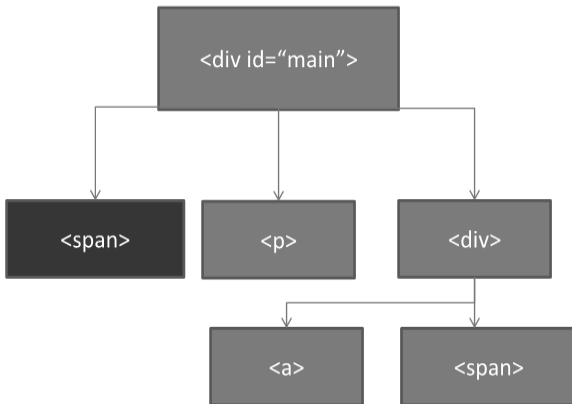


Рис. 5.4. Використання селектора $$(\"E>F\")$

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Будуть вибрані всі теги *"span"*, які є прямими нащадками тегу з ідентифікатором *"main"*.

Селектор $\$("E+F")$

```
| $("span+p").css("background-color", "red");
```

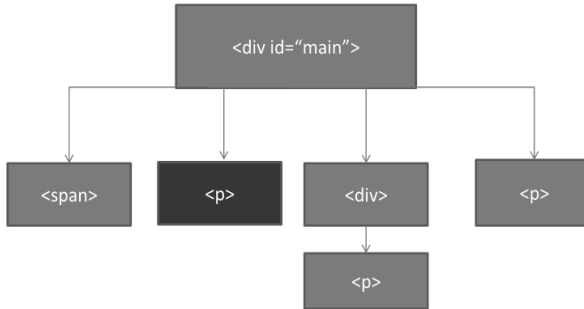


Рис. 5.5. Використання селектора $\$("E+F")$

Будуть вибрані всі теги *"p"*, які розташовані відразу після тегу *"span"*.

Селектор $\$("E~F")$

```
| $("span~p").css("background-color", "red");
```

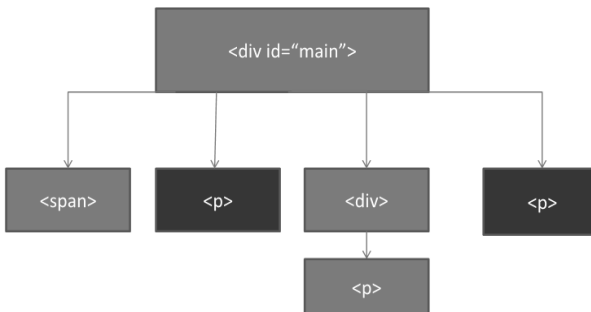


Рис. 5.6. Використання селектора $\$("E~F")$

Будуть вибрані всі теги *"p"*, які розташовані після тегу *"span"* на тому ж рівні вкладеності.

5.3 Фільтри в jQuery

Для більш тонкої вибірки елементів web-сторінки використовують спеціальні фільтри.

Основні групи фільтрів у jQuery:

1. базові фільтри;
2. фільтри контенту;
3. фільтри атрибутів;
4. фільтри елементів форм.

5.3.1 Базові фільтри

Табл. 5.2

Базові фільтри в jQuery

Конструкція	Пояснення	Приклад
<i>:even</i>	вибір парних елементів на сторінці	<code>\$("table tr:even")</code>
<i>:odd</i>	вибір непарних елементів на сторінці	<code>\$("table tr:odd")</code>
<i>:eq(index)</i>	вибір елемента з масиву за індексом	<code>\$("table tr:eq(4)")</code>
<i>:first</i>	вибір першого збігу на сторінці	<code>\$("table tr:first")</code>
<i>:last</i>	вибір останнього збігу на сторінці	<code>\$("table tr:last")</code>

Базовий фільтр *:even*

:even – вибір парних за індексом елементів на сторінці.

```

$ ("p:even").css ("color", "Red");

```

Параграф 0
 Параграф 1
 Параграф 2
 Параграф 3
 Параграф 4
 Параграф 5
 Параграф 6

Парні параграфи

Рис. 5.7. Використання фільтра *:even*

Базовий фільтр *:odd*

:odd – вибір непарних за індексом елементів на сторінці.

```
$( "p:odd" ).css( "color", "Red" );
```

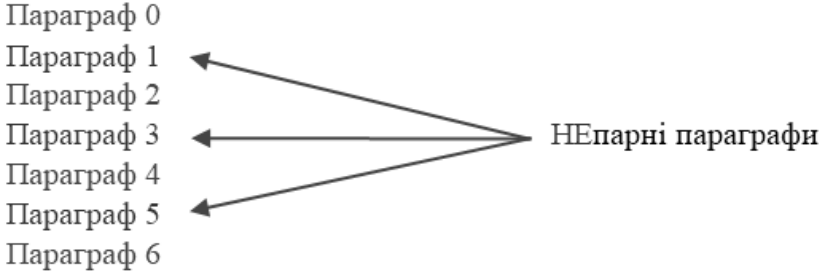


Рис. 5.8. Використання фільтра *:odd*

Базовий фільтр *:eq(index)*

:eq(index) – вибір елемента з масиву за індексом.

```
$( "p:eq(2)" ).css( "color", "Red" );
```

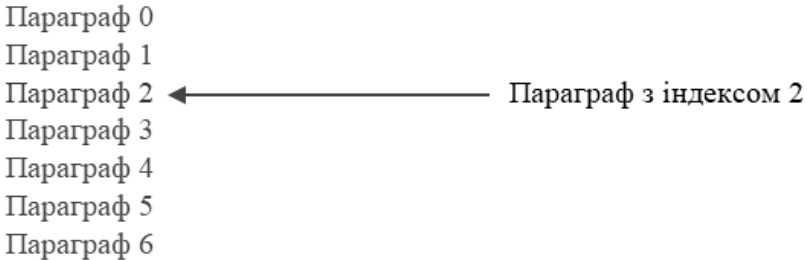


Рис. 5.9. Використання фільтра *:eq*

Базовий фільтр *:lt(index)*

:lt(index) – вибір елементів, розташованих перед n-м елементом (крім нього).

```
$( "p:lt(3)" ).css( "color", "Red" );
```

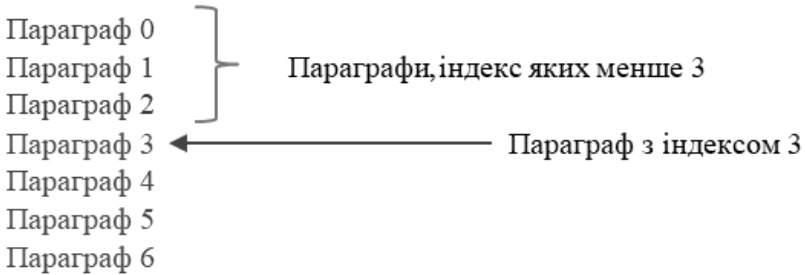


Рис. 5.10. Використання фільтра :lt(index)

Базовий фільтр :gt(index)

:gt(index) – вибір елементів, розташованих після n-го елемента (крім нього).

```
$( "p:gt(3)" ).css("color", "Red");
```

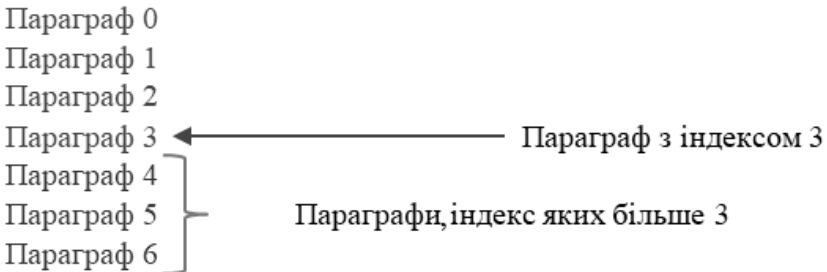


Рис. 5.11. Використання фільтра :gt(index)

Базовий фільтр :not(filter/selector)

:not(filter/selector) – вибір усіх елементів, крім тих, які потрапляють під вибірку, зазначену в круглих дужках.

```
$( "p:not(:eq(3))" ).css("color", "Red");
```


Параграф 0
 Параграф 1
 Параграф 2
 Параграф 3 ←————— Параграф з індексом 3
 Параграф 4
 Параграф 5
 Параграф 6

Рис. 5.12. Використання фільтра `:not(filter|selector)`

5.3.2 Фільтри контенту

Табл. 5.3

Фільтри контенту в jQuery

Конструкція	Пояснення	Приклад
<code>:contains("text")</code>	вибір елементів, що містять "text"	<code>\$("p:contains('JavaScript'))</code>
<code>:empty</code>	вибір усіх елементів, у яких відсутній вміст	<code>\$("div:empty")</code>
<code>:has()</code>	вибір усіх елементів, що містять певний у параметрах елемент	<code>\$("div:has(p)")</code>

Фільтр `:has(tag)`

`:has(tag)` – вибір усіх елементів, що містять певний у параметрах елемент.

```
$("div:has(a)").css("color", "red");
```

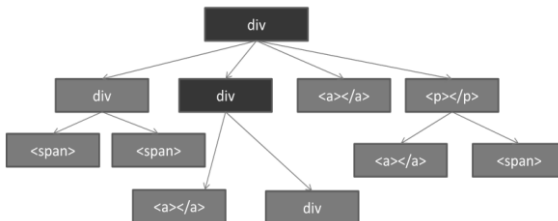


Рис. 5.13. Використання фільтра `:has(tag)`

5.3.3 Додаткові фільтри атрибутів

Табл. 5.4

Додаткові фільтри атрибутів

Конструкція	Пояснення	Приклад
<code>\$(<i>"E[AttrName~='value']"</i>)</code>	вибір усіх елементів E з атрибутом AttrName, що містить value в списку значень, розділених пробілом	<code>\$(<i>"img[alt~='new']"</i>)</code>
<code>\$(<i>"E[AttrName!='value']"</i>)</code>	вибір усіх елементів E з атрибутом AttrName не рівним значенню value	<code>\$(<i>"a[target!=_blank]"</i>)</code>

Фільтр `E[AttrName~='value']`

Фільтр `E[AttrName~='value']` – вибір усіх елементів E з атрибутом AttrName, що містить value в списку значень, розділених пробілом.

```
$("img[alt~='new']").css("border-color", "green");

```

5.3.4 Фільтри елементів форм

Табл. 5.5

Фільтри елементів форм

Конструкція	Пояснення	Приклад
<code>:<i>button</i></code>	вибір усіх кнопок на сторінці	<code>\$(<i>":button"</i>)</code>
<code>:<i>checkbox</i></code>	вибір усіх елементів "checkbox"	<code>\$(<i>":checkbox"</i>)</code>
<code>:<i>checked</i></code>	вибір лише позначених прапорців або перемикачів	<code>\$(<i>"input:checked"</i>)</code>

<i>:disabled</i>	вибір тільки тих елементів форм, які знаходяться в неактивному стані	<code>\$("input:disabled")</code>
<i>:enabled</i>	вибір тільки тих елементів форм, які знаходяться в активному стані	<code>\$("input:enabled")</code>

5.3.5 Маніпуляція атрибутами в jQuery

У jQuery для маніпуляції атрибутами є дві основні функції:

1. функція **attr()** дозволяє отримати доступ до HTML атрибуту першого тегу на сторінці, який був обраний селектором;
2. функція **removeAttr()** – видаляє вказаний атрибут з елемента.

Функція attr()

Функція **attr()** повертає або змінює значення атрибутів у вибраних елементів сторінки.

Є кілька варіантів використання цієї функції:

Читання значення атрибута .attr(attrName). Повертає значення атрибута attrName у вибраного елемента. Якщо вибрано кілька елементів, то значення буде взято у першого.

Зміна значення атрибута .attr(attrName, value). Атрибуту attrName буде присвоєно значення value, у всіх вибраних елементів.

Зміна або встановлення декількох атрибутів `.attr({attrName1: value1, attrName2: value2})`. Групі атрибутів attrName1, attrName2, ..., будуть присвоєні значення value1, value2, ..., у всіх вибраних елементів.

Зміна або встановлення значення атрибута за допомогою функції `.attr(attrName, function (index, value))`. Атрибуту attrName буде присвоєно значення, що повертає функція користувача (якщо вона нічого не повертає, то значення атрибута

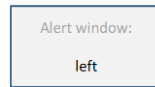
залишитися без змін). Функція викликається окремо для кожного з вибраних елементів. При виклику їй передаються такі параметри: `index` – позиція елемента в наборі, `value` – поточне значення атрибута `attrName` у елемента.

Функція `attr("атрибут")`

Функція `attr("attrName")` повертає значення атрибута `attrName` у вибраного елемента.

Код виглядає таким чином:

```
var divAlign = $("div").attr("align");  
alert(divAlign);  
...  
<div align="left">...</div>  
...
```



Якщо на сторінці присутні кілька елементів, що потрапили під вибірку, то тоді повернеться значення атрибута першого з набору елементів.

Функція `attr("атрибут", "значення")`

Функція `attr("attrName", "value")` задає атрибуту `attrName` значення `value`, у всіх вибраних елементів.

Код виглядає таким чином:

```
$("div").attr("align", "left");  
...  
<div>...</div>  
<div align="right">...</div>  
<div>...</div>  
...
```

Атрибут `align` буде встановлений для всіх елементів з набору в значення «left», незалежно від того, був він у них визначений раніше чи ні.

Функція `attr({атрибут: "значення", ...})`

Функція `attr({атрибут: "значення", ...})` встановлює значення відразу для декількох атрибутів. Для цього необхідно передавати об'єкт з властивостями, імена яких збігаються з іменами властивостей, а значення зі значеннями, які необхідно встановити.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Код виглядає таким чином:

```
$( "div" ).attr (
{
style: "color:red; font-size:22pt",
title: "jQueryAttrTest"
}
);
...
<div>...</div>
<div align="right">...</div>
<div>...</div>
...
```

Функція *attr("атрибут",функція())*

Функція *attr("attrName",функція())* задає атрибуту attrName значення, що повертає функція користувача (якщо вона нічого не повертає, то значення атрибута залишиться без змін). Функція викликається окремо для кожного з обраних елементів. При виклику їй передаються такі параметри: index – позиція елемента в наборі, value – поточне значення атрибута attrName у елемента.

Код виглядає таким чином:

```
$( "p:first" ).attr (
"style", CalculateValue ()
);
...
function CalculateValue () {
return "color: red; "
}
...
<p>...</p>
<p>...</p>
...
```

Функція *removeAttr("атрибут")*

Функція *removeAttr("attrName")* видаляє атрибут attrName у вибраних елементах на сторінці.

Код застосунку виглядає таким чином:

```
$( "div" ).removeAttr ("align");
...
<div>...</div>
```

```

п51
<div align="right">...</div>
<div>...</div>
...

```

У результаті атрибут видалиться в усіх елементів з набору, якщо він був визначений; якщо атрибут не був визначений, то нічого не відбудеться.

5.3.6 Функції для маніпуляції класами

У jQuery для маніпуляції класами існують спеціальні функції (табл. 5.6).

Табл. 5.6

Функції для маніпуляції класами

Функція	Пояснення	Приклад
addClass()	додає клас(и) обраним елементам сторінки (замінюючи їх)	<code>\$("#table tr:even").addClass("Class1");</code>
removeClass()	видаляє задані класи в елементів на сторінці	<code>\$("#table tr:odd").removeClass("Class1");</code>
hasClass()	перевіряє наявність класу в елемента і повертає <i>true</i> якщо клас є або <i>false</i> якщо немає	<code>If(\$("#p:first").hasClass("Class1")) { ... }</code>
toggleClass()	додає або видаляє заданий клас(и) за принципом перемикача	<code>\$("#table tr:first").toggleClass("class1");</code>

Функція `addClass("ім'я класу")`

`addClass("ім'я класу")` – функція, яка додає вказаний клас для обраного елемента, або набору елементів.

Приклад застосування функції:

```

$("#div").addClass("Class1");

```

Можна було б зробити ту ж дію за допомогою функції `attr:`

Практичний web-дизайн: проектування, створення та супроводження web-вузла

```
| $("div").attr("class", "Class1");
```

Важливо зауважити, що функція не замінює значення атрибута `class`, а додає ще один клас до загального набору.

Функція `removeClass("ім'я класу")`

`removeClass("ім'я класу")` – функція, яка видаляє вказаний клас у обраного елемента, або набору елементів.

Приклад застосування функції:

```
| $("div").removeClass("Class1");  
...  
<div class= "Class1 Class2 Class3">...</div>
```



```
| <div class= "Class2 Class3">...</div>
```

У результаті в елементів набору буде видалений саме той клас, який був переданий як аргумент методу `removeClass()`.

Функція `hasClass("ім'я класу")`

`hasClass("ім'я класу")` – функція, яка перевіряє наявність класу в елемента і повертає `true`, якщо клас є, або `false`, якщо немає.

Приклад застосування функції:

```
| if ($("div:first").hasClass("Class1"))  
{  
...  
}  
...
```

Якщо викликати цю функцію на наборі елементів, а не на одному, то тоді слід пам'ятати, що перевірка на наявність класу пройде тільки в першому.

Функція `toggleClass("ім'я класу")`

`toggleClass("ім'я класу")` – функція, яка додає або видаляє заданий клас(и) за принципом перемикача.

Приклад застосування функції:

```
| $("p").toggleClass("Class1");
```

```
...  
<p>...</p>  
<p class="Class1">...</p>  
<p>...</p>
```



```
...  
<p class="Class1">...</p>  
<p>...</p>  
<p class="Class1">...</p>
```

5.4 Робота з HTML та CSS в jQuery

5.4.1 Методи jQuery для роботи з html

html(val) – додає html-код у вибрані елементи.

```
$( "div.sp" ).html ( "<span>Hello</span>" );
```

text()/val() – повертає текстовий вміст елемента / значення елемента.

```
$( "p" ).text ( );
```

text(val)/val(val) – встановить текст для елемента / значення для елемента.

```
$( "p" ).text ( "Hello!" );
```

5.4.2 Методи jQuery для роботи з CSS

Функція css("властивість")

css("styleName") – повертає значення css-властивості styleName у обраного елемента. Якщо вибрано кілька елементів, то значення буде взято у першого.

Приклад застосування функції:

```
$( "div" ).css ( "width" );
```

...

```
<div style="width: 300px">...</div>
```

...

Функція css("властивість", "значення")

css("styleName", "value") – css-властивості styleName буде присвоєно значення value, у всіх обраних елементів.

Приклад застосування функції:

```
$( "p" ).css ( "color", "green" );
```

...

```
<p>...</p>
```

```
<p>...</p>
```

```
<p>...</p>
```

...



Функція `css({набір властивостей})`

`css({styleName1:"value1", ... })` – групі `css`-властивостей `styleName1, ...` будуть присвоєні значення `value1, ...` у всіх обраних елементах.

Приклад застосування функції:

```
$( "p" ).css ( { backgroundcolor: "Red", fontsize: "18pt" } );  
...  
<p>...</p>  
<p>...</p>  
<p>...</p>  
...
```



5.4.3 Маніпуляція елементами, що вставляються

`append(content)` – додає `content` всередину всіх обраних елементів ПІСЛЯ існуючого контенту.

```
$( "p" ).append ( "<b>Hello</b>" );
```

`appendTo(content)` – зворотна операція, додає в `content` усі вибрані елементи.

```
$( "p" ).appendTo ( "#main" );
```

`$(A).append(B)` – додає **B** у кінець **A**.

`$(A).appendTo(B)` – додає **A** у кінець **B**.

`prepend(content)` – додає `content` всередину всіх обраних елементів ДО існуючого контенту.

```
$( "p" ).prepend ( "<b>Hello</b>" );
```

`prependTo(content)` – зворотна операція, додає в `content` усі вибрані елементи.

```
$( "p" ).prependTo ( "#main" );
```

`$(A).prepend(B)` – додає **B** на початок **A**.

`$(A).prependTo(B)` – додає **A** на початок **B**.

`after(content)` – додає `content` ПІСЛЯ всіх обраних елементів (зауважте ПІСЛЯ елемента, а не в кінець елемента, як у випадку з `append`).

```
$( "p" ).after ( "<b>Hello</b>" );
```

insertAfter(content) – додає всі вибрані елементи ПІСЛЯ content-a.

```
| $("p").insertAfter("#main");
```

\$(A).after(B) – додає **B** після **A**.

\$(A).insertAfter(B) – додає **A** після **B**.

before(content) – додає content ДО всіх обраних елементів (зауважте ДО елемента, а не в початок елемента, як у випадку з prepend).

```
| $("p").before("<b>Hello</b>");
```

insertBefore(content) – додає всі вибрані елементи ПЕРЕД content-ом.

```
| $("p").insertBefore("#main");
```

\$(A).before(B) – вставить **B** перед **A**.

\$(A).insertBefore(B) – вставить **A** перед **B**.

wrap(html) – обертає кожен елемент в елемент-обгортку.

```
| $("p").wrap("<span></span>");
```

wrapAll(html) – обертає всі елементи в ОДИН елемент-обгортку.

```
| $("p").wrapAll("<span></span>");
```

wrapInner(html) – обертає внутрішній зміст кожного елемента.

```
| $("p").wrapInner("<em></em>");
```

replaceWith(content) – заміщає одні елементи іншими.

```
| $("span").replaceWith("<div></div>");
```

replaceAll(selector) – зворотна операція, тобто всі selector-и будуть замінені на елементи.

```
| $("span").replaceAll("<div></div>");
```

empty() – видаляє з елемента всі вузли-нащадки, включаючи текст.

```
| $("span").empty();
```

remove() – видаляє самі елементи.

```
| $("span").remove();
```

clone() – клонує елементи.

```
| $("b").clone().append("div");
```

5.5 Робота з DOM в jQuery

HTML-документи мають ієрархічну структуру, представлену в DOM-дереві. Вузли дерева представляють різні типи вмісту документа.

У першу чергу, деревоподібне уявлення HTML-документа містить вузли, що представляють елементи, такі як `<a>`, `<p>`, `<div>` і вузли, що представляють рядки тексту.

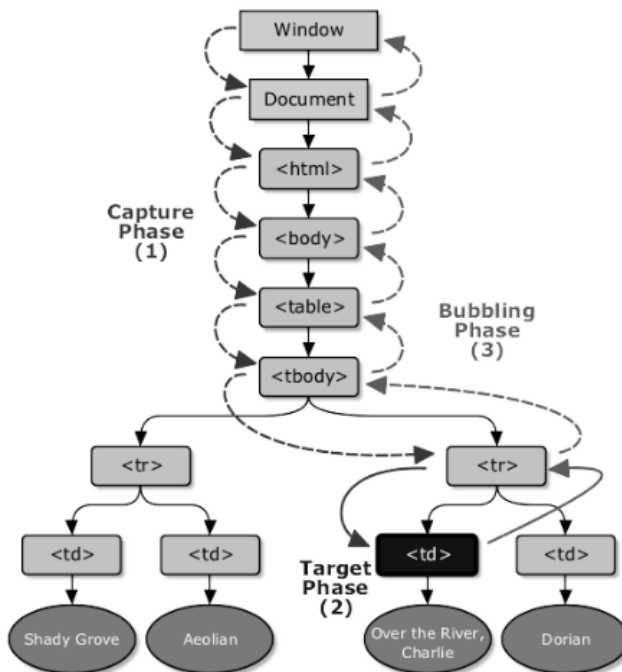


Рис. 5.14. DOM дерево

jQuery – потужна бібліотека JavaScript, орієнтована на спрощення взаємодії з HTML через JavaScript.

У jQuery передбачена робота з DOM-деревом за допомогою спеціальних методів.

5.5.1 Методи фільтрації набору елементів

Табл. 5.7

Методи фільтрації елементів

Функція	Пояснення
.eq(index)	скорочує число елементів, що співпали, до одного; аргументом є позиція елемента в наборі елементів, що співпали
.not(selector)	видаляє елементи, що відповідають вказаному виразу, з набору елементів, які співпали
.filter(function)	видаляє всі елементи, які не задовольняють вимогам функції, з набору елементів, що співпали
.is(selector)	перевіряє поточний набір елементів на відповідність вказаному виразу і повертає <i>true</i> , якщо хоча б один елемент відповідає виразу
.slice(index, index)	виділяє підмножину з набору елементів, що співпали

Метод eq(index)

.eq(index) – скорочує число елементів, що співпали, до одного. Аргументом є позиція елемента в наборі елементів, що співпали, значення *index* починається в межах від 0 до *length-1*. Некоректне значення індексу тягне за собою повернення порожнього набору елементів, а не нуля, оскільки запит відфільтровує всі елементи, які не відповідають указаному індексу.

```

$ ("p").eq(2).css("color", "Red");

```

[0]	<p> Параграф №1 </p>	Параграф №1
[1]	<p> Параграф №2 </p>	Параграф №2
[2]	<p> Параграф №3 </p>	Параграф №3
[3]	<p> Параграф №4 </p>	Параграф №4
[4]	<p> Параграф №5 </p>	Параграф №5
[5]	<p> Параграф №6 </p>	Параграф №6



 Буде вибраний елемент з попередньої вибірки з індексом "2"

Рис. 5.15. Використання методу eq(index)

Метод not(selector)

.not(selector) – видаляє елементи, відповідні вказаному виразу, з набору елементів, що співпали. Вираз може бути як окремим селектором, так і складною селекторною конструкцією.

```
$( "p" ).not ( "eq (2) " ).css ( "color", "Red" );
```

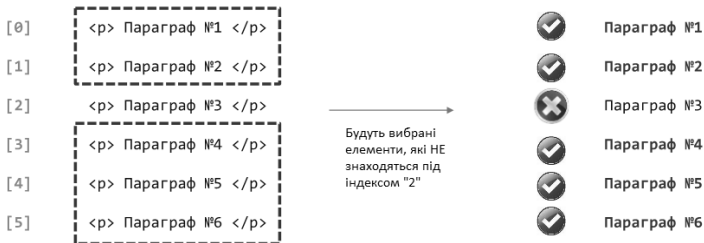


Рис. 5.16. Використання методу not(selector)

Методи filter()

.filter(selector) – видаляє всі елементи, які не відповідають вказаному виразу, з набору елементів, що співпали.

```
$( "p" ).filter ( ".class1" ).css ( "background-color", "Red" );
```

.filter(function). Якщо передати методу filter() деяку функцію, він викличе її для кожного елемента в наборі і виключить ті елементи, для яких ця функція поверне значення *false*.

```
$( "p" ).filter (
function (i) {
return i!=3 && i<7;
}
).css ( "background-color", "Red" );
```

Метод slice(start, stop)

.slice(start, stop) – виділяє підмножину з набору елементів, що співпали, починаючи з елемента з індексом start+1 до елемента з індексом stop (включно). Працює як вбудований метод поділу масивів (допускаються негативні значення).

```
$( "p" ).slice (2, 5) .css ( "color", "Red" );
```

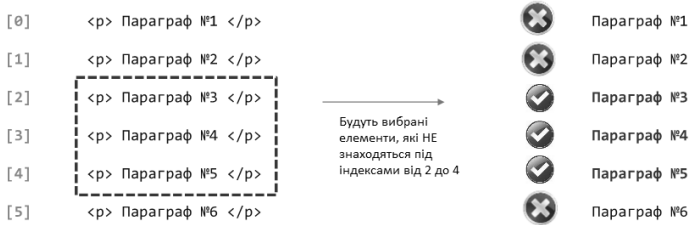


Рис. 5.17. Використання методу slice(start, stop)

5.5.2 Функції пошуку по DOM у jQuery

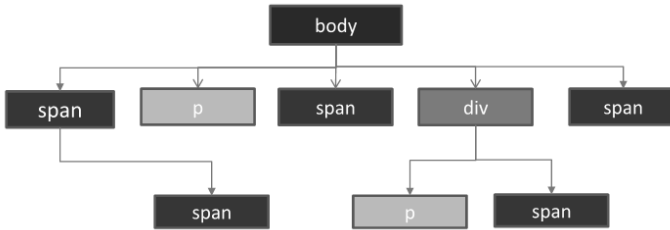


Рис. 5.18. Переміщення по DOM-дереву

Табл. 5.8

Функції пошуку по DOM в jQuery

Функція	Пояснення
.add(вираз)	додає у вже існуючий набір додаткові елементи, які задовольняють зазначеному виразу
.find(вираз)	здійснює пошук дочірніх елементів, які задовольняють зазначеному виразу
.contents()	пошук усіх дочірніх вузлів у наборі елементів, що співпали (включаючи текстові) або у вмісті документа, якщо він є фреймом
.children()	отримує набір елементів, що містить усіх безпосередніх унікальних нащадків для кожного елемента, що співпав
.sibling()	отримує набір елементів, що містить усі унікальні родинні елементи для набору елементів, що співпали.
.andSelf()	додавання попереднього набору до поточного.
.end()	скасовує останню деструктивну дію

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Метод add(selector)

.add(вираз) – додає у вже існуючий набір додаткові елементи, які відповідають зазначеному виразу.

```
$( "p" ).add( "div" ).css( "color", "Red" );
```

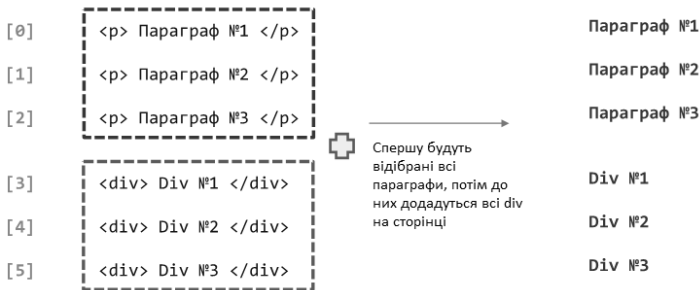


Рис. 5.19. Використання методу add(selector)

Метод end()

.end() – скасовує останній фільтр, повертаючи тим самим набір елементів до його попереднього стану (до фільтрації).

```
$( "p" ).add( "div" ).css( "color", "green" ).end().css( "color", "Red" );
```



Рис. 5.20. Використання методу end()

Метод find(selector)

.find(вираз) – здійснює пошук дочірніх елементів, які задовольняють зазначеному виразу.

```
$( "div" ).find( "span" ).css( "background-color", "Red" );
```

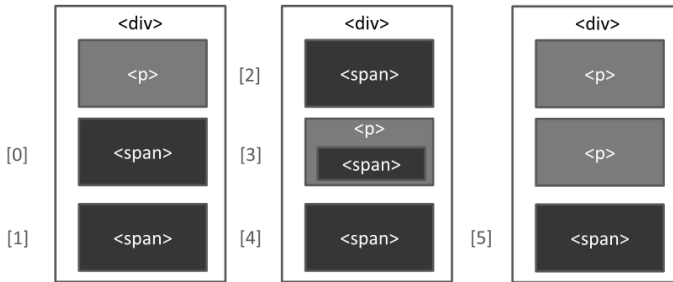


Рис. 5.21. Використання методу find(selector)

Метод children(selector)

.children(вираз) – отримує набір елементів, що містить усіх безпосередніх унікальних нащадків для кожного елемента, що співпав.

```
$( "div" ).children( "span" ).css( "background-color", "Red" );
```

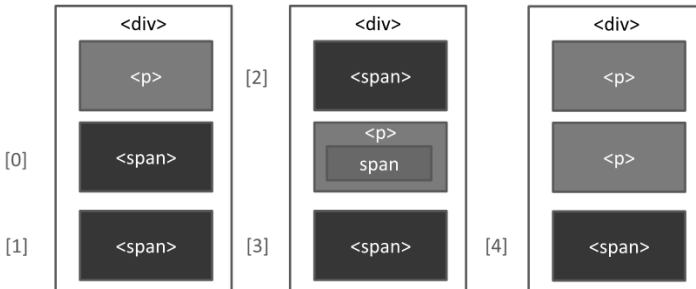


Рис. 5.22. Використання методу children(selector)

Табл. 5.9

Додаткові функції пошуку по DOM в jQuery

Функція	Пояснення
.parent(вираз)	отримує прямого "батька" елемента
.parents(вираз)	отримує набір елементів, що містить унікальних "батьків" для елементів, що співпали (крім кореневого елемента)

Закінчення табл. 5.9

.closest()	отримує набір, що містить найближчі батьківські елементи, які задовольняють зазначеному селектору, включаючи початковий елемент
.offsetParent()	повертає колекцію jQuery з "батьком" по позиціонування першого елемента, що співпав
.next(), .prev()	отримує набір елементів, що містить унікальні наступні (попередні) споріднені елементи для всіх елементів існуючого набору
.nextAll(), .prevAll()	пошук усіх споріднених елементів після поточного елемента (перед поточним елементом)
.nextUntil(селектор), .prevUntil(селектор)	пошук усіх споріднених елементів після поточного елемента (перед поточним елементом) до зазначеного селектора

Метод parent()

.parent() – повертає посилання на найближчого батька.

```
$( "p" ).parent().css( "background-color", "Red" );
```

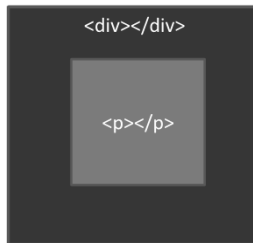


Рис. 5.23. Використання методу parent()

Метод parents()

.parents() – пошук усіх предків обраних елементів. Будуть обрані не тільки прямі батьки, а й прабатьки, прапрародичі, і так далі до початку DOM-дерева.

```
$( "p" ).parents().css( "background-color", "Red" );
```

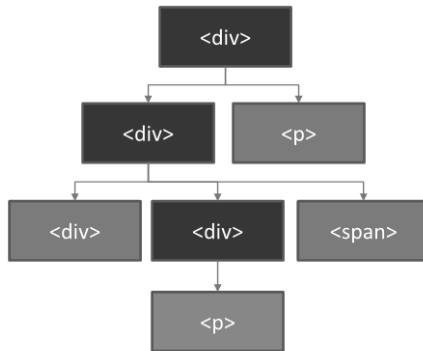


Рис. 5.24. Використання методу `parents()`

Метод `closest(selector)`

.closest(вираз) – починає пошук безпосередньо з обраного (обраних) елемента і рухається вгору ієрархією. Рухаючись вгору ієрархією зупиняє пошук після першого елемента, що виконує умови (після чого починає здійснювати пошук для наступного обраного елемента). Тому, знаходить не більше одного елемента для кожного з обраних.

```
$( "p" ).closest( "div" ).css( "background-color", "Red" );
```

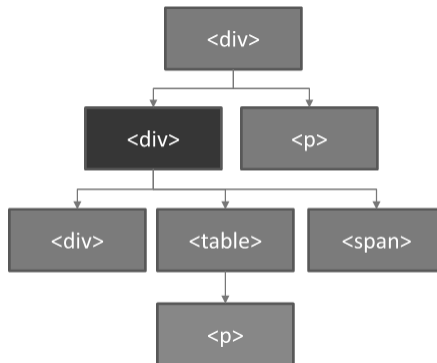


Рис. 5.25. Використання методу `closest(selector)`

Метод `next()`

Практичний web-дизайн: проектування, створення та супроводження web-вузла

.next() – отримує набір елементів, який містить тільки сусідні елементи, що слідують безпосередньо за вказаним елементом.

```
$( "p:first" ).next() .css ( "color", "Red" );
```

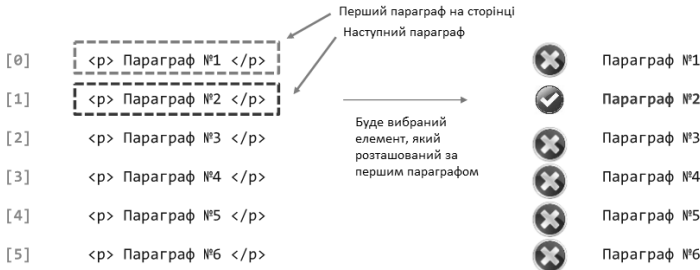


Рис. 5.26. Використання методу next()

Метод nextAll()

.nextAll() – знаходить усі наступні братські елементи, які відповідають селектору і йдуть після вибраного елемента.

```
$( "p:first" ).nextAll ( "p" ) .css ( "color", "Red" );
```

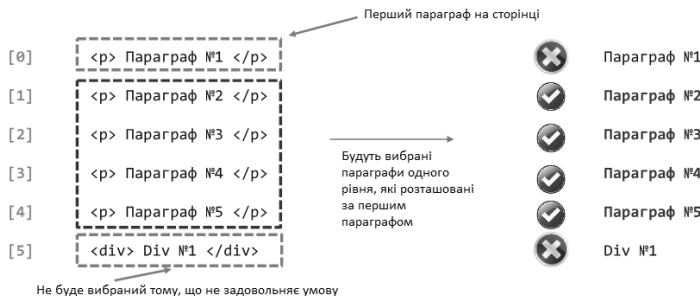


Рис. 5.27. Використання методу nextAll()

Метод nextUntil()

.nextUntil() – отримує всіх наступних братів кожного елемента до вибраного елемента, але не включаючи елемент, відповідний переданому селектору, вузлу DOM або об'єкта jQuery.

```
$( "p:first" ).nextUntill ( "p:eq (4)" ) .css ( "color", "Red" );
```

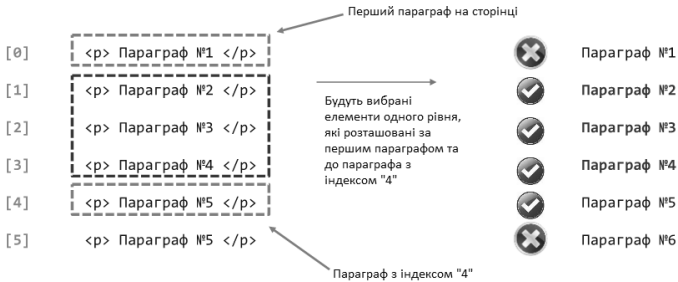


Рис. 5.28. Використання методу `nextUntil()`

5.5.3 Обробники подій у jQuery

`addEventListener('a', b(c){}, d)` – метод JavaScript, який призначений для додавання обробника («слухача») події. Він приймає такі параметри: *a* – назва події, за якої виконується заданий функціонал; *b* – функція-обробник події; *c* – об'єкт події, що передається у функцію-обробник; *d* – етап проходження події.

`removeEventListener('a', b, d)` – метод JavaScript, який призначений для ВИДАЛЕННЯ обробника («слухача») події. Він приймає такі параметри: *a* – назва події, за якої виконується заданий функціонал; *b* – функція, виконання якої необхідно скасувати; *c* – етап проходження події.

Обробники подій. Методи `on()` та `off()`

У JQuery аналогом функції `addEventListener()` виступає функція `on()`. Вона більш універсальна і кросбраузерна.

Для видалення обробника події будемо використовувати функцію `off()` – аналог `removeEventListener`.

Також є можливість прямого додавання події до елемента за допомогою методів подій:

```

| $("#button").click(function(){ ... });

```

Не забувайте, що у функцію-обробник події ми можемо передавати об'єкт події для використання додаткових властивостей і методів.

Момент виконання коду

Не забувайте, що для того, щоб код працював коректно, потрібно дочекатися або завантаження DOM-дерева, або завантаження всієї сторінки. Для досягнення першої мети ми повинні відстежити подію **ready()** на об'єкті `document`:

```
$(document).ready(SomeFunc);  
function SomeFunc()  
{  
...  
}
```

Для досягнення другої мети нам потрібно відстежити подію **load()** на об'єкті `window`:

```
$(window).load(SomeFunc);  
function SomeFunc()  
{  
...  
}
```

Набір основних подій миші:

- `click`;
- `dblclick`;
- `mousedown`;
- `mouseup`;
- `mousemove`;
- `mouseover`;
- `mouseout`.

Набір основних подій клавіатури:

- `keydown`;
- `keyup`;
- `keypress`.

Щоб ідентифікувати код натиснутої клавіші, необхідно передати в якості аргументу обробника об'єкт події, що була згенерована.

Код натиснутої клавіші буде зберігатися у властивостях *`event.keyCode`* і *`event.charCode`*.

Функції-обробники подій

Табл. 5.10

Функції для роботи з обробкою подій елементів у jQuery

Функція	Пояснення
.on()	функція додає обробник на вказану подію до елемента або набору елементів
.off()	функція видаляє обробник події з вибраного елемента
.one()	функція додає обробник на вказану подію до елемента або набору елементів, але викликається тільки ОДИН раз
.trigger()	функція дозволяє запустити подію іншого елемента управління на сторінці

Метод on()

on(подія, обробник) використовується для додавання обробників події до об'єкта або об'єктів. Він приймає в якості аргументів подію і функцію, яку необхідно виконати при виникненні певної події.

Так само можна передати обробнику, за допомогою JSON-об'єкта, набір параметрів, звернення до яких відбувається через *event.data.param_name*.

Для того щоб обробник міг використовувати параметри, необхідно передати йому посилання на подію, яка відбулась. Як аргумент вказати: *event* (звертатися потім до параметрів слід через вкладений об'єкт *data*).

```
$("#p").on("click", {par1:"Hello",par2:"World"},Handler);
function Handler(event) {
  alert(event.data.par1 + " " + event.data.par2);
}
```

Метод off()

off(подія, обробник) – видаляє з обраних елементів сторінки обробники подій, встановлені за допомогою методу **.on()**.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Якщо викликати функцію без аргументів, то видаляться всі обробники на всі події з цього елемента.

```
function show()
{
  alert('Hello world');
}
function remove()
{
  $('#first').off('click',
  'show');
}
```

```
$('#first').on('click', 'show');
$('#second').on('click', 'remove');
```



Метод trigger()

.trigger() – викликає подію для кожного елемента набору. Це також викличе виконання браузером дії за замовчуванням для цієї події. Наприклад, передача 'submit' у функцію **trigger()** спровокує відправку браузером форми. Дію за замовчуванням можна запобігти шляхом повернення значення *false* від однієї з функцій, пов'язаних з подією.

```
$("#p").on("mousemove", handler);
$("#button").on("click", function() {
  $("#p").trigger("mousemove");
})
```

Після натискання на кнопку в параграфі відбудеться подія "mousemove".

Методи one()

.one(подія, обробник) – прив'язує до вибраного елемента один обробник події або більше. Прив'язані обробники при цьому можуть бути викликані тільки один раз.

```
$("#p").one("click", handler);
```

5.6 Анімація та ефекти

5.6.1 Функція animate()

За допомогою функції **animate()** можна анімувати окремі елементи DOM-дерева

```
$("#selector").animate({styles}, duration, easing, function);
```

Параметри *styles* та *duration*

styles – значення стилів, які повинні бути задані певному об'єкту до кінця анімації.

```
$("#p").animate({left:"10px"}, duration, easing, function);
```

анімація руху елемента *p* з поточної позиції в позицію *left=10px*.

duration – швидкість анімації – "slow", "normal" або "fast".

Також можна вказати швидкість у мілісекундах.

```
$("#p").animate({left:"100px"}, 1000, easing, function);
```

анімація завершиться через 1 секунду (1000 мілісекунд).

easing

easing – ім'я ефекту ослаблення, який ви хочете використувати. Існує два вбудованих значення: «linear» і «swing».

```
$("#p").animate({color:"red";text-align:"center" },  
1000, linear, function);
```

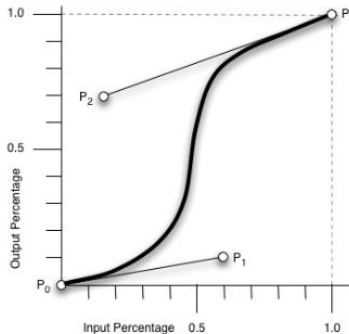


Рис. 5.29. Ефекти ослаблення

Complete function

Callback-функція запускається після закінчення анімаційних ефектів. Виконується один раз для кожного елемента.

```
$("#div").animate(  
{opacity: 0.25, marginLeft: '150px'},  
5000,  
function () {  
alert("Кінець заняття!");  
});
```


queue

.queue – логічне значення, яке вказує, чи слід додавати поточну анімацію в чергу функцій. У разі *false*, анімація буде завантажена відразу, не встаючи в чергу.



Рис. 5.30. Черга

5.6.2 Ефекти в jQuery

Табл. 5.11

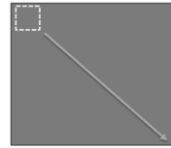
Методи створення ефектів

Функція	Пояснення
.show()	відображає кожен з елементів набору, якщо вони були приховані (або через функцію <i>hide()</i> , або <i>display:none</i>)
.hide()	приховує кожен з елементів набору, якщо вони були видимими; якщо всі елементи приховані, то нічого не відбудеться
.toggle()	перемикає режим відображення кожного з елементів набору
.slideDown()	відображає всі елементи набору, використовуючи ефект зміни висоти елементів
.slideUp()	приховує всі елементи набору, використовуючи ефект зміни висоти елементів
.slideToggle()	перемикає видимість елементів набору, використовуючи ефект зміни висоти елементів
.fadeIn()	робить видимими всі елементи набору, використовуючи зміну прозорості елементів.
.fadeOut()	робить невидимими всі елементи набору, потім встановлює CSS властивість <i>display</i> в « <i>none</i> »

.fadeTo()	робить більш/менш видимими всі елементи набору, змінюючи прозорість елементів до величини, зазначеної в аргументі
------------------	---

.show()

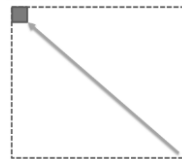
.show() – функція відображає кожен з елементів набору, якщо вони були приховані (або через функцію *hide()*, або *display:none*). Появу реалізовано збільшенням висоти і ширини по діагоналі.



1. **element.show()** – швидкість відображення за замовчуванням.
2. **element.show(speed)** – швидкість відображення передається як аргумент, при цьому можна вказати кількість мілісекунд або одне зі стандартних значень (slow, normal, fast).
3. **element.show(speed, function)** – другим аргументом можна вказати ім'я функції, яка виконується при завершенні анімації.

.hide()

.hide() – функція приховує кожен з елементів набору, якщо вони були видимими. Якщо всі елементи приховані, то нічого не відбудеться. Зникнення реалізоване зменшенням висоти і ширини по діагоналі.

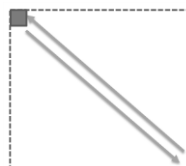


1. **element.hide()** – швидкість зникнення за замовчуванням.
2. **element.hide(speed)** – швидкість зникнення передається як аргумент, при цьому можна вказати кількість мілісекунд або одне зі стандартних значень (slow, normal, fast).

3. ***element.hide(speed,function)*** – другим аргументом можна вказати ім'я функції, яка виконається при завершенні анімації.

.toggle()

.toggle() – функція перемикає режим відображення кожного з елементів набору. Якщо елемент прихований, то функція відображає його (використовуючи метод show); якщо елемент видимий – приховує його (використовуючи метод hide).



1. ***element.toggle()*** – швидкість відображення за замовчуванням.
2. ***element.toggle(speed)*** – швидкість відображення передається як аргумент, при цьому можна вказати кількість мілісекунд або одне зі стандартних значень (slow, normal, fast).
3. ***element.toggle(bool)*** – як аргумент передається значення перемикача (*true* – відображає елементи, *false* – приховує елементи).
4. ***element.toggle(speed,function)*** – другим аргументом можна вказати ім'я функції, яка виконається при завершенні анімації.

.slideDown()

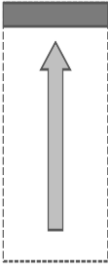
.slideDown() – функція приховує всі елементи набору, використовуючи ефект зміни висоти елементів.



1. ***element.slideDown()*** – швидкість зникнення за замовчуванням.
2. ***element.slideDown(speed)*** – швидкість зникнення передається як аргумент, при цьому можна вказати кількість мілісекунд або одне зі стандартних значень (slow, normal, fast).
3. ***element.slideDown(speed,function)*** – другим аргументом можна вказати ім'я функції, яка виконається при завершенні анімації.

.slideUp()

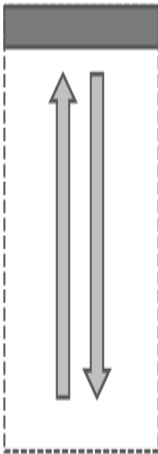
.slideUp() – функція приховує всі елементи набору, використовуючи ефект зміни висоти елементів.



1. ***element.slideUp()*** – швидкість появи за замовчуванням.
2. ***element.slideUp(speed)*** – швидкість появи передається як аргумент, при цьому можна вказати кількість мілісекунд або одне зі стандартних значень (slow, normal, fast).
3. ***element.slideUp(speed,function)*** – другим аргументом можна вказати ім'я функції, яка виконається при за-вершенні анімації.

.slideToggle()

slideToggle() – функція перемикає видимість елементів набору, використовуючи ефект зміни висоти елементів.



1. ***element.slideToggle()*** – швидкість відображення за замовчуванням.
2. ***element.slideToggle(speed)*** – швидкість відображення передається як аргумент, при цьому можна вказати кількість мілісекунд або одне зі стандартних значень (slow, normal, fast).
3. ***element.slideToggle(bool)*** – як аргумент передається значення перемикачів (*true* – відображає елементи, *false* – приховує елементи).
4. ***element.slideToggle(speed,function)*** – другим аргументом можна вказати ім'я функції, яка виконається при завершенні анімації.

.fadeIn()

.fadeIn() – функція робить видимими всі елементи набору, використовуючи зміну прозорості елементів.



1. ***element.fadeIn()*** – швидкість появи за замовчуванням.
2. ***element.fadeIn(speed)*** – швидкість появи передається як аргумент, при цьому можна вказати кількість мілісекунд або одне зі стандартних значень (slow, normal, fast).
3. ***element.fadeIn(speed,function)*** – другим аргументом можна вказати ім'я функції, яка виконається після завершення анімації.

.fadeOut()

.fadeOut() – функція робить невидимими всі елементи набору, після чого встановлює CSS властивість display в «none».



1. ***element.fadeOut()*** – швидкість зникнення за замовчуванням.
2. ***element.fadeOut(speed)*** – швидкість зникнення передається як аргумент, при цьому можна вказати кількість мілісекунд або одне зі стандартних значень (slow, normal, fast).
3. ***element.fadeOut(speed,function)*** – другим аргументом можна вказати ім'я функції, яка виконається після завершення анімації.

.fadeTo()

.fadeTo() – функція робить більш/менш видимими всі елементи набору, змінюючи прозорість елементів до величини, зазначеної в аргументі.



1. ***element.fadeTo()*** – швидкість зміни видимості за замовчуванням.
2. ***element.fadeTo(speed)*** – швидкість зміни видимості передається як аргумент, при цьому можна вказати

кількість мілісекунд або одне зі стандартних значень (slow, normal, fast).

3. ***element.fadeTo(speed,function)*** – другим аргументом можна вказати ім'я функції, яка виконається при завершенні зміни видимості.

Функція дозволяє робити елементи більш прозорими, та навпаки.

5.6.3 Функції для роботи з чергою

Табл. 5.12

Функції для роботи з чергою

Функція	Пояснення
.delay()	встановлює таймер затримки виконання чергових функцій-ефектів (наступних пунктів у черзі)
.queue()	повертає посилання на перший елемент у черзі (масив із функцій)
.clearQueue()	видаляє з черги всі елементи, які ще не були виконані
.deQueue()	виконує наступну функцію в черзі для відповідних елементів
.stop()	зупиняє анімацію, що працює в цей час

.delay()

.delay(мілісекунди) – функція встановлює таймер затримки виконання чергових функцій-ефектів (наступних елементів у черзі) для відповідних елементів набору jQuery на відповідну кількість мілісекунд.

```
element.fadeOut("slow").delay(2000).fadeIn("slow");
```



Встановиться затримка на 2 секунди

Рис. 5.31. Встановлення затримки

.queue()

.queue() – повертає перший елемент черги. Якщо передати у функцію параметр "fx", то ми отримаємо набір з усіх елементів черги.

```
var q = $("p").queue("fx");
```

.queue('fx',function) – функція, додає нову функцію, яка виконується в кінці черги для всіх елементів набору. Також можна передати цілий набір функцій.

```
var q = $("p").queue("fx", new_Func);
```

.clearQueue()

.clearQueue() – функція видаляє з черги всі елементи.

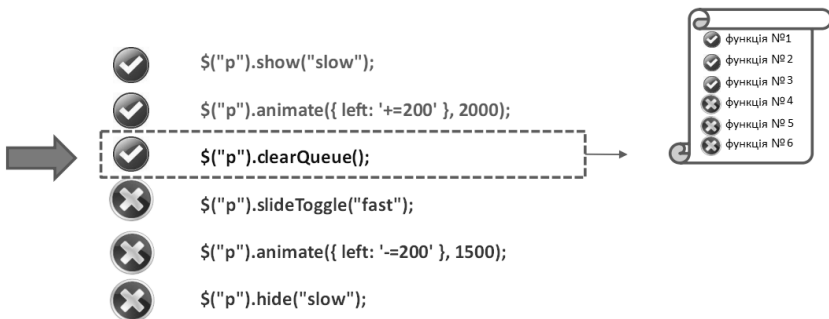


Рис. 5.32. Використання функції .clearQueue()

Якщо у визначеному місці (рис. 5.30) застосувати функцію, то всі наступні функції не будуть виконані, а будуть викреслені з черги.

.deQueue()

.deQueue() – функція виконує наступну функцію в черзі для відповідних елементів.

Функція перейде відразу на виконання наступної анімації.

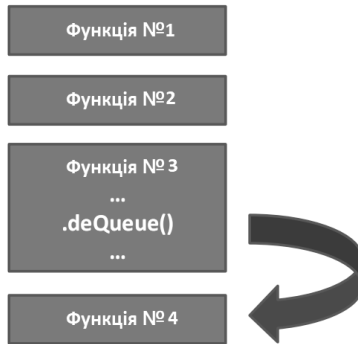


Рис. 5.33. Використання функції `.deQueue()`

`.stop()`

`.stop()` – функція зупиняє анімацію, що працює в цей час.



1. `.stop()` – зупиняє всі запущені анімаційні ефекти для всіх зазначених елементів.
2. `.stop(bool)` – залежно від аргумента видаляє чергу до кінця (*true*) чи ні (*false*).
3. `.stop(bool, bool)` – приймає другим аргументом значення, яке вказує закінчити поточну анімацію негайно (*true*) чи ні. Значення за замовчуванням – *false*.

5.6.4 Допоміжні функції jQuery

`.each()`

`$(E).each(array, function(i, n))` – виконує функцію для кожного елемента набору.

`function(i, n)` – callback-функція, яка виконується для кожного елемента набору. У callback-функції ключове слово *'this'* посилається на поточний елемент у наборі.

Аргументи callback-функції: *i* – індекс поточного елемента, *n* – значення поточного елемента.

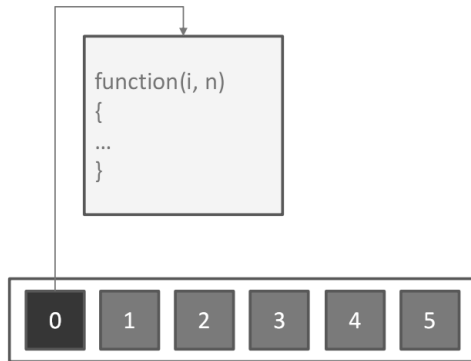


Рис. 5.34. Використання функції .each()

.grep()

\$.grep(array, callback, invert) – здійснює фільтрацію масиву.

Аргументи:

1. масив;
2. функція сортування;
3. прапор – *false* (за замовчуванням), *true* – інверсія фільтрації набору обраних елементів.

Сама функція-аргумент також може приймати аргументи: *i* – індекс поточного елемента, *n* – значення поточного елемента.

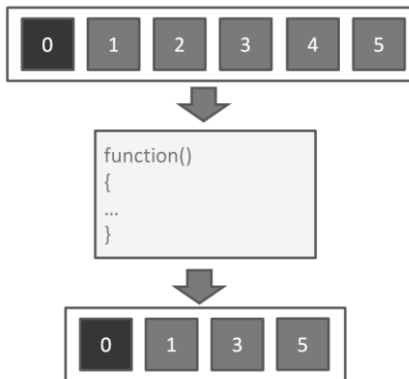


Рис. 5.35. Використання функції .grep()

.map()

\$.map(array, function) – здійснює перетворення масиву.

Приймає два аргументи:

1. масив;
2. Callback-функція для перетворення.

Аргументи функції перетворення: i – індекс поточного елемента, n – значення поточного елемента.

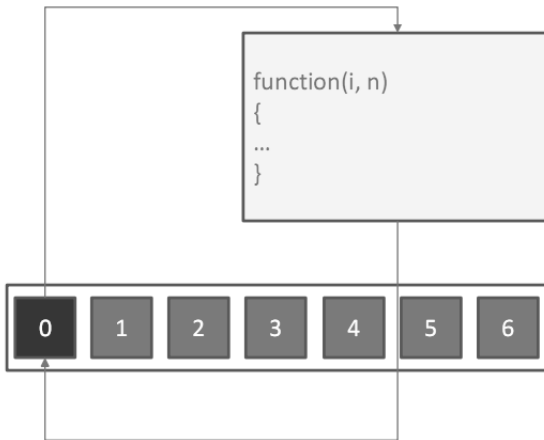


Рис. 5.36. Використання функції `.map()`

.inArray()

\$.inArray(value, array) – повертає індекс елемента, якщо він є в масиві або `-1` у випадку його відсутності. При цьому пошук проходить за значенням елемента.

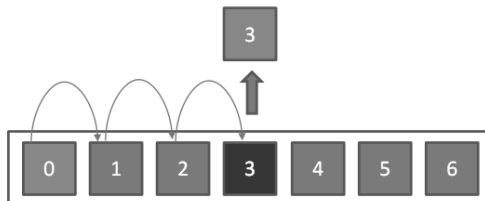


Рис. 5.37. Використання функції `.inArray()`

.makeArray()

\$.makeArray(obj) – перетворює об’єкт (наприклад, набір елементів jQuery), що має довжину та індекси, на масив, з яким можна працювати як з масивом.

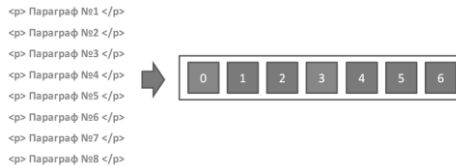


Рис. 5.38. Використання функції `.makeArray()`

.extend()

.extend() – здійснює злиття вмісту двох або більше об’єктів в один об’єкт, який передається першим параметром.

При збігу імен відбувається заміна вмісту властивості або методу.

Також може використовуватися для додавання функцій у простір імен jQuery.

```
$.extend(someObject, SomeOtherObject);
```

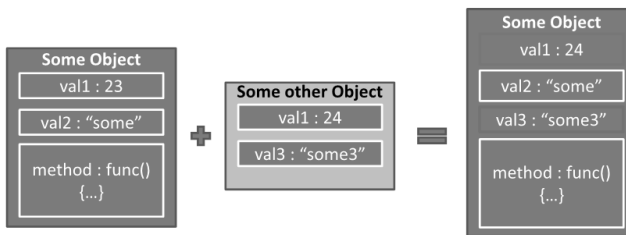


Рис. 5.39. Використання функції `.extend()`

.index()

.index() – повертає індекс елемента в наборі, якщо знаходить його. Індексация відбувається з нуля.

```
var index=$( "div" ).index( this );
```

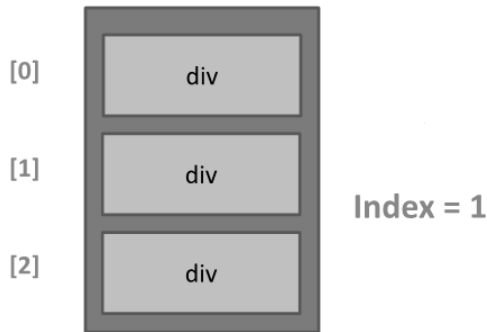


Рис. 5.40. Використання функції `.index()`

5.6.5 Метадані в JQuery

`.data()`

`.data()` – додає дані будь-якого типу до елементів DOM без ризику циклічної залежності, а також витоків пам'яті.

За допомогою функції `data()` можна виконувати зчитування і запис додаткової інформації до елементів DOM-дерева.

1. `element.data("param", "value")` – приєднує значення "value" під ім'ям "param";
2. `element.data()` – повертає всі прикріплені дані до елемента у вигляді об'єкта.
3. `element.data("key")` – дістає значення з даних під ім'ям "key".

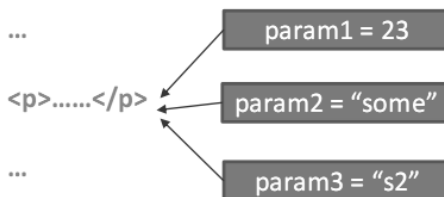


Рис. 5.41. Використання функції `.data()`

.data(object)

У якості аргумента **.data (object)** можна використовувати об'єкт `element.data("obj", {val1: 23, val2: "some"});`

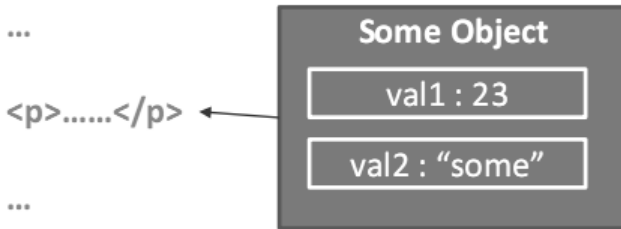


Рис. 5.42. Використання функції `.data(object)`

.removeData()

.removeData() – видаляє дані, додані за допомогою `data()`.

При виконанні функції **removeData()** без параметрів відбувається видалення всіх значень, доданих через **data()**.

.data() і **removeData()** не впливають на будь-які атрибути HTML5 `data` – у документі;

1. **element.removeData()** – видаляє всі прикріплені значення.
2. **element.removeData("key")** – видаляє значення за ключем.
3. **element.removeData("key1 key2 key3")** – видаляє значення за списком ключів.

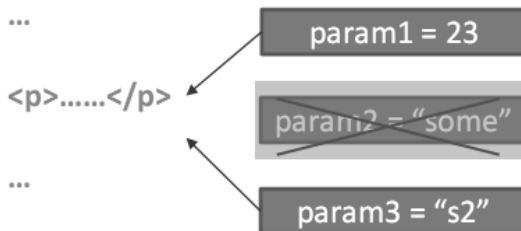


Рис. 5.43. Використання функції `.removeData()`

РОЗДІЛ 6

БІБЛІОТЕКА БУТСТРАП

6.1 Знайомство з Bootstrap

Bootstrap – вільний набір інструментів для створення сайтів і web-застосунків. Включає в себе HTML і CSS шаблони оформлення для типографіки, web-форм, кнопок, міток, блоків навігації та інших компонентів web-інтерфейсів, включаючи JavaScript розширення.

Bootstrap використовує найсучасніші напрацювання в області CSS і HTML, тому необхідно бути уважним при підтримці старих браузерів.

Bootstrap складається з таких файлів:

```
bootstrap/
├── css/
│   ├── bootstrap-grid.css
│   ├── bootstrap-grid.css.map
│   ├── bootstrap-grid.min.css
│   ├── bootstrap-grid.min.css.map
│   ├── bootstrap-reboot.css
│   ├── bootstrap-reboot.css.map
│   ├── bootstrap-reboot.min.css
│   ├── bootstrap-reboot.min.css.map
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   └── bootstrap.min.css.map
└── js/
    ├── bootstrap.bundle.js
    ├── bootstrap.bundle.js.map
    ├── bootstrap.bundle.min.js
    ├── bootstrap.bundle.min.js.map
    ├── bootstrap.js
    ├── bootstrap.js.map
    ├── bootstrap.min.js
    └── bootstrap.min.js.map
```

Рис. 6.1. Структура файлів Bootstrap

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Для використання додаткових компонентів Bootstrap необхідна бібліотека jQuery.

6.2 Колонкова верстка

Колонкова верстка в Bootstrap складається з рядків у 12 колонок по горизонталі. При використанні динамічних CSS-запитів, сітка і елементи адаптуються під різні екрани пристроїв.

Принцип створення макета:

- створення контейнера (*div* з класом "**container**");
- створення "рядка" (*div* з класом "**row**") у *div* з класом "**container**";
- створення колонок у "рядку".

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# of columns	12				
Gutter width	30px (15px on each side of a column)				

Рис. 6.2. Принцип колонкової верстки в Bootstrap 4

* кількість колонок у "рядку" (максимум 12).

Клас "container" робить блок фіксованої ширини:

- при ширині вікна браузера від 798px до 991px – **750px**;
- при ширині вікна браузера від 992px до 1199px – **970px**;
- при ширині вікна браузера вище 1200px – **1170px**.

Клас "**container-fluid**" робить блок розтягнутим по всій доступній ширині.

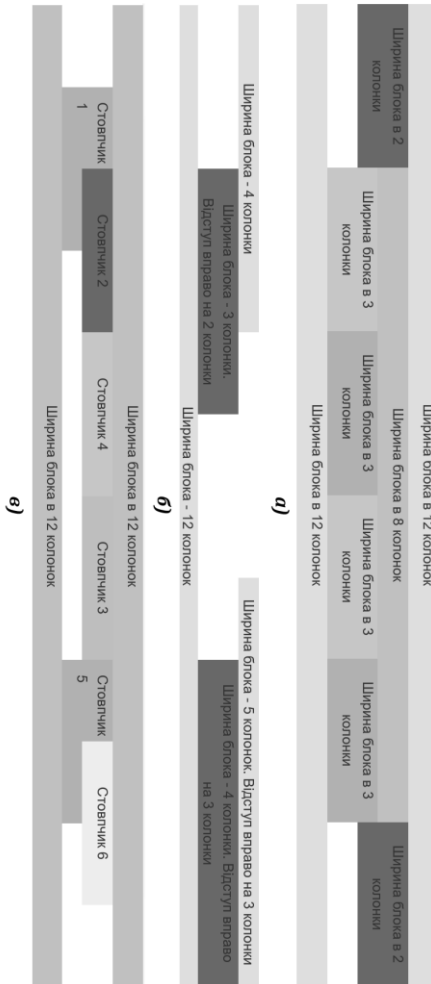


Рис. 6.3. Колонкова верстка:

- а – додавання нових стовпчиків у вже існуючий;
- б – переміщення колонок;
- в – зміна порядку слідування стовпчиків

ПРАВИЛО! Якщо кількість колонок в одному рядку становить більше 12, то колонки, що перевищують це число, будуть перенесені на нові рядки.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Для того, щоб додати нові стовпці у вже існуючий стовпчик (для статичного макета), слід у потрібний стовпчик додати *div* з класом **"row"** і в нього помістити потрібну кількість стовпчиків (12 максимум) (рис. 6.3.а).

Для переміщення колонок вправо використовуються класи **.offset-lg-*** (рис. 6.3.б). Ці класи збільшують відступ зліва на * стовпців. Наприклад, *.offset-lg-4* переміщує *.col-lg-4* на 4 стовпці вправо.

Для того, щоб змінити порядок слідування стовпчиків, досить додати спеціальний клас: **order-*** (рис. 6.3.в).

*порядковий номер колонки

6.3 Bootstrap CSS

Headings

```
<h1>h1. Bootstrap heading</h1>      <!-- Semibold 36px -->
<h2>h2. Bootstrap heading</h2>      <!-- Semibold 30px -->
<h3>h3. Bootstrap heading</h3>      <!-- Semibold 24px -->
<h4>h4. Bootstrap heading</h4>      <!-- Semibold 18px -->
<h5>h5. Bootstrap heading</h5>      <!-- Semibold 14px -->
<h6>h6. Bootstrap heading</h6>      <!-- Semibold 12px -->
```



Рис. 6.3. Форматування заголовків

Для того, щоб виділити текст параграфа, використовується клас **"lead"**.

Виділений текст: тег **"mark"** або клас **".mark"**.

Класи вирівнювання: *text-left*, *text-center*, *text-right*, *text-justify*, *text-nowrap*.

Класи для трансформації тексту: *text-lowercase*, *text-uppercase*, *text-capitalize*.

Абревіатури:

```
<abbr title="Hyper Text Markup Language"> HTML </abbr>
```

Список без стилів: клас *"list-unstyled"*. Горизонтальний список: клас *"list-inline"*.

Створення звичайної цитати здійснюється за допомогою тегу **<blockquote>**. За замовчуванням з лівого боку буде вертикальна смуга.

Додавання футера в цитату для визначення джерела. Джерело прийнято поміщати в тег **<cite>**.

Клас *"blockquote text-right"* переміщує цитату вправо.

Приклад коду:

```
<blockquote class = "blockquote text-right">  
<p> Lorem ipsum dolor sit amet, consectetur adipis-cing  
elit. Integer posuere erat a ante. </p>  
<footer class = "blockquote-footer"> Someone famous in  
<cite title = "Source Title"> Source Title </cite>  
</footer>  
</blockquote>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.

Someone famous in Source Title —

Тег **"code"** використовується для додавання фрагментів коду, що відображаються на web-сторінці.

Приклад коду:

```
<p> В тезі <code>&lt;head&gt;</code> розташована службова  
інформація. </p>
```

В тезі **<head>** розташована службова інформація.

Тег **"kbd"** використовується для позначення тексту, що набирається на клавіатурі.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Приклад коду:

```
<p> Для запуску застосунку натисніть комбінацію клавіш  
<kbd> ctrl + e </kbd></p>
```

Для запуску застосунку натисніть комбінацію клавіш **ctrl + e**

Оформлення таблиць

Робота з Bootstrap полягає у використанні потрібних класів для досягнення необхідного зовнішнього вигляду елементів сторінки. У таблицях клас «**table**» для тегу `<table>` задає базовий вид.

Приклад коду:

```
<table class="table">  
  <tr>  
    <th>Product</th>  
    <th>Price</th>  
  </tr>  
  <tr>  
    <td>Item 1</td>  
    <td>800</td>  
  </tr>  
  <tr>  
    <td>Item 2</td>  
    <td>1200</td>  
  </tr>  
  <tr>  
    <td>Item 3</td>  
    <td>1140</td>  
  </tr>  
</table>
```

Product	Price
Item 1	800
Item 2	1200
Item 3	1140

Рис. 6.4. Оформлення таблиць

Також можна інвертувати кольори – з легким текстом на темних фонах – з «*table-dark*».

Клас «*table-striped*» для тегу `<table>` виділяє непарні рядки кольором.

Клас «*table-bordered*» для тегу `<table>` задає рамку таблиці.

Приклад коду:

```
<table class="table table-striped table-bordered">
  <tr>
    <th>Product</th>
    <th>Price</th>
  </tr>
  <tr>
    <td>Item 1</td>
    <td>800</td>
  </tr>
  <tr>
    <td>Item 2</td>
    <td>1200</td>
  </tr>
  <tr>
    <td>Item 3</td>
    <td>1140</td>
  </tr>
</table>
```

Product	Price
Item 1	800
Item 2	1200
Item 3	1140

Рис. 6.5. Рамка та виділення непарних рядків у таблицях
Клас «*table-hover*» – вмикає стан наведення на рядки таблиці в межах `<tbody>`.

Клас «*table-sm*» вдвічі зменшує відступи всередині таблиці.

Dropdowns



Рис. 6.6. Приклад *Dropdowns*

Основні складові:

- оболонка (`div class = "dropdown"`);
- кнопка (`<button>`);
- гіперпосилання (`<a>`).

Приклад коду:

```
<div class="btn-group">
  <button type="button" class="btn btn-primary dropdown-
toggle" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
    Action
  </button>
  <div class="dropdown-menu">
    <a class="dropdown-item" href="#">Action</a>
    <a class="dropdown-item" href="#">Another action</a>
    <a class="dropdown-item" href="#">Something else
here</a>
    <div class="dropdown-divider"></div>
    <a class="dropdown-item" href="#">Separated link</a>
  </div>
</div>
```

Button groups



Рис. 6.7. Приклад *Button group*

Основні складові:

- оболонка (*div class = "btn-group"*);
- набір кнопок (*<button>*).

Приклад коду:

```
<div class="btn-group" role="group" aria-label="Basic
example">
  <button type="button" class="btn btn-secondary">
Left</button>
  <button type="button" class="btn btn-secondary">
Middle</button>
  <button type="button" class="btn btn-secondary">
Right</button>
</div>
```

Input groups



@ Username

Рис. 6.8. Приклад *input-group*

Основні складові:

- оболонка (*div class = "input-group"*);
- контейнер для іконки (**);
- елемент форми (*<input type = "text">*).

Приклад коду:

```
<div class="input-group mb-3">
  <div class="input-group-prepend">
    <span class="input-group-text" id="basic-
addon1">@</span>
  </div>
  <input type="text" class="form-control"
placeholder="Username" aria-label="Username" aria-
describedby="basic-addon1">
</div>
```

Navbars



Navbar Home Link Dropdown ▾ Disabled Search Search

Рис. 6.9. Приклад *navbar*

Основні складові:

- оболонка (*nav class = "navbar"*);
- набір елементів.

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Приклад коду:

```
<nav class="navbar navbar-expand-lg navbar-light bg-
light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarSupportedContent"
aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
  <span class="navbar-toggler-icon"></span>
</button>

  <div class="collapse navbar-collapse"
id="navbarSupportedContent">
  <ul class="navbar-nav mr-auto">
    <li class="nav-item active">
      <a class="nav-link" href="#">Home <span class="sr-
only">(current)</span></a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Link</a>
    </li>
    <li class="nav-item dropdown">
      <a class="nav-link dropdown-toggle" href="#"
id="navbarDropdown" role="button" data-toggle="dropdown"
aria-haspopup="true" aria-expanded="false">
        Dropdown
      </a>
      <div class="dropdown-menu" aria-
labelledby="navbarDropdown">
        <a class="dropdown-item" href="#">Action</a>
        <a class="dropdown-item" href="#">Another
action</a>
        <div class="dropdown-divider"></div>
        <a class="dropdown-item" href="#">Something else
here</a>
      </div>
    </li>
    <li class="nav-item">
      <a class="nav-link disabled" href="#">Disabled</a>
    </li>
  </ul>
  <form class="form-inline my-2 my-lg-0">
    <input class="form-control mr-sm-2" type="search"
placeholder="Search" aria-label="Search">
```

```
<button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Search</button>
</form>
</div>
</nav>
```

Breadcrumbs

Home / Library / Data

Рис. 6.10. Приклад *Dropdowns*

Основні складові:

- оболонка (*ol class = "breadcrumb"*);
- елементи списку (**).

Приклад коду:

```
<nav aria-label="breadcrumb">
<ol class="breadcrumb">
<li class="breadcrumb-item"><a href="#">Home</a></li>
<li class="breadcrumb-item"><a href="#">Library</a>
</li>
<li class="breadcrumb-item active" aria-current=
"page">Data</li>
</ol>
</nav>
```

Pagination

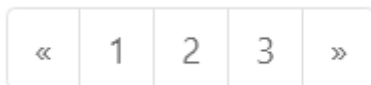


Рис. 6.11. Приклад *pagination*

Основні складові:

- оболонка (*ul class = "pagination"*);
- елементи списку (**).

Приклад коду:

```
<nav aria-label="Page navigation example">
<ul class="pagination">
<li class="page-item">
<a class="page-link" href="#" aria-label="Previous">
<span aria-hidden="true">&laquo;</span>
<span class="sr-only">Previous</span>
</a>
```


Практичний web-дизайн: проектування, створення та супроводження web-вузла

```
</li>
<li class="page-item"><a class="page-link" href="#">
1</a></li>
<li class="page-item"><a class="page-link" href="#">
2</a></li>
<li class="page-item"><a class="page-link" href="#">
3</a></li>
<li class="page-item">
  <a class="page-link" href="#" aria-label="Next">
    <span aria-hidden="true">&raquo;</span>
    <span class="sr-only">Next</span>
  </a>
</li>
</ul>
</nav>
```

Progress bars



Рис. 6.12. Приклад *progress*

Основні складові:

- оболонка (*div class = "progress"*);
- смуга прогресу (*div class = "progress-bar"*).

Приклад коду:

```
<div class="progress">
  <div class="progress-bar" role="progressbar" aria-
  valuenow="60" aria-valuemin="0" aria-valuemax="100"
  style="width: 60%;">
    60%
  </div>
</div>
```

List groups

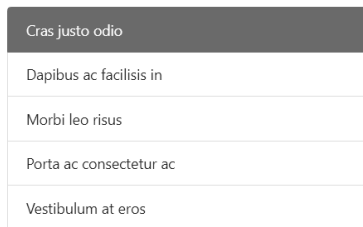


Рис. 6.13. Приклад *list-group*

Основні складові:

- оболонка (*ul class = "list-group"*);
- елементи списку (**).

Приклад коду:

```
<div class="list-group">
  <a href="#" class="list-group-item active">Cras
justo odio</a>
  <a href="#" class="list-group-item">Dapibus ac
facilisis in</a>
  <a href="#" class="list-group-item">Morbi leo
risus</a>
  <a href="#" class="list-group-item">Porta ac
consectetur </a>
  <a href="#" class="list-group-item">Vestibulum at
eros</a>
</div>
```

Panels



Рис. 6.14. Приклад *panel*

Основні складові:

- оболонка (*div class = "panel"*);
- заголовок панелі (*div class = "panel-heading"*);
- тіло панелі (*div class = "panel-body"*);
- футер панелі (*div class = "panel-footer"*).

Приклад коду:

```
<div class="panel panel-default">
  <div class="panel-heading">
    <h3 class="panel-title">Panel title</h3>
  </div>
  <div class="panel-body">
    Panel content
  </div>
</div>
```

Tabs with dropdowns



Рис. 6.15. Приклад *nav-tabs*

Основні складові:

- контейнер з закладками (*ul class = "nav-tabs"*);
- елементи списку (**).

Приклад коду:

```
<ul class="nav nav-tabs">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" data-
toggle="dropdown" href="#" role="button" aria-
haspopup="true" aria-expanded="false">Dropdown</a>
    <div class="dropdown-menu">
      <a class="dropdown-item" href="#">Action</a>
      <a class="dropdown-item" href="#">Another action</a>
      <a class="dropdown-item" href="#">Something else
here</a>
      <div class="dropdown-divider"></div>
      <a class="dropdown-item" href="#">Separated link</a>
    </div>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
  <li class="nav-item">
    <a class="nav-link disabled" href="#">Disabled</a>
  </li>
</ul>
```

Tooltips



Рис. 6.16. Приклад *tooltip*

Основні складові:

- атрибут *"title"* з контентом підказки;
- набір призначених для користувача атрибутів для JS.

Приклад коду:

```
<button type="button" class="btn btn-secondary" data-  
toggle="tooltip" data-placement="bottom" title="Tooltip on  
bottom">  
  Tooltip on bottom  
</button>  
  
<script>  
  $(function () {  
    $('[data-toggle="tooltip"]').tooltip()  
  })  
</script>
```

Carousel



Рис. 6.17. Приклад *carousel*

Основні складові:

- оболонка (*div class = "carousel slide"*);
- контейнер з іконками перемикачів слайдів (*ol class = "carousel-indicators"*);
- контейнер для слайдів (*div class = "carousel-inner"*);
- контейнер для окремого слайда (*div class = "item"*);
- основні перемикачі слайдів (* ... *).

Практичний web-дизайн: проектування, створення та супроводження web-вузла

Приклад коду:

```
<div id="carouselExampleIndicators" class="carousel slide" data-ride="carousel">
  <!-- Контейнер для іконок кожного слайда -->
  <ol class="carousel-indicators">
    <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>
    <li data-target="#carouselExampleIndicators" data-slide-to="1"></li>
    <li data-target="#carouselExampleIndicators" data-slide-to="2"></li>
  </ol>
  <!-- Контейнер для слайдів -->
  <div class="carousel-inner">
    <div class="carousel-item active">
      
    </div>
    <div class="carousel-item">
      
    </div>
    <div class="carousel-item">
      
    </div>
  </div>
  <!-- Контейнер для перемикачів між слайдами -->
  <a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

Більше компонентів Bootstrap можна знайти на офіційному сайті <https://getbootstrap.com/>.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Офіційний сайт Adobe [Електронний ресурс].
URL: <https://www.adobe.com/ua/products/photoshop.html>.
2. Уроки Photoshop [Електронний ресурс].
URL: <https://www.site-do.ru/>.
3. Фотошоп мастер [Електронний ресурс].
URL: <https://photoshop-master.ru/>.
4. Мельник Р. А. Програмування інтернет-застосувань : навч. посіб. / Р. А. Мельник, Р. Б. Тушницький. – Львів : Видавництво Львівської політехніки, 2013. – 256 с.
5. Основы HTML 5 [Електронний ресурс].
URL: <https://myrusakov.ru/html-5.html>.
6. Учебник CSS [Електронний ресурс].
URL: <https://ru.html.net/tutorials/css>.
7. Учебник CSS для начинающих [Електронний ресурс].
URL: <https://www.webremeslo.ru/css>.
8. Самоучитель HTML [Електронний ресурс].
URL: <https://htmlbook.ru>.
9. Современный учебник Javascript [Електронний ресурс].
URL: <https://learn.javascript.ru>.
10. Online Web Tutorials [Електронний ресурс].
URL: <https://www.w3schools.com/>.
11. Офіційний сайт jQuery [Електронний ресурс].
URL: <https://jquery.com/>.
12. Офіційний сайт Bootstrap [Електронний ресурс].
URL: <https://getbootstrap.com/>.

ДЛЯ НОТАТОК

Навчальне видання

Альона Володимирівна

Швед

Євген Олександрович

Давиденко

**ПРАКТИЧНИЙ WEB-ДИЗАЙН:
ПРОЕКТУВАННЯ, СТВОРЕННЯ
ТА СУПРОВОДЖЕННЯ WEB-ВУЗЛА**

Навчальний посібник

Редактор *Я. Котенко*.

Технічний редактор, комп'ютерна верстка *Д. Кардаш*.

Дизайн обкладинки *Д. Кардаш*.

Друк *С. Волинець*. Фальцювальньо-палітурні роботи *О. Кутова*.

Підп. до друку 22.01.2018

Формат 60x84¹/₁₆. Папір офсет.

Гарнітура "Times New Roman". Друк ризограф.

Ум. друк. арк. 11,16. Обл.-вид. арк. 4,19.

Тираж 300 пр. Зам. № 5614.

Видавець і виготовлювач: ЧНУ ім. Петра Могили.

54003, м. Миколаїв, вул. 68 Десанників, 10.

Тел.: 8 (0512) 50-03-32, 8 (0512) 76-55-81, e-mail: rector@chmnu.edu.ua.

Свідоцтво суб'єкта видавничої справи ДК № 6124 від 05.04.2018.