МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Одеська юридична академія»

Кафедра інформаційних технологій

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з дисципліни
«WEB-технології та програмування»

для підготовки здобувачів вищої освіти
галузі знань 12 «Інформаційні технології»

Одеса 2020

УДК 004.43

Автори:
Орлов С. В. – старший викладач кафедри інформаційних технологій Національного університету «Одеська юридична академія»
Толокнов А. А. – асистент кафедри інформаційних технологій Національного університету «Одеська юридична академія»

Рекомендовано навчально-методичною радою Національного університету «Одеська юридична академія», протокол № _2_ від __4 грудня__ 2020 р.

Рецензенти:
Яценко В.О. – керівник проекту Освітній Фонд KeepSolid;
Бойко Д.В. – доцент, кандидат технічних наук, доцент кафедри кібербезпеки Національного університету «Одеська юридична академія»

Розглянуто основи створення веб-сторінок засобами мови розмітки гіпертекстових документів HTML та таблиць стилів CSS. Для надання інтерактивності веб-сторінкам розглядається мова програмування JavaScript.
Призначено для здобувачів вищої освіти галузі знань 12 «Інформаційні технології», які вивчають дисципліни «Веб-технології та веб-дизайн» та «Веб-програмування».

Методичні вказівки до виконання лабораторних робіт з дисципліни «Web-технології та програмування» для здобувачів вищої освіти галузі знань 12 «Інформаційні технології» / С. В. Орлов, А. А. Толокнов – Одеса: НУ «ОЮА», 2020. – 50 с.

Зміст

Вступ

У сучасному світі все більшу роль відіграють інформаційні мережі. Вони використовуються для обміну інформацією, рекламної діяльності. Виникають елементи управління в електронному вигляді: е-деканати, е-школи, е-пенсійні фонди тощо. Інтернет речей (IoT) передбачає ще більшу інтеграцію мереж в повсякденне життя. Тому вивчення web-технологій і web-програмування стає нагальною потребою.

Web-технології – технології створення і підтримки інформаційних ресурсів в Internet, таких як сайти, блоги, чати, електронні бібліотеки та ін. Реалізація цих завдань передбачає використання web-програмування.

Web-програмування – область програмування орієнтована на проектування, реалізацію і подальший супровід додатків для Internet простору, WEB.

Мови програмування працюють як на клієнтському боці (браузер), так і на боці сервера (web-сервер). Даний курс орієнтований на вивчення HTML, CSS, JavaScript.

Internet, мережева модель OSI. Стек протоколів TCP/IP.

Internet – система об'єднаних комп'ютерних мереж для зберігання, обробки і передачі інформації. Згадується як Всесвітня мережа і Глобальна мережа, а також просто Мережа. Побудована на базі стека протоколів TCP/IP. На основі Інтернету працює Всесвітня павутина (World Wide Web, WWW) та інші систем передачі даних.

Прототипом мережі Internet була **ARPA NET** (англ. *Advanced Research Projects Agency Network* – Агентство передових дослідницьких мережевих проектів) – комп'ютерна мережа, створена у 1969 році в США Агентством Міністерства оборони по перспективним дослідженням.

Стек протоколів TCP/IP був створений на основі Network Control Protocol групою розробників у 1972 році. У січні 1983 року вона стала першою в світі мережею, яка перейшла на маршрутизацію пакетів даних.

Великий внесок у розвиток стека TCP/IP вніс університет Берклі, реалізувавши протоколи стека у своїй версії ОС UNIX BSD 4.2 в 1983 році.

Мережева модель **OSI** (англ. *Open systems interconnection basic reference model* – Базова Еталонна Модель Взаємодії Відкритих Систем (ЕМВ ВС)) – мережева модель стека мережевих протоколів.

На фізичному рівні по мережі пересилаються біти. На цьому рівні вирішуються проблеми з шумами і згасанням сигналу. На канальному рівні це *кадр* або *фрейм*. Кадри доповнюються контрольної інформацією, що дозволяє відновлювати їх у разі помилок в мережі. У протоколі IP – пакет, а в протоколі

TCP – *сегмент (частина пакета)*. Ми будемо дотримуватися універсального терміну "пакет"

UDP (*User Datagram Protocol* – протокол призначених для користувача датаграм) – один з ключових елементів набору мережевих протоколів для Internet.

Datagram (датаграма) – блок інформації, що передається протоколом через мережу зв'язку без попереднього встановлення з'єднання і створення віртуального каналу.

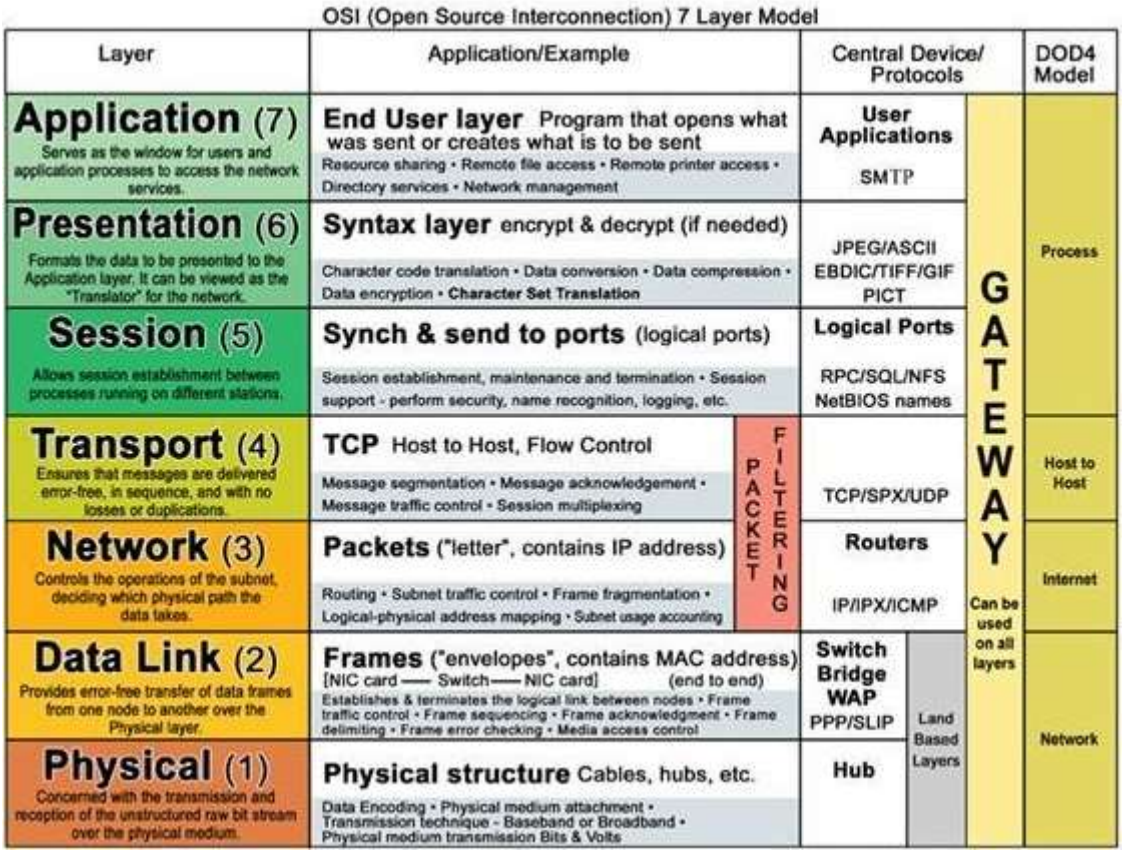


Рис. 1. OSI – Модель взаємодії відкритих систем

Коли пакет передається вниз по стеку протоколів, готуючись до відправки, кожен протокол додає в нього свій власний заголовок. Закінчений пакет одного протоколу стає корисним вмістом пакета, що генерується таким протоколом. Ця операція відома як інкапсуляція. На приймаючій машині інкапсульовані кадри відновлюються в зворотному порядку.

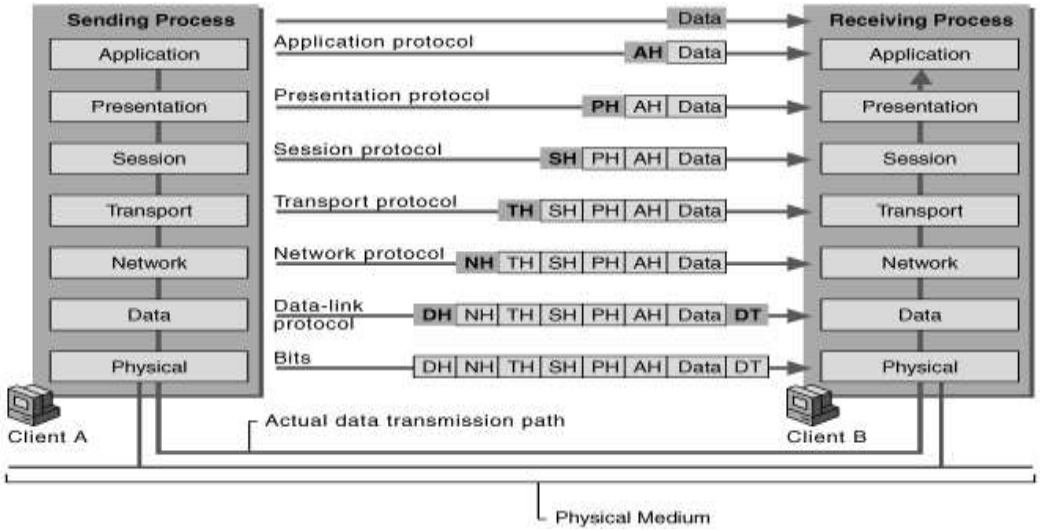


Рис. 2. Передача інформації за допомогою стека протоколів OSI.

Застосунки, що функціонують на основі стека TCP/IP, деякі рівні OSI використовують одночасно. У моделі OSI це виглядає наступним чином.

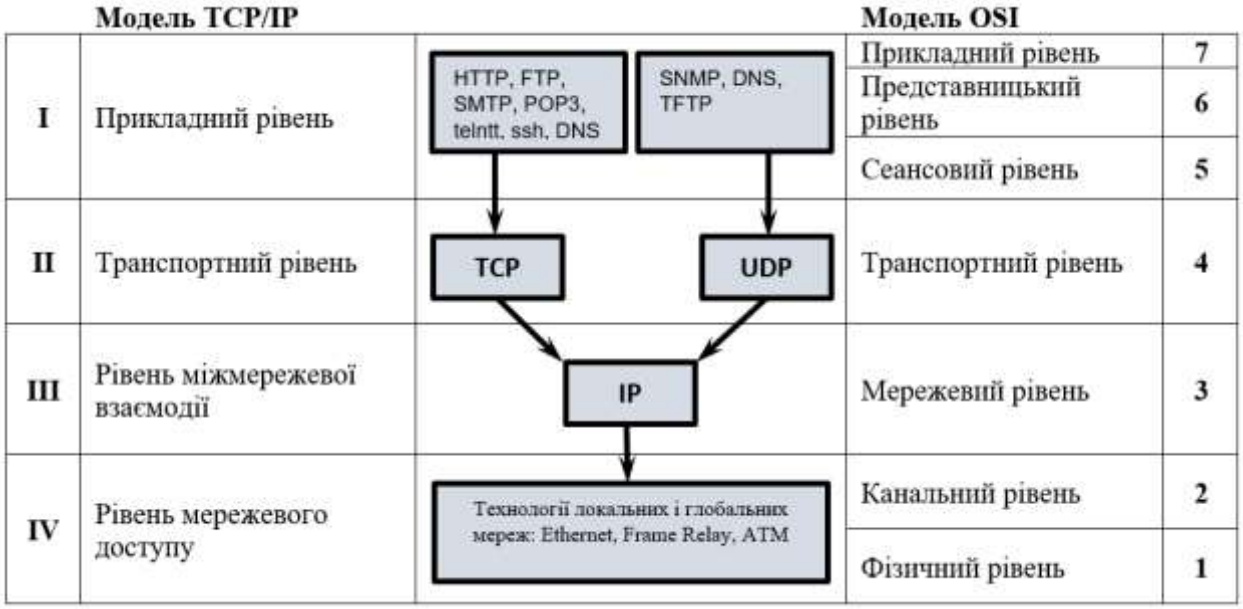


Рис. 3 Стек протоколів TCP/IP.

Для організації взаємозв'язку окремих вузлів в глобальну мережу Internet потрібні спеціальні пристрої.

Hub (*концентратор*) – повторює на всіх вихідних портах вхідну інформацію. Працює на фізичному рівні. Використовується при організації LAN.

Bridge (*міст*) – працює на *канальному* рівні. Використовує MAC-адресу (фізичний унікальний ідентифікатор пристрою) і внутрішню таблицю відповідності MAC-адрес. Кадри пересилаються адресно. Деякі мости можуть працювати на мережевому рівні

Switch (*перемикач, комутатор*) – пристрій, призначений для з'єднання декількох вузлів комп'ютерної мережі в межах одного або декількох сегментів мережі. Комутатор працює в основному на *канальному* рівні моделі OSI. Комутатори були розроблені з використанням мостових технологій і часто розглядаються як багатопортові мости

Router (*маршрутизатор*) – спеціалізований комп'ютер, який пересилає пакети між різними сегментами мережі на основі правил і таблиць маршрутизації. Маршрутизатори працюють на *мережному* рівні OSI.

Gateway (*мережевий шлюз*) – апаратний або програмний маршрутизатор призначений для сполучення комп'ютерних мереж (наприклад, локальної і глобальної), що використовують різні протоколи

Для адресації окремого вузла мережі може використовуватися IPv4 адреса – Internet Protocol версії 4. Зазвичай цю адресу просто позначають як IP. Вона використовується в більшій частині Internet. У цій версії IP-адреса являє собою 32-бітове число, яке зазвичай складається із чотирьох десяткових чисел значенням від 0 до 255, розділених крапками, наприклад, 192.168.0.8.

У розширеному вигляді, IP-адреса складається з двох частин: номера мережі і номера вузла. Наприклад, IP-адреса виду «192.168.7.0/24» замінює собою вказівку діапазону IP-адрес. 24 – означає кількість одиничних розрядів в масці підмережі.

Тобто 192.168.7.0/24 означає діапазон адрес хостів від 192.168.7.1 до 192.168.7.254.

192.168.7.0 – адреса мережі

192.168.7.255 – широкомовна адреса мережі.

Адреси IP, використовувані в локальних мережах, відносять до приватних.

Адреси Internet, які використовуються для організації внутрішніх приватних мереж, Intranet:

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16
- 169.254.0.0/16
- 192.0.0.0/24
- 198.51.100.0/24
- 240.0.0.0/4

Адреси для внутрішнього використання в *Internet*:

- 127.0.0.0/8 - використовується для комунікацій усередині хоста з ім'ям *localhost*.

- блок з 169.254.1.0 по 169.254.254.255 (підмережа 169.254.0.0/16 за винятком підмереж 169.254.0.0/24 і 169.254.255.0/24) – використовується для автоматичної настройки мережевого інтерфейсу в разі відсутності DHCP

Кількість адрес IPv4 в глобальній мережі дозволяє трохи більше 4 мільярдів. Інтернет речей, наприклад, передбачає інтегрування все більшої кількості елементів в мережу. Це і камери спостереження, всілякі датчики та інші пристрої. Тому адрес IPv4 може в майбутньому не вистачити.

IPv6 – нова версія інтернет-протоколу (IP) 1996 року. Довжина її адреси 128 біт замість 32.

Цей протокол може забезпечити до $5 \cdot 10^{28}$ адрес. Частка IPv6 адресації, на початку 2020 року, в мережевому трафіку становить близько 30%

Приклад звернення до хосту за допомогою протоколу IPv6:

http://[2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d]:8080/

Спеціальний запис вбудованого IPv4 на IPv6. Нехай IPv4=192.0.2.1, тоді адреса в форматі IPv6 буде виглядати так:

::ffff:192.0.2.1

Незначущі старші нулі в групах можуть бути опущені. Наприклад, 2001:0db8:0000:0000:0000:0000:ae21:ad12 може бути скорочений до 2001:db8::ae21:ad12

Web-сервера. HTTP протокол.

Для роботи зі сторінками сайтів в Internet виникла необхідність в протоколі обміну документами.

HTTP (*HyperText Transfer Protocol* – протокол передачі гіпертексту 1992) – протокол прикладного рівня передачі даних спочатку у вигляді гіпертекстових документів в форматі «HTML», зараз використовується для передачі довільних даних. Основою HTTP є технологія «клієнт-сервер», тобто передбачається існування:

- Споживачів-клієнтів (*браузери*: Firefox, Google Chrome, Opera, ...), які ініціюють з'єднання і надсилають запит;
- Постачальників-серверів (*web-сервери: apache, nginx, Resin, ...*), які очікують з'єднання для отримання запиту, виробляють необхідні дії і повертають назад повідомлення з результатом.

HTTP – протокол прикладного рівня; аналогічними йому є FTP і SMTP. Обмін повідомленнями йде по звичайній схемі «запит-відповідь».

Proxy (проксі-сервер – «представник», «уповноважений»), сервер-посередник – проміжний сервер (комплекс програм) в комп'ютерних мережах, що виконує роль посередника між користувачем і цільовим сервером (при цьому про посередництво можуть знати, так і не знати обидві сторони), що дозволяє клієнтам як виконувати непрямі запити (приймаючи і передаючи їх через проксі-сервер) до інших мережних служб, так і отримувати відповіді.

Спочатку клієнт підключається до проксі-сервера і запитує який-небудь ресурс (наприклад e-mail), розташований на іншому сервері. Потім проксі-сервер або підключається до вказаного серверу і отримує ресурс у нього, або повертає ресурс із власного кешу (у випадках, якщо проксі має свій кеш). У деяких випадках запит клієнта або відповідь сервера може бути змінений проксі-сервером в певних цілях.

Проксі-сервер дозволяє захищати комп'ютер клієнта від деяких мережевих атак і допомагає зберігати анонімність клієнта, але також може використовуватися шахраями для приховування адреси сайту, викритого в шахрайстві, зміни вмісту цільового сайту (підміна), а також перехоплення запитів самого користувача.

Тема 1. Основи HTML.

Базові елементи мови

Web-сторінка – це текстовий файл з розширенням *htm* або *html*. Він може бути створений в будь-якому текстовому редакторі, наприклад, Notepad ++. Це один з редакторів тексту чутливий до синтаксису HTML. Тобто, якщо документ має розширення *html/htm*, то синтаксичні помилки будуть підсвічуватися і видно відразу, як тільки ви їх зробите. Це особливо актуально в разі найпростіших описок і невідповідності відкриваючих і закриваючих тегів для документів великого розміру.

Web-документ, що відображається браузером, – це результат відповіді web-сервера на запит браузера-клієнта, можливо доопрацьований на клієнтській стороні. При цьому можливо буде потрібно виконання програм, створених на різних мовах, виконуваних як на серверній, так і на стороні клієнта. Для опису структури і загального зовнішнього вигляду використовується мова розмітки HTML, для опису зовнішнього вигляду – мова стилів CSS. Для опису поведінки документа, його реакції на дії користувача, використовується мова сценаріїв Java Script. Потім цей файл буде відкриватися будь-якою програмою браузером (IE, Opera, ...).

Теги

Все що обмежено кутовими дужками < і > називається тегами. Теги є командами мови HTML і при перегляді сторінки браузером у вікні не відображаються. Залежно від тегів текст сторінки може відображатися по різному.

Приклади тегів:

```
1   <html>
2   </body>
3   <br />
```

Теги бувають:

- початковими (відкриваючими) (1),
- кінцевими (закриваючими) (2 – зі зворотним слешем "/" після <)
- пустими[1] (3 – з пробілом і зворотним слешем "/" перед >).

Елементи

Елементом називається відкриваючий і закриваючий теги, з однаковим ім'ям і тим, що між ними міститься. Деякі елементи – такі, як `meta`, `br`, `hr`, `img`

[1] Такий запис притаманний для XHTML, згідно якому всі теги, навіть пусті, повинні бути закритими. В HTML4 і HTML5 пусті теги можна писати без слеша. Наприклад:
, <hr>, <img>.

нічого не містять. Елементи називають контейнерами, якщо між відкриваючий і закриваючим тегами міститься деяка інформація. Більшість елементів – контейнерні.

Приклади елементів:

```
1. <h1>Заголовок</h1>
2. <p>Будь-який текст</p>
3. <br />
```

Вкладення елементів

Елементи можуть міститися всередині один одного, але вони не повинні перекриватися. Наприклад:

```
<body>
    <p>
        Текст1
        Текст2
    </p>
</body>
```

Тобто, в елемент `<body>` вкладені два тексти і один елемент `<p>`. В HTML неможливо перетин елементів. Тег, відкритий першим, закривається останнім, і навпаки.

Атрибути елементів

Елементи можуть містити будь-яку кількість атрибутів. Атрибути є як би додатковими настройками. Вони впливають на поведінку елемента завдяки тому, що можуть набувати різних значень. Для деяких елементів існують обов'язкові атрибути, без яких елемент не буде працювати.

Атрибути завжди записуються в відкриваючому тегу у вигляді пар «`ім'я атрибута="значення атрибута"`». Значення атрибута потрібно брати або в подвійні лапки «`"`» або в одинарні «`'`». Атрибути поділяються пробілами або порожніми рядками. У закриваючому тегу атрибути не пишуть.

Наприклад:

```
<a href="page.html" class="anchor" title="anchor">
```

Значення атрибута може бути пустим. Наприклад:

```
<img alt="">
```

Атрибут, також, може не мати значення. Наприклад, атрибут `checked`:

```
<input type="checkbox" checked>
```

Коментарі

Коментарі полегшують розуміння документа. Текст коментаря на web-сторінку не виводиться. Коментар починається з послідовності `<!--` і закінчуються послідовністю `-->`.

Приклад коментаря:

```
<!-- Текст коментаря -->
```

Загальний вигляд гіпертекстової сторінки

```
<!DOCTYPE html>
<html lang="uk">
  <head>
    <meta charset="utf-8">
    <title>Заголовок веб-сторінки</title>
  </head>
  <body>
    Текст на веб-сторінці
  </body>
</html>
```

Завдання 1

1. Скопіювати в свою папку файл свого варіанту (*1_1.txt* для першого варіанту або файл *1_2.txt* для другого варіанту, *1_3.txt*, *1_4.txt*, *1_5.txt*, *1_6.txt*).
2. Зробити розмітку цього фала:
   - У головній частині документа `head` зробити заголовок вікна `title`
   - Заголовки в тексті першого і другого рівнів – `h1` і `h2`
   - Абзаци `p`
3. Змінити розширення файлу з *txt* на *html*;
4. Відкрити цей файл у браузері.

   Номер варіанту вибираємо таким чином.

   Від номера студента за списком беремо залишок по модулю N (кількість варіантів). Отримуємо число від 0 до N-1. Додаємо до нього 1 і отримуємо номер варіанта завдання.

У разі необхідності перекодувати файл з win1251 в utf-8, можна скористатися перекодувальником, наприклад Штірліц.

Тема 2. HTML. Гіперпосилання

Гіперпосилання – частина web-сторінки (текст або картинка), що посилається на якийсь ресурс. При натисканні на гіперпосилання в браузер завантажується вказаний ресурс.

Можливі посилання на:

1. Будь-який елемент в цьому ж документі.
2. На початок іншої web-сторінки (файл, з розширенням *html*, розташований на локальному диску або в комп'ютерній мережі).
3. На будь-який елемент іншої web-сторінки.

Тег `<a>`

Гіперпосилання створюються за допомогою тега `<a>` з атрибутами.

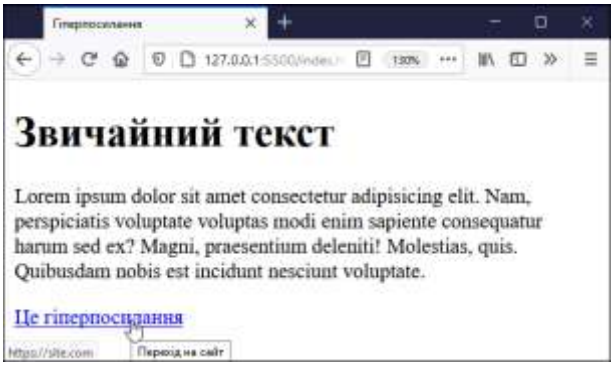
Атрибут `href`

Атрибут `href` обов'язковий. Він вказує, куди буде виконаний перехід.

```
<a href="ресурс">Текст посилання</a>
```

Текст посилання за замовчуванням підкреслений і синього кольору. Якщо перехід за гіперпосиланням вже виконувався, то колір тексту може змінюватися на фіолетовий.

При наведенні вказівника миші на гіперпосилання, він набирає вигляду руки в кулаку з відігнутим вказівним пальцем.



Закладки

При натисканні на гіперпосилання, що посилається на закладку, документ буде прокручений таким чином, щоб адресований елемент знаходився у верхній частині вікна.

Елемент, на який посилаються, може перебувати як до посилання на нього, так і після. На один і той же елемент може бути кілька посилань.

Для того, щоб зробити посилання на закладку потрібно:

1. Присвоїти ідентифікатор (*id*) елементу, на який буде виконуватися перехід. Для цього в відкриваючому тегу елемента потрібно вказати атрибут `id="ім'я"`. Наприклад: `<p id="paragraph">`
2. Посилання на цей елемент буде мати вигляд:
   `<a href="#paragraph">Посилання на параграф</a>`

Завдання 2. Гіперпосилання

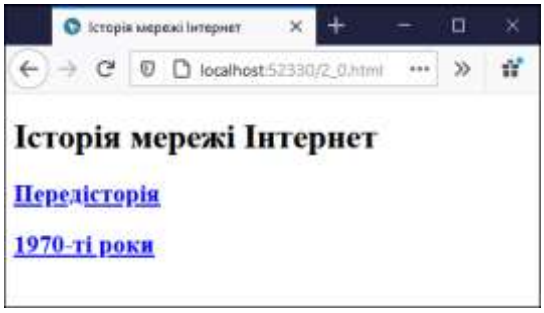
Обов'язково збережіть файли 2_0.html, 2_1.html, 2_2.html. Вони нам будуть потрібні ще в декількох практичних роботах

1. Створений в практичній роботі №1 файл *1_1.html* скопіювати в свою папку ще чотири рази.
2. Дати файлам імена *2.html*, *2_0.html*, *2_1.html*, *2_2.html*
3. У файлі *2.html* зробити закладки. На початку файлу зміст. У змісті посилання на глави в цьому ж файлі. Кожне посилання оформлена абзацом `p`. Після кожного розділу посилання на початок сторінки.
4. У файлі *2_0.html* залишити тільки зміст. Кожне посилання оформлено як заголовок `h2`. При натисканні на першому пункті змісту повинен відкриватися файл *2_1.html*. При натисканні на другому пункті повинен відкриватися файл *2_2.html*.
5. У файлі *2_1.html* залишити тільки перший розділ. В кінці сторінки посилання на зміст (файл *2_0.html*). Посилання оформлене як заголовок `h2`.
6. У файлі *2_2.html* залишити тільки другий розділ. В кінці сторінки посилання на зміст (файл *2_0.html*). Посилання оформлене як заголовок `h2`.

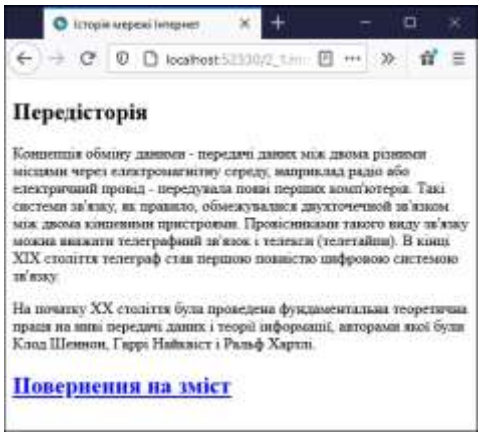
Варіант 1

1. Файл *2.html* повинен мати вигляд:

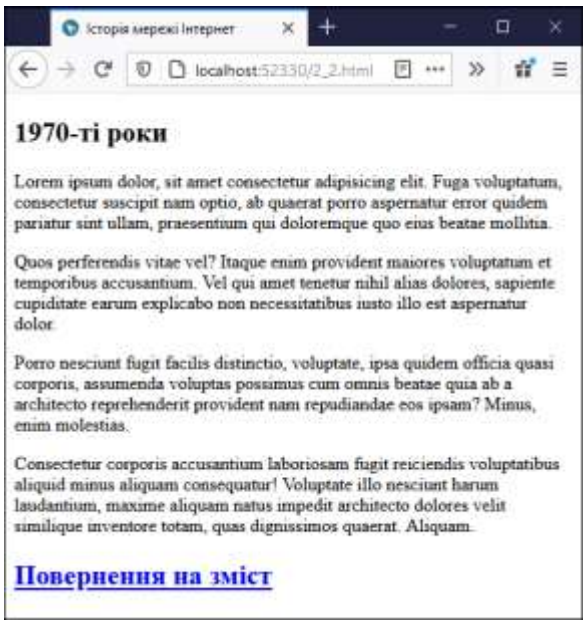
2. Файл *2_0.html* повинен мати вигляд:



3. Файл *2_1.html* повинен мати вигляд:



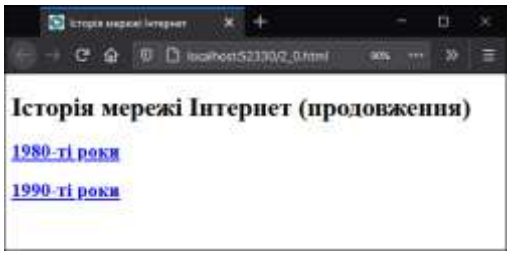
4. Файл *2_2.html* повинен мати вигляд:



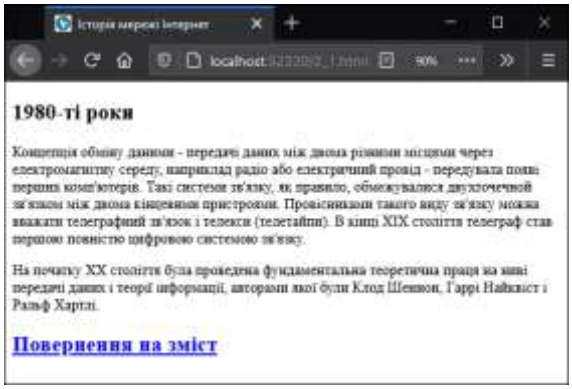
Варіант 2

1. Файл *2.html* повинен мати вигляд:

2. Файл *2_0.html* повинен мати вигляд:



3. Файл *2_1.html* повинен мати вигляд:



4. Файл *2_2.html* повинен мати вигляд:

Тема 3. HTML. Графічні елементи

Графічні формати

У Web зазвичай використовуються три графічних формату: GIF, JPEG, PNG.

Всі ці формати зберігають стисле зображення. Стиснення здійснюється за рахунок відкидання несуттєвої інформації і пошуку в малюнку повторюваних фрагментів.

Формат GIF підходить для збереження зображень з обмеженою кількістю кольорів і великими областями, заповненими суцільним кольором.

Переваги:

- 100% підтримка усіма браузерами.
- Стиснення. У GIF для стиснення застосовується алгоритм, який шукає повторювані послідовності пікселів.
- Прозорість. GIF-файли можуть містити прозорі області, через які буде проглядатися фон. Іншими словами, зображення може бути не тільки прямокутним, але будь-якої форми.
- Анімація. В одному файлі може бути збережено кілька зображень, які, послідовно змінюючись, створюють анімаційний ефект.

Недоліки:

- Обмежена палітра. У GIF можливе збереження не більше 256 різних кольорів.
- GIF не підходить для збереження фотографічних зображень.

Формат JPEG

Формат JPEG був розроблений спеціально для збереження фотографій. JPEG не підходить для збереження зображень з суцільними областями і безліччю деталей.

Переваги:

- 100% підтримка. Всі графічні web-оглядачі підтримують формат JPEG.
- Стиснення. В JPEG зображення стискується за принципом плавних переходів. Це робить неможливим збереження в JPEG дрібних деталей.
- 24-розрядний колір. В JPEG можливо повне збереження трійки RGB для кожного пікселя, тобто JPEG не обмежений 256-а кольорами.

Недоліки:

- Прямокутне зображення. JPEG не підтримує прозорість, тому JPEG-зображення завжди прямокутні.

- Муар. Стиснення з втратами призводить до появи на контрастних областях зображення муару.

Формат PNG

Формат PNG був розроблений як альтернатива GIF. Формат PNG – єдиний з поширених форматів, що дозволяє отримувати повнокольорові зображення з прозорим фоном. PNG підходить для збереження фотографій, на яких неприпустимі ніякі втрати. Крім цього, PNG використовується на зображеннях із суцільними областями.

Переваги:

- Стиснення. У форматі PNG використаний алгоритм стиснення без втрат інформації, заснований на пошуку повторюваних послідовностей пікселів. Конкретний варіант стиснення вибирається залежно від самого зображення для отримання файлу найменшого розміру.
- 48-розрядний колір. В PNG можливо зберігати зображення з 281 474 976 710 656 кольорами.
- Напівпрозорість. В PNG-зображенні можуть бути напівпрозорі області, через які просвічується фон.
- 97% підтримка. На жаль, формат PNG не підтримується всіма браузерами, тому що він був розроблений порівняно недавно.

Єдиним недоліком PNG є відсутність 100% підтримки всіх його можливостей оглядачами і графічними редакторами.

Елемент `<img>`

За допомогою порожнього елемента `<img>` на web-сторінку вставляються зображення. Найпростіший вид тега:

```
<img src="image.png" alt="альтернативний текст">
```

Атрибут `src` вказує шлях до файлу, який вбудовується у web-сторінку. Цей атрибут обов'язковий.

Атрибут `alt` описує альтернативний текст для зображення. Він відображається замість зображення, коли сам малюнок ще не завантажений.

Графічні посилання

Для того, щоб зробити зображення посиланням, елемент `<img>` необхідно розмістити між відкриваючим і закриваючим тегами елементу `<a>`.

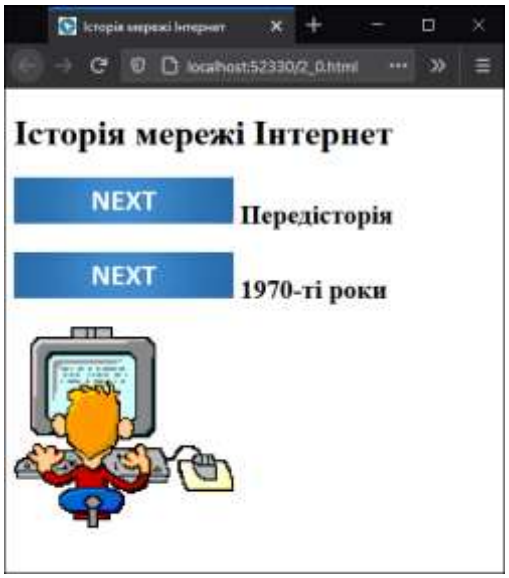
Завдання 3. Робота з графікою

Обов'язково збережіть файли 2_0.html, 2_1.html, 2_2.html. Вони нам будуть потрібні ще в декількох практичних роботах.

Варіант 1.

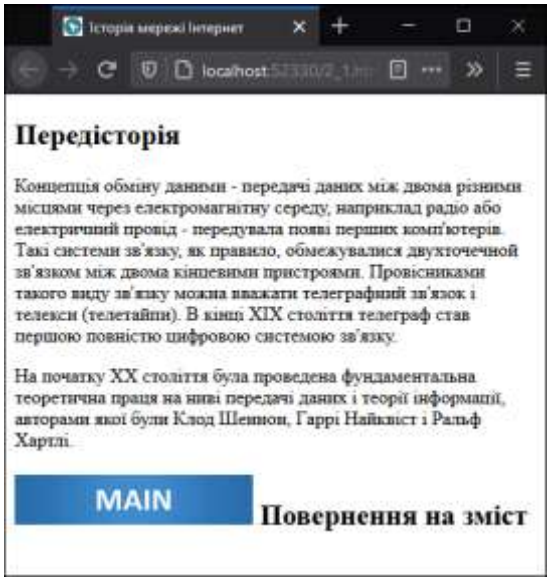
Скопіюйте в свою папку файли: *user.gif, button_next_1.gif, button_main_1.gif*. У даній практичній роботі будемо змінювати файли *2_0.html*, *2_1.html*, *2_2.html*, створені в практичній роботі №2.

1. У файл *2_0.html* додамо зображення *user.gif*, *button_next_1.gif*. Картинки *button_next_1.gif* зробимо посиланнями на файли *2_1.html* і *2_2.html*. Результат:

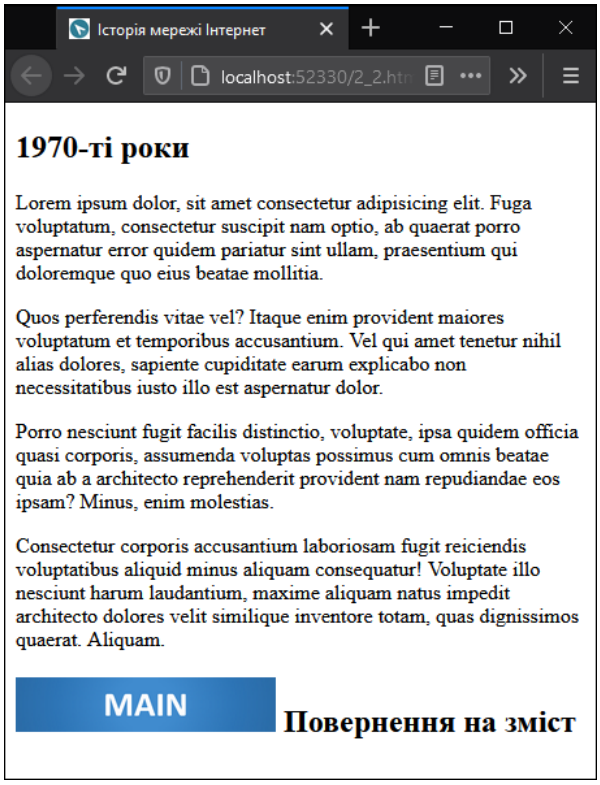


Зображення *user.gif* оформимо як заголовок `h2`.

2. У файл *2_1.html* додамо зображення *button_main_1.gif*. Зробимо це зображення посиланням на файл *2_0.html*. Результат:



3. У файл *2_2.html* додамо картинку *button_main_1.gif*. Зробимо цю картинку посиланням на файл *2_0.html*. результат:

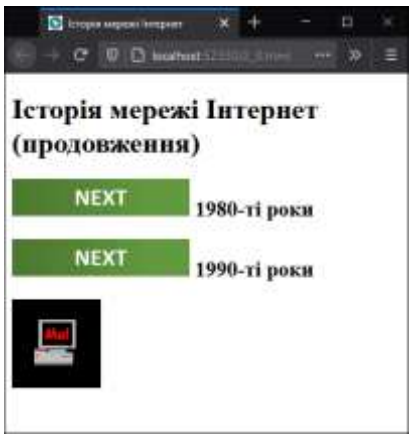


Варіант 2

Скопіюйте в свою теку файли: *virus.gif*, *button_next_2.gif*, *but_3_1.gif*.

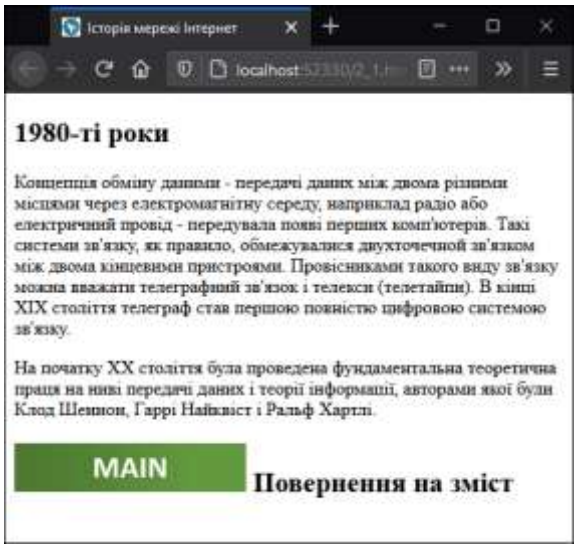
У даній практичній роботі будемо змінювати файли *2_0.html*, *2_1.html*, *2_2.html*, створені в практичній роботі №2.

1. У файл *2_0.html* додамо зображення *virus.gif*, *button_next_2.gif*. Зображення *button_next_2.gif* зробимо посиланнями на файли *2_1.html* і *2_2.html*. Результат:



Зображення *virus.gif* оформимо як заголовок `h2`.

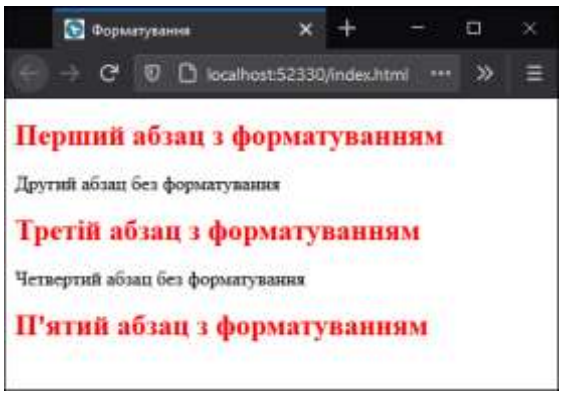
2. У файл *2_1.html* додамо зображення *button_main_2.gif*. Зробимо цю картинку посиланням на файл *2_0.html*. Результат:



3. У файл *2_2.html* додамо зображення *button_main_2.gif*. Зробимо цю картинку посиланням на файл *2_0.html*. Результат:

Тема 4. Введення в CSS

До появи таблиць стилів форматування HTML документів представляло собою форматування за допомогою атрибутів, що визначають колір, вирівнювання, шрифти, розміри і т. д. Наприклад, щоб відформатувати документ так:



Для кожного абзацу з таким форматуванням, доводилося повторювати однакові теги.

При створенні документів великого обсягу таке форматування вимагало величезної кількості часу і сил, оскільки таку конструкцію необхідно було вставляти щоразу, коли було необхідно відформатувати текст.

Це збільшувало розмір документа, і без того сильно роздутого кількістю форматуючих тегів, а, отже, збільшувало тривалість завантаження документа.

```
<body>
    <p><font color=red size=5><b>Перший абзац з форматуванням</b></font></p>
    <p>Другий абзац без форматування</p>
    <p><font color=red size=5><b>Третій абзац з форматуванням</b></font></p>
    <p>Четвертий абзац без форматування</p>
    <p><font color=red size=5><b>П'ятий абзац з форматуванням</b></font></p>
</body>
```

Зрозуміло, що було б зручно окремо задавати стиль для будь-якого з тегів документа.

Поняття стилю

Стиль – це набір правил оформлення та форматування, який може бути застосований до різних елементів сторінки.

Суть таблиць стилів полягає в окремому попередньому завданні стилю для будь-якого з тегів документа.

Наприклад, призначивши один раз, наведений у прикладі стиль для тега <p> нічого не треба більше робити, тобто всюди в документі, де зустрінеться тег <p> він буде представлений в заданому вигляді.

Тобто зміст документа відокремлюється від його оформлення.

Синтаксис таблиці стилів

Розглянемо синтаксис на наступному прикладі:

Кожен рядок називається правилом.

Кожне правило складається з селектора (тут `body`, `h2` і `p`) і визначень. Визначення укладені у фігурні дужки. Якщо визначень кілька, то вони вказуються через крапку з комою.

Кожне визначення складається з властивості (`background-color`, `color`, `font-size`, `text-align`) і значення (`black`, `yellow`, `red`, `36pt`, `center`). Значення вказуються через двокрапку.

Властивості і їх можливі значення залежать від селектора і перераховуються в спеціальних таблицях.

Синтаксис стильового правила:

```
селектор {
    властивість: значення;
    властивість: значення;
  }
```

Таблиці стилів

Стильове правило всередині тега

Стильове правило записується як значення атрибута `style` (після знака = і в лапках). Цей атрибут може бути в кожному тегу.

Стиль, заданий таким чином, поширюється тільки на цей тег і використовується рідко.

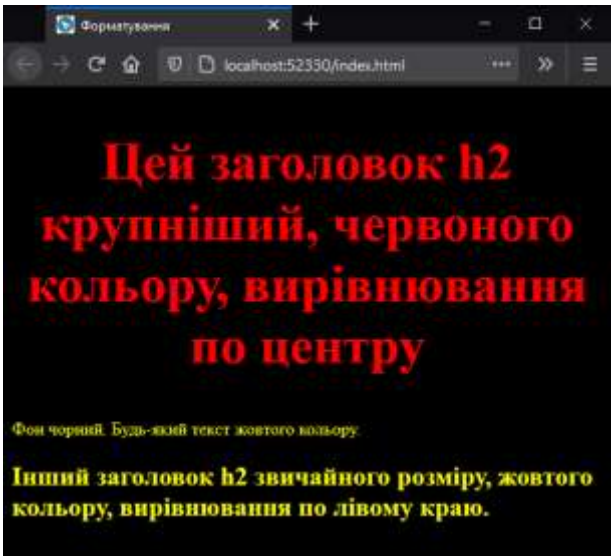
Синтаксис стильового правила звичайний:

```
<p style="text-align: center; color: red;">
```

Загальні для всього документа властивості задаються в тезі `<body>`, а властивості одного заголовка задаються в тезі цього заголовка `<h2>`.

```html
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Форматування</title>
</head>
<body style="background-color: black; color: yellow;">
    <h2 style="font-size: 36pt; color: red; text-align: center;">
        Цей заголовок h2 крупніший, червоного кольору, вирівнювання по центру
    </h2>
    <p>Фон чорний. Будь-який текст жовтого кольору.</p>
    <h2>
        Інший заголовок h2 звичайного розміру, жовтого кольору, вирівнювання по лівому краю.
    </h2>
</body>
</html>
```

В результаті, в даному документі фон чорний, колір будь-якого тексту жовтий, крім одного заголовка. Цей заголовок, не жовтий, а червоний, крім того більше за розміром звичайного і вирівняний по центру.



Таблиця стилів в `head` документа

В даному випадку таблиця стилів поміщається в елемент `style` (між тегами `<style>` і `</style>`) в головній частині між тегами `<head>` і `</head>`. Наприклад:

```html
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Форматування</title>
    <style>
        p {color: blue;}
        h2 {font-size: 36px; color: red; text-align: center;}
    </style>
</head>
<body>
    <h2>Перший заголовок h2 крупніший, червоного кольору, вирівнювання по центру</h2>
    <p>Перший абзац p синього кольору</p>
    <h2>Другий заголовок h2 має такий же стиль, як і перший заголовок h2.</h2>
    <p>Другий абзац p синього кольору</p>
</body>
</html>
```

Стиль, заданий таким чином, поширюється на весь документ.

Синтаксис таблиці стилів звичайний – селектор і до нього в фігурних дужках стильові правила через «;».

Форматування для елементів p, h2 були написані один раз в таблиці стилів, а застосовувалися в кожному елементі p, h2.

Таблиця стилів в окремому файлі

В даному випадку таблиця стилів поміщається в окремий текстовий файл. Ім'я цього файлу може бути будь-яким, а розширення обов'язково *css*. Синтаксис таблиці стилів звичайний – селектор і до нього в фігурних дужках стильові правила через «;». Нехай, наприклад, вміст файлу *my.css* буде таким

```css
body {background-color: pink;}
p {color: maroon;}
h2 {font-size: 36px; color: red; text-align: center;}
```

Стиль, заданий в цьому файлі, буде поширюватися на всі документи, що містять посилання на цей файл. Наприклад:

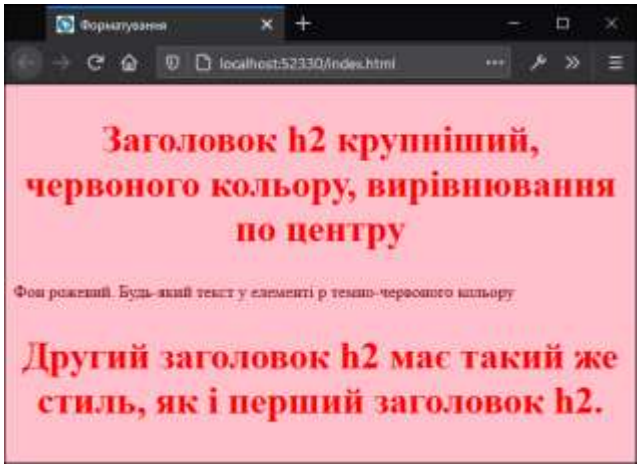
```html
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Форматування</title>
    <link rel="stylesheet" href="my.css">
</head>
```

```
<body>
    <h2>Заголовок h2 крупніший, червоного кольору, вирівнювання по центру</h2>
    <p>Фон рожевий. Будь-який текст у елементі p темно-червоного кольору</p>
    <h2>Другий заголовок h2 має такий же стиль, як і перший заголовок h2.</h2>
</body>
</html>
```

Посилання на таблицю стилів з основного файлу вказується в елементі `link`. Цей елемент розміщується в `head`. Атрибути елемента `link` мають такі значення:

```
<link rel="stylesheet" href="style.css">
```

Результат форматування за допомогою цієї таблиці стилів:



Завдання 4. Властивості тексту, таблиці стилів CSS.

Обов'язково збережіть файли 2_0.html, 2_1.html, 2_2.html. Вони нам будуть потрібні ще в декількох практичних роботах!!!

Змініть файли *2_0.html*, *2_1.html*, *2_2.html*, створені в попередній роботі таким чином, щоб вони містили посилання на таблицю стилів, яка буде знаходитися в файлі *my.css*. В останньому абзаці файлу *2_2.html* для виділення використовується елемент `span`.

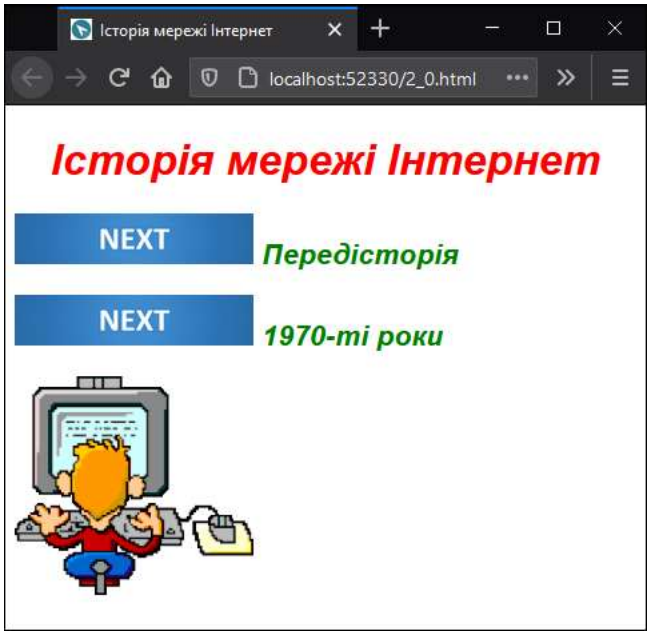
Варіант 1

У файлі *my.css* встановіть наступні стилі:

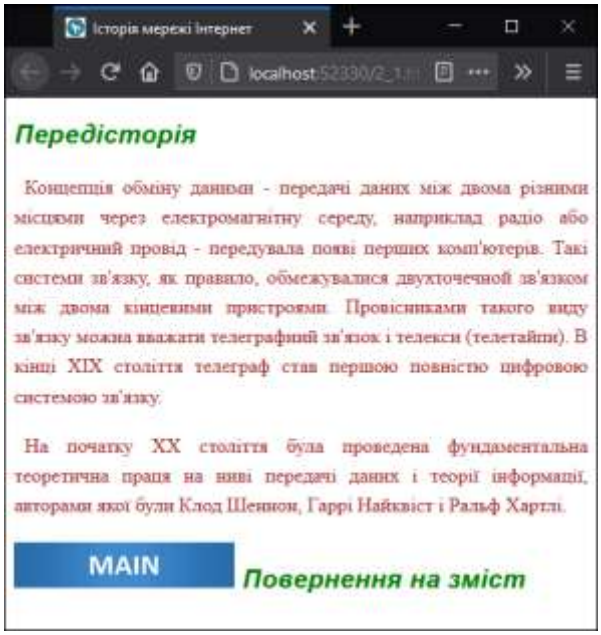
- Заголовок першого рівня (`h1`): шрифт `Arial`, розмір `36px`, колір червоний, курсив;
- Заголовок другого рівня (`h2`): шрифт `Arial`, розмір `24px`, колір зелений, курсив;
- Абзац (`p`): шрифт `Times New Roman`, розмір `18px`, колір коричневий;
- `span`: колір синій, жирний, розряджений `4px`.

- Заголовок першого рівня (`h1`) вирівняний по центру;
- Абзац (`p`) вирівняний по ширині, міжрядковий інтервал полуторний, червоний рядок;
- `span`: підкреслення виділених слів.

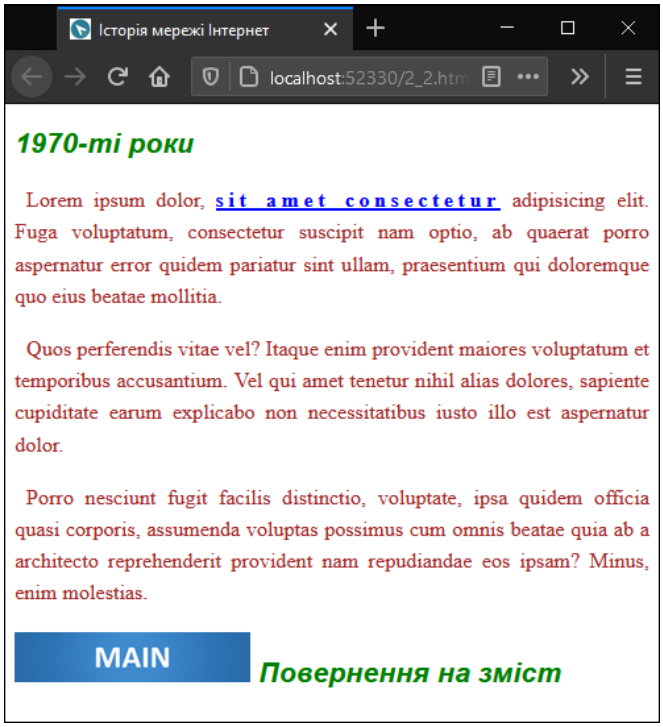
1. Результат в файлі *2_0.html*:



2. Результат в файлі *2_1.html*:



3. Результат в файлі *2_2.html*:

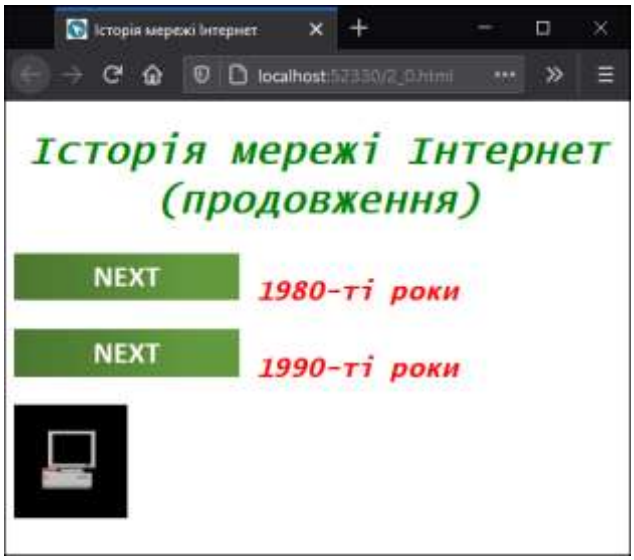


Варіант 2

У файлі *my.css* встановіть наступні стилі:

- Заголовок першого рівня (`h1`): шрифт `Lucida Console`, розмір `36px`, колір зелений, курсив;
- Заголовок другого рівня (`h2`): шрифт `Lucida Console`, розмір `24px`, колір червоний, курсив;
- Абзац (`p`): шрифт `Courier New`, розмір `18px`, колір коричневий;
- `span`: колір синій, жирний, розряджений `4px`.
- Заголовок першого рівня (`h1`) вирівняний по центру;
- Абзац (`p`) вирівняний по ширині, міжрядковий інтервал полуторний, червоний рядок;
- `span`: підкреслення виділених слів.

1. Результат в файлі *2_0.html*:



2. Результат в файлі *2_2.html*:

Тема 5. Javascript. Основні оператори мови

Мова JavaScript була розроблена Бренданом Айком з корпорації Netscape Communications у 1995 році. JavaScript має більше спільного з Self, Lisp і ALGOL ніж з Java.

Комітет TC39 об'єднав програмістів Netscape, Sun, Microsoft, Borland, NOMBAS і інших компаній, які проявляють інтерес до майбутнього мов сценаріїв, і за кілька місяців розробив стандарт ECMA-262, який визначив нову мову сценаріїв з назвою ECMAScript.

Розробники браузерів зі змінним успіхом використовували ECMAScript як основу для реалізації своїх версій JavaScript.

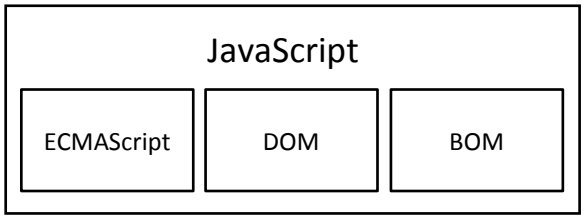
Будемо розглядати мову ECMAScript 6 (ES6) опублікований Ecma в червні 2015 року.

Web-документ, який відображається браузером, – це результат виконання програм, створених на різних мовах. Для опису структури і загального зовнішнього вигляду можна використовувати мову розмітки HTML, для опису особливостей зовнішнього вигляду, відмінного від стандартизованого – краще використовувати мову стилів CSS. Для опису поведінки документа, його реакції на дії користувача використовується мова сценаріїв JavaScript.

JavaScript код виконує браузер. Тобто він працює на стороні клієнта. У нього вбудований інтерпретатор JavaScript. Отже, виконання програми залежить від того, коли цей інтерпретатор отримує управління і який саме це браузер. Останні версії JavaScript можуть не підтримуватися деякими браузерами.

Повна реалізація JavaScript складається з трьох частин:

- ядро (ECMAScript);
- об'єктна модель документа (Document Object Model, DOM);
- об'єктна модель браузера (Browser Object Model, BOM).



На базовому рівні ECMAScript визначає наступні, незалежні від браузера, частини мови:

- синтаксис;
- типи;
- інструкції;
- ключові слова;
- зарезервовані слова;

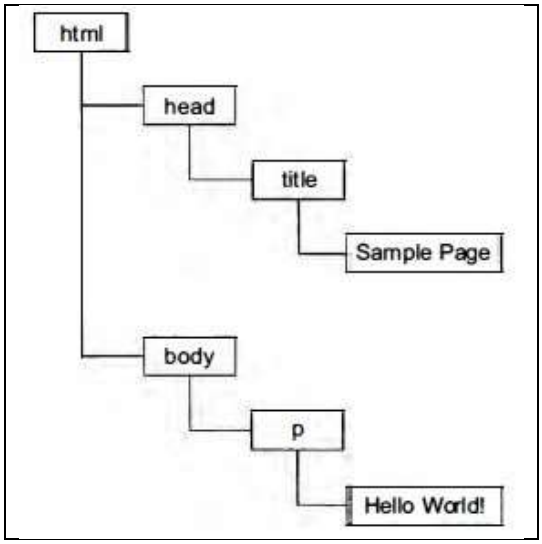
- оператори;
- об'єкти.

Об'єктна модель документа

Об'єктна модель документа (DOM) – це прикладний програмний інтерфейс (Application Programming Interface, API) для XML, застосування якого було розширено на HTML. В DOM вся сторінка представляється як ієрархія вузлів. Кожен елемент HTML- або XML-сторінки є вузлом певного типу, що містить ті чи інші дані. Розглянемо наступну HTM-сторінку:

```
<html>
    <head>
      <title>Sample Page</title>
    </head>
    <body>
      <p>Hello World!</p>
    </body>
</html>
```

Цей код можна зобразити за допомогою DOM як ієрархію вузлів.



Використовуючи DOM API, можна з легкістю видаляти вузли, додавати, замінювати і змінювати їх. Завдяки реалізації динамічного HTML (Dynamic HTML, DHTML), в Internet Explorer 4 і Netscape Navigator 4 розробники вперше змогли змінювати вид і контент веб-сторінок без їх перезавантаження. Це стало важливим етапом розвитку веб-технологій, але виникла серйозна проблема. Netscape і Microsoft вибрали різні шляхи розвитку DHTML, що завершило період, коли можна було писати HTM-сторінки, не замислюючись про веб-браузери.

Для роботи з новими інтерфейсами в DOM Level 2 були представлені нові модулі:

- DOM Views – інтерфейси для відстеження різних представлень документа (наприклад, документ до і після застосування стилів CSS);
- DOM Events – інтерфейси для подій і обробки подій;
- DOM Style – інтерфейси для роботи зі стилізацією елементів за допомогою CSS;
- DOM Traversal and Range – інтерфейси для обходу елементів дерева документа і виконання операцій над ними.

DOM Level 3 доповнила DOM уніфікованими методами завантаження і збереження документів.

Об'єктна модель браузера

В Internet Explorer 3 і Netscape Navigator 3 була представлена об'єктна модель браузера (BOM), яка забезпечує доступ до вікна браузера і дозволяє маніпулювати його елементами. Використовуючи BOM, можна взаємодіяти з браузером поза контекстом відображуваної сторінки. До недавніх пір BOM була єдиною частиною реалізації JavaScript, що не має стандарту, через що при роботі з нею часто виникали проблеми. Формалізація багатьох елементів BOM в HTML5 змінила ситуацію на краще, прояснила багато неясних аспектів моделі.

BOM регламентує роботу з вікном і фреймами браузера, але будь-яке специфічне для браузера JavaScript розширення теж зазвичай вважається частиною BOM. Ось деякі такі розширення:

- функція відображення спливаючих вікон в браузері;
- можливість переміщати, закривати і змінювати розміри вікна браузера;
- об'єкт `navigator`, що надає докладні відомості про браузер;
- об'єкт `location`, що надає докладні відомості про сторінку, завантажену в браузері;
- об'єкт `screen`, що надає докладні відомості про дозвіл екрана;
- підтримка cookie-файлів;
- настроюються настроюються об'єкти, включаючи `XMLHttpRequest`, а також `ActiveXObject` в Internet Explorer.

Підключення JavaScript коду на сторінку

Виконує JavaScript код браузер. У нього вбудований інтерпретатор JavaScript. Отже, виконання програми залежить від того, коли цей інтерпретатор отримує управління. Наведемо кілька способів розміщення коду JavaScript:

1. В контейнері заголовка сторінки `<head>`

```
<head>
    …
    <script type="text/javascript">команди скрипта</script>
    …
</head>
```

2. В контейнері тіла сторінки <body>

```
<body>
    …
    <script>команди скрипта</script>
    …
</body>
```

3. Обробник події вказується безпосередньо в тому тегу, події якого потрібно обробляти, без укладення в теги `<script></script>`

```
<input type="button" value="Натиснути" onClick="window.alert('Натисніть ще раз
')">
```

4. У зовнішньому файлі. За аналогією з тим, як стилі підключаються до сторінки за допомогою елемента `link`, сценарії підключаються за допомогою елемента `script`, тільки файл має розширення не *css*, а *js*.

```
<head>
    …
    <script type="text/javascript" src="my.js"></script>
    …
</head>
```

Виконання операторів сценарію

Існує кілька способів визначення моменту запуску сценарію. Ось деякі з них:

- при завантаженні документа;
- відразу після завантаження документа;
- у відповідь на дії користувача.

Коди елементів `<script>` обробляються в порядку їх розташування на сторінці HTML якщо у цих тегів немає атрибутів `defer` і `async`. Дані атрибути дозволяють відкласти виконання сценарію зовнішнього файлу зі скриптом:

- до повного завантаження і візуалізації web-сторінки,
- паралельно із завантаженням даного скрипта виконувати наступні дії на сторінці.

Велика кількість сценаріїв, виконуваних при завантаженні, наприклад, в заголовку сторінки, може істотно сповільнити візуалізацію документа. Тому

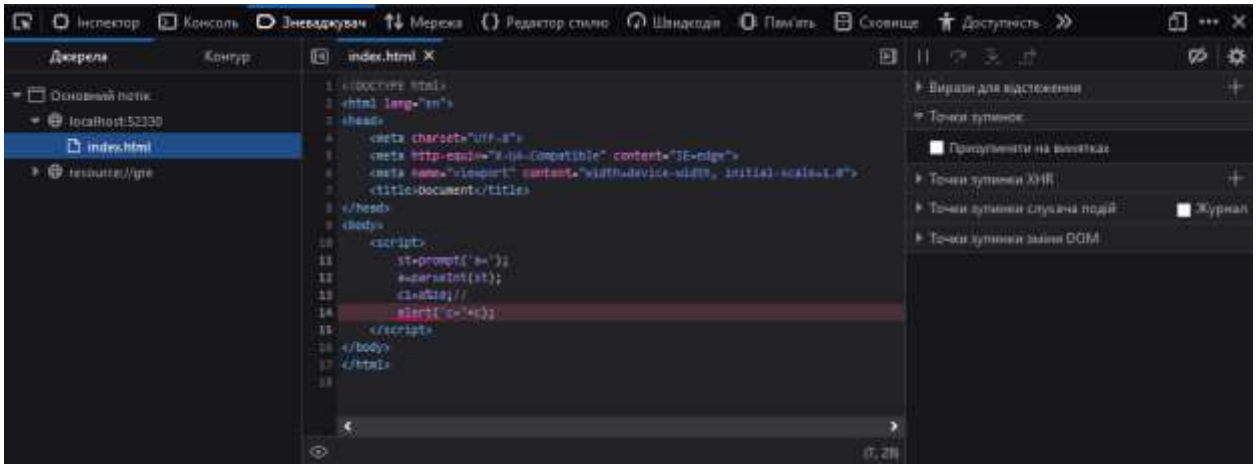
найчастіше ми будемо користуватися сценаріями, виконуваними у відповідь на дії користувача.

Налагодження скрипта

Як правило в браузерах за замовчуванням вже активізований дозвіл використовувати JavaScript. Якщо це не так, то це можна легко зробити в налаштуваннях будь-якого з них.

Помилки при використанні HTML, CSS браузером просто ігноруються. А виводяться лише коректна частина сторінки. Налагодження подібних документів тому ускладнюється. Для контролю синтаксису для HTML, CSS, JavaScript як і раніше можемо використовувати редактори текстів Notepad++. Налагодження JavaScript коду спрощується, тому що у всіх браузерах є спеціальна панель, яка називається «інструменти розробника» (засоби розробника). Цю панель можна запустити за допомогою натискання на клавіатурі сполучення клавіш [Ctrl]+[Shift]+[I] або клавіши [F12].

В Firefox ця панель виглядає наступним чином:



Номер рядка з помилкою, в наведеному прикладі, – 14. У операторі зроблено спробу вивести значення змінної «c», а в програмі значення зберігається в змінній «c1»!

У самому налагоджувачи ми можемо побачити посилання на помилки, але коригувати код потрібно все ж в редакторі.

Вступ в JavaScript

JavaScript – залежить від регістра. Імена JavaScript і Javascript – різні імена!

Всі ключові слова використовують тільки нижній регістр.

Вимоги до імен змінних такі ж, як в C, Python.

Оператори розділяються крапкою з комою, яку можна опустити, якщо оператор закінчується символом нового рядка (Enter).

Коментарі:

```
// однорядковий коментар,

/*
багаторядковий
коментар
*/
```

Типи даних

Змінні оголошувати не обов'язково. Однак можна і оголошувати або оголошувати за допомогою оператора `var`. Якщо змінна оголошена в тілі функції, то вона є локальною змінною. Немає суворої типізації змінних.

У момент використання значення змінної вона повинна мати тип і значення. Можливо оголошення с одночасною ініціалізацією.

Приклади:

Оголошується цілочисельна змінна x, що має десяткове значення 123:
```
var s = 123
```

Оголошується змінна з плаваючою точкою, яка має десяткове значення:
```
var d = 3.14
```

Оголошується строкова змінна:
```
var str23 = 'Строкова змінна'
```

Оголошується логічна змінна:
```
var p = true
```

Тип змінної може змінюватися в процесі виконання програми. Якщо у виразі містяться і числові і рядкові змінні, то числові змінні автоматично приводяться до рядкового вигляду.

Операції

Арифметичні операції

Операція	Дія	Приклад	Значення, яке прийме X
+	Додавання	x=100+5 str2='Початок' str1=str2+' кінець'	105 Початок кінець
-	Віднімання	x=100-5	95
*	Множення	x=2*3	6
/	Ділення	x=12/3	4
%	Залишок від ділення (аналогічно mod)	x=16%3	1
++	Значення збільшується на 1	X=2; X++;	3
--	Значення зменшується на 1	X=2; X--;	1

Операції порівняння

Операція	Дія	Приклад	Аналогічно, в Паскалі
==	Дорівнює	X==10	X=10
!=	Не дорівнює	x!=5	x<>5
>	Більше	x>0	x>0
<	Менше	x<4	x<4
>=	Більше або дорівнює	x>=y	x>=y
<=	Менше або дорівнює	X<=5	x<=5

Логічні операції

Операція	Дія	Приклад	Аналогічно, в Паскалі
&&	Аналогічно логічній операції and	X>=2 && y>=2	(X>=2)and(Y>=2)
||	Аналогічно логічній операції or	x>0 || y>0	(x>0)or(y>0)
!	Аналогічно логічній операції not	!(1 < x && x < 10)	Not((1 < x) and (x < 10))

Оператори присвоювання

Операція	Дія	Приклад	Значення, яке прийме X	Аналогічно, в Паскалі
=	Привласнює значення змінній	`X=1000;`	1000	`X:=1000;`
+=	Збільшує значення змінної на зазначену величину	`X=1000;` `X+=100;`	1100	`X:=1000;` `X:=X+100;`
-=	Зменшує значення змінної на зазначену величину	`X=1000;` `X-=12;`	988	`X:=1000;` `X:=X-12;`
*=	Примножує значення змінної на зазначену величину	`X=1000;` `X*=2;`	2000	`X:=1000;` `X:=X*2;`
/=	Поділяє значення змінної на зазначену величину	`X=1000;` `X/=2;`	500	`X:=1000;` `X:=X/2;`
%=	Ділить значення змінної на зазначену величину і повертає залишок	`X=1000;` `X%=5;`	0	`X:=1000;` `X:=X mod 5;`

Умовний оператор

JavaScript	Pascal
`if (a==2) z=2; else z=3`	`if a=2 then z:=2 else z:=3;`

`if (x>=2 && x<=6)`
`{`
`  y=0; z=1;`
`}`
`else`
`{`
`  y=1; z=0;`
`}`
	`if (x>=2)and(x<=6) then`
	`begin`
	`  y:=0; z:=1;`
	`end`
	`else`
	`begin`
	`  y:=1; z:=0;`
	`end;`

Особливості.

Вся умова береться в дужки. Прості умови в дужки можна не брати. Крапка з комою перед `else` обов'язково ставиться, якщо перед `else` немає фігурної дужки «`}`». Якщо дужка є, то «`;`» ставити не можна.

Найпростіший ввод/виведення даних. Діалогові вікна

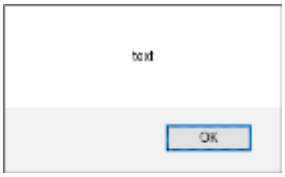
В JavaScript існує 3 найпростіші функції, методу, що дозволяють користувачеві виводити діалогові вікна:

```
alert ("строка")
```

Метод `alert` використовується для виведення найпростішого діалогового вікна, що містить текст повідомлення і єдину кнопку [Ok]. Програма виводить повідомлення і чекає натиснення кнопки. Після натискання на кнопку, програма починає виконуватися далі. Текст повідомлення може зчіплюватися з будь-якою текстовою змінною за допомогою знака «+». Щоб текст виводився в кілька рядків використовують символи «\ n»

Приклад

```
<body>
    <script>
        let t='text';
        alert(t);
    </script>
</body>
```



Confirm (“текст”)

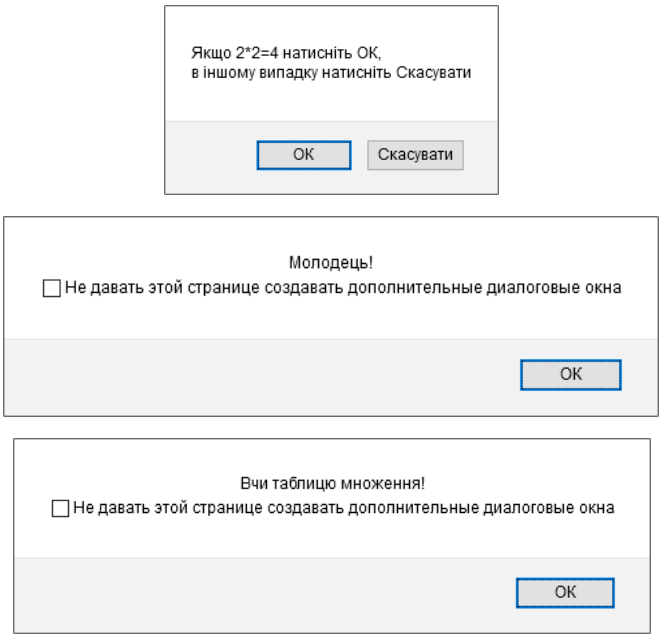
Метод `confirm` використовується в тих випадках, коли користувач повинен зробити вибір. Метод `confirm` дозволяє користувачеві вивести діалогове вікно, що містить текст питання і кнопки **[ОК]** і **[Відмина]**.

Функція `confirm` повертає логічне значення в залежності від натиснутої користувачем кнопки: **[ОК]** відповідає значенню `true`, **[Відмина]** – значенню `false`.

Як правило, результат роботи функції присвоюють логічній змінній, для подальшого аналізу, як це показано в прикладі.

Приклад

```
<body>
    <script>
        var result=confirm("Якщо 2*2=4 натисніть ОК,"+"\n"+"в іншому випадку на
тисніть Скасувати");
        if(result) alert("Молодець!")
        else alert("Вчи таблицю множення!");
    </script>
</body>
```



prompt (“текст1”, “текст2”)

Метод `prompt` використовується в тих випадках, коли користувачеві потрібно ввести значення в змінну. У вікно виводиться повідомлення «***текст1***», в поле введення поміщується значення за замовчування «***текст2***».

Цей метод дозволяє вивести діалогове вікно запиту на введення даних. Результат роботи функції привласнюють змінній рядкового типу.

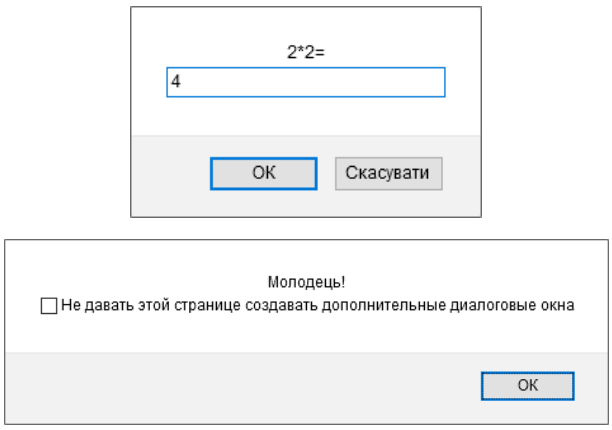
Якщо введені дані потрібно використовувати в арифметичних виразах, необхідно виконати перетворення введеного рядка до числового типу. Це можна зробити за допомогою наступних функцій:

- `parseInt("текст")` – перетворює рядок у ціле число;
- `parseFloat("текст")` – перетворює рядок в дійсне число.

Приклад 1

```
<body>
    <script>
        var str=prompt('2*2=','0');
        if(str=="4") alert('Молодець!')
        else alert('Вчи таблицю множення!');
    </script>
</body>
```

Прибираємо 0 і вводимо 4



Оператор циклу for

```
for(початок; умова; крок)
 {
    //...тіло циклу...
 }
```

Наприклад, цикл нижче виводить значення від 0 до 5 (не включаючи 3):

```
for(i=0; i<5; i++)
 {
     alert(i);
 }
```

Оператор вибору `switch`

На відміну від умовного оператора `if` відразу організовує розгалуження алгоритму не на 2 частини, а на безліч. Як умова розгалуження можна вибирати вираз, в найпростішому випадку змінну.

Приклад 1

Порівняння значення змінної ***a*** з різними константами.

```
var a = 4;
switch(a)
{
  case 3:   alert('Мало');      break
  case 4:   alert('Точно!');    break
  case 5:   alert('Перебір');   break
  default:  alert('Я таких значень не знаю')
}
```

Результат буде – «Точно!»

Завдання 5. Основні оператори JavaScript

Варіант 1.

1. Дано значення відстані в сантиметрах. Написати програму переведення його в міліметри.
2. Дано ціле число N. Якщо воно закінчується на 4 або 5, виведіть "Так", інакше – "Ні".
3. Дано натуральне число N. Знайдіть кількість його парних дільників.

Варіант 2.

1. Дано дві сторони різних квадратів. Обчисліть на скільки площа одного квадрата більше площі іншого.
2. Дано два цілих числа X і Y. Якщо число X закінчується так само, як і Y виведіть "Так", інакше – "Ні".
3. Знайти кількість цілих чисел з інтервалу від 1 до 50, у яких є 3 дільника.

Тема 6. Вбудовані об'єкти мови JavaScript

Основною одиницею в мові JavaScript є об'єкт, який об'єднує в собі дані (властивості) і засоби обробки цих даних (методи).

Всі об'єкти, які використовуються в мові JavaScript, діляться на такі великі групи:

- Вбудовані об'єкти базового мови
  - ✓ Об'єкт `Math`
  - ✓ Об'єкт `Array`
  - ✓ Об'єкт `Date`
  - ✓ Об'єкт `String`
- Об'єкти документа

Вбудовані функції

`parseInt(str)`

Перетворення `str` строки в ціле число.

Якщо `parseInt` стикається з неприпустимим символом, то повертає значення, засноване на підстроки, наступного до цього символу, ігноруючи всі наступні. Якщо перший же символ не допустимий, `parseInt` повертає значення `NaN`.

`parseFloat(str)`

Перетворення рядка `str` в число з плаваючою точкою.

Якщо `parseFloat` стикається з неприпустимим символом, то повертає значення, засноване на підрядку, наступного до цього символу, ігноруючи всі наступні. Якщо перший же символ не допустимий, `parseFloat` повертає `NaN`.

`isNaN(str)`

Якщо рядок `str` не є числом, повертає `true`, інакше – `false`.

Об'єкт `Math`

У мові JavaScript існує спеціальний об'єкт `Math`, в якому зібрані основні математичні функції і константи. Всі властивості і методи об'єкта є статичними, тому сам об'єкт створювати не потрібно. У таблицях нижче наведено список деяких з його методів.

Метод	Опис
`abs(число)`	Повертає модуль (абсолютну величину) числа.

`sqrt(число)`	Повертає квадратний корінь з числа.
`pow(основа, ступінь)`	Повертає результат піднесення основи в зазначену ступінь. Наприклад, `Math.pow(5, 3)` поверне $5^3 = 125$.
`random()`	Повертає псевдовипадкове число в діапазоні від 0 до 1.
`ceil(число)`	Проводить округлення в більшу сторону, тобто повертає найменше ціле число, більше або рівне аргументу.
`floor(число)`	Проводить округлення в меншу сторону, тобто повертає найбільше ціле число, менше або рівне аргументу.
`round(число)`	Округлює вказаний аргумент до цілочисельного значення.
`cos(число)`	Повертає косинус числа.
`sin(число)`	Повертає синус числа.
`exp(число)`	Зводить число e (основу натурального логарифма) в зазначену ступінь.
`log(число)`	Повертає натуральний логарифм числа.
`PI`	Число пи

Щоб задіяти метод (зазвичай це називають викликом методу), в операторі JavaScript потрібно зробити на нього посилання. Для цього використовується синтаксис з використанням точки. Наприклад:

```
x=Math.sqrt(4.1);
```

Об'єкт Array

Об'єкт `Array` призначений для зберігання масивів даних. Масив – це упорядкований набір елементів. Доступ до окремого елементу проводиться по імені і індексу (номеру). Нумерація елементів в JavaScript починається з нуля.

Приклад:

Заповнити одновимірний цілочисельний масив випадковими числами з (5;20) і вивести на екран

```
<script>
  ar = new Array(10);
  s = "";
  for (i = 0; i < ar.length; i++) {
    x = Math.random() * 15 + 5;
    ar[i] = Math.floor(x);
    s = s + " " + ar[i];
  }
  alert(s);
</script>
```

Об'єкт `Date`

Об'єкт `Date` призначений для маніпуляцій з датами і часом. Його примітивним значенням є число, яке дорівнює кількості мілісекунд щодо базового часу, рівного півночі 1 січня 1970 р. День складається з 86400000 мілісекунд.

Об'єкт і екземпляр об'єкта

Об'єкт – це шаблон. Примірник об'єкта – це робоча копія. Наприклад, об'єктом є комплект документації на заводі, по якій виготовляються телевізори. Сам телевізор є екземпляром цього об'єкта. Всі телевізори, які сходять з конвеєра, мають одні і ті ж властивості зображення і одні і ті ж методи управління цими властивостями.

Створення екземпляра об'єкта `Date`

Для створення екземпляра об'єкта використовується ключове слово `new`.

Створення екземпляра об'єкта з поточною датою і часом: `new Date()`

```
var a = new Date();
```

В змінній `a` поточна дата і час.

Створення екземпляра об'єкта з датою і часом, заданими аргументами конструктора:

```
new Date(год, месяц, день [,часы [,минуты [,секунды [,мс]?]?]?]?)
```

Тут:

- рік – числовий вираз, що задає повний номер року (наприклад, 1988, а не 88);
- місяць – числовий вираз, що задає номер місяця (0 = січень, 1 = лютий, ..., 11 = грудень);
- день – числовий вираз, що задає номер дня у місяці від 1 до 31;
- години – необов'язковий числовий вираз, що задає номер години від 0 до 23;
- хвилини – необов'язковий числовий вираз, що задає номер хвилини від 0 до 59;
- секунди – необов'язковий числовий вираз, що задає номер секунди від 0 до 59;
- мс – необов'язковий числовий вираз, що задає номер мілісекунди від 0 до 999.

Наприклад,

```
var c = new Date(1958, 4, 21, 10, 15);
```

В змінній «c» дата – 21 травня 1958 року і час 10 годин 15 хв. Нумерація місяців починається з 0.

Методи отримання компонентів об'єкта `Date`

Метод – це дія яка виконується для об'єкта або з об'єктом. За своєю суттю це команда, але її дії пов'язані з певним об'єктом. У кожного об'єкта може бути багато методів.

Метод	Опис
`getTime()`	Повертає примітивне значення об'єкта.
`getFullYear()`	Повертає номер року за місцевим часом
`getMonth()`	Повертає місяць за місцевим часом. Нумерація місяців з 0 (0 – січ, 1 – лют ...)
`getDate()`	Повертає число за місцевим часом.
`getDay()`	Повертає день тижня за місцевим часом. Нумерація днів тижня з 0 (0 – неділя, 1 – понед ...)
`getHours()`	Повертає години за місцевим часом.
`getMinutes()`	Повертає хвилини за місцевим часом.
`getSeconds()`	Повертає секунди за місцевим часом.

Наприклад, виведемо в вікно номер поточного року.

```
<script>
    td = new Date();
    God = td.getFullYear();
    alert(God);
</script>
```

Об'єкт `String`

Об'єкт `String` призначений для зберігання текстових даних. Доступ до окремого елементу проводиться по імені і індексу (номера). Нумерація символів в рядку в JavaScript починається з нуля. Методи об'єкта:

- Метод `length` повертає довжину рядка.
- Метод `indexOf` повертає перше входження символу
- Метод `lastIndexOf`, який повертає останнє входження символу
- Метод `charAt` повертає символ, що стоїть на зазначеній позиції.

Приклад пошуку і виведення довжини рядка:

```
<script>
    str = "Це рядок символів";
    ln = str.length;
    alert('Довжина рядка=' + ln);
</script>
```

Завдання 6. Вбудовані об'єкти мови JavaScript

Варіант 1.

1. Написати програму, яка водить значення **x**, **y**, а потім обчислює і виводить значення виразу **z**:

$$z = \frac{xy}{x - y} + \sqrt{|y^3 - x|}$$

2. Заповніть цілочисельний масив А з 10 елементів випадковими цілими числами з проміжку (0;50). Знайдіть в цьому масиві числа, які починаються на 3.
3. Дано число, номер місяця і рік. Створити сторінку, яка при завантаженні виводить у вікно цю дату у вигляді: число, назва місяця, рік і назва дня тижня.
4. Дано рядок і символ. Скільки разів зустрічається цей символ в рядку?

Варіант 2.

1. Написати програму, яка вводить значення **x**, **y**, а потім обчислює і виводить значення виразу **z**:

$$z = \frac{|x^3 - y^3|}{(x + y)^2} - \sqrt{\frac{1 + |y|}{x}}$$

2. Заповніть цілочисельний масив А з 20 елементів випадковими цілими числами з проміжку (0;60). Знайдіть в цьому масиві числа, які кратні 7.
3. Створити сторінку, яка при завантаженні виводить у вікно сьогоднішню дату у вигляді: число, назва місяця, рік і назва дня тижня.
4. Дан рядок. Скільки разів зустрічаються цифри в цьому рядку?

Литература

1. Олифер В. Г., Олифер Н. А., Компьютерные сети. Принципы, технологии, протоколы: Учебник для вузов.4-е изд. – СПб.: Питер, 2010. – 944 е.
2. Куроуз Дж., Компьютерные сети: Нисходящий подход, 6-издание, М. Издательство «Э», 2016. – 912с.
3. Никсон Р.Н64 Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 4-е изд. – СПб.: Питер, 2016. – 768 с.
4. Браун, Этан, Изучаем JavaScript: руководство по созданию современных вебсайтов, 3-е изд.:Пер. с англ. – СпБ. : ООО «Альфа-книга»; 2017. – 368 с.
5. Закас Н., JavaScript для профессиональных веб-разработчиков / [Пер. с англ. А. Лютича]. – СПб.: Питер, 2015. – 960 с.
6. http://it.inf.od.ua – сайт кафедри інформаційних технологій, на якому розміщено робочі матеріали з курсу
7. http://dspace.onua.edu.ua - eNUOLAIR – депозитарій (архів) НУ ОЮА
8. https://dl.nure.ua/course/view.php?id=73 – курс Web_технології та Web-дизайн

ЕЛЕКТРОННЕ НАВЧАЛЬНЕ ВИДАННЯ

Сергій В'ячеславович Орлов
Анатолій Арнольдович Толокнов

WEB-ТЕХНОЛОГІЇ ТА ПРОГРАМУВАННЯ

Методичні рекомендації
до виконання лабораторних робіт
з дисципліни
«Web-технології та програмування»

для підготовки здобувачів вищої освіти
галузі знань 12 «Інформаційні технології»