МЕТОДИЧНІ ВКАЗІВКИ

до виконання практичних робіт з дисципліни

«Основи WEB-дизайну»

Міністерство освіти і науки України
Вінницький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до виконання практичних робіт з дисципліни

«Основи WEB-дизайну»

Вінниця
ВНТУ
2025

Рекомендовано до видання Радою з якості освіти Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 3 від 24.10.2024 р.)

Рецензенти:
О. В. Войцеховська, кандидат технічних наук, доцент
І. В. Богач, кандидат технічних наук, доцент

Методичні вказівки до виконання практичних робіт з дисципліни «Основи WEB-дизайну» [Електронний ресурс] / уклад. Р. Ю. Чехместрук. – Вінниця : ВНТУ, 2025. – 42 с.

У методичних вказівках наведено основні теоретичні дані до виконання самостійної роботи з дисципліни «Основи WEB-дизайну» та рекомендовану літературу. Методичні вказівки розроблено відповідно до навчальної програми дисципліни «Основи WEB-дизайну».

ЗМІСТ

Практична робота №1

РОЗРОБКА СТРУКТУРНОЇ СХЕМИ САЙТУ

Мета роботи: отримати базові поняття про структуру та розробку вебсайтів із застосуванням мови HTML.

ЗАГАЛЬНІ ВІДОМОСТІ

При розробці структури сайту варто визначити потрібну кількість сторінок і встановити зв'язки між ними. Існують лінійні, ієрархічні та довільні структури сайту.

Наприклад, якщо ви хочете послідовно відображати інформацію, таку як товари та послуги або матеріали підручників, рекомендовано використовувати лінійну структуру на вашому вебсайті. Відображення таких сайтів здійснюється послідовно від першої (головної) до останньої сторінки. Кожна сторінка містить посилання лише на одну, наступну сторінку сайту. Посилання на попередню сторінку також може бути прикріплене до сторінки для полегшення навігації по сайту.

В ієрархії створюється одна сторінка (головна сторінка), яка містить посилання на інші сторінки, що представляють розділи сайту. На інших сторінках є тільки одна попередня сторінка (рис. 1). Ієрархічна структура дозволяє кожній сторінці містити посилання на будь-яку кількість підсторінок. Така структура ідеально підходить для сайтів, що містять інформацію різної тематики, таких як каталоги, збірники статей на різні теми або добірки повідомлень.

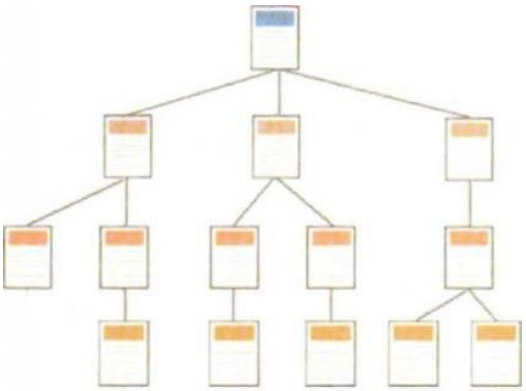

Рисунок 1 – Ієрархічна структура сайту

Найчастіше для створення сайту використовується довільна структура. При такій структурі сайту його сторінки пов'язані один з одним будь-яким чином (рис. 2). Для сайтів будь-якої структури ви можете вибрати лінійні або ієрархічні фрагменти.

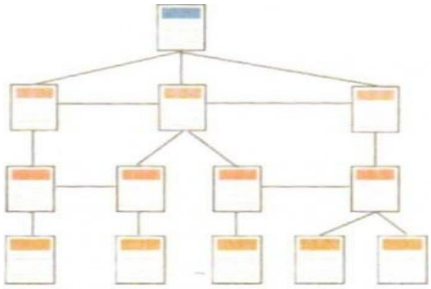

Рисунок 2 – Довільна структура сайту

У сайтах довільної структури можна виділити фрагменти, які є лінійними або ієрархічними. Прикладом довільної побудови структури сайту може бути Вікіпедія.

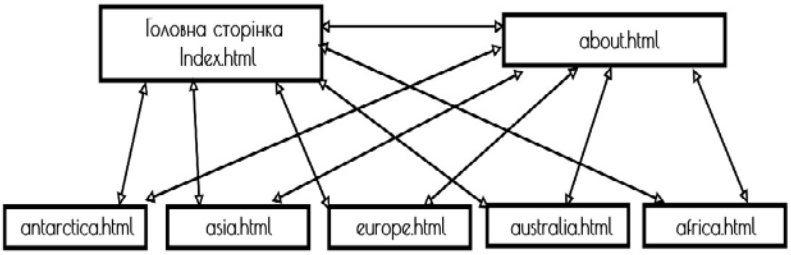

Рисунок 3 – Приклад направленого графу переходів
між сторінками сайту

Сайт може мати як статичну, так і динамічну реалізацію. Якщо у динамічній реалізації сторінки створюються у адмінпанелі та зберігаються у базі даних, то статична реалізація передбачає, що кожна сторінка буде представлена у вигляді HTML-файлів (рис. 4).

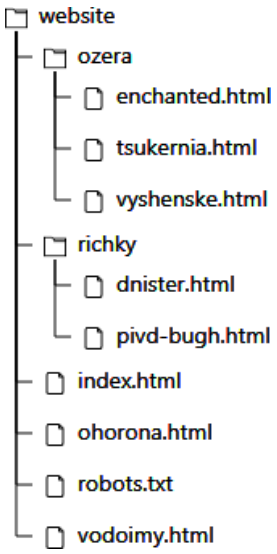

Рисунок 4 – Приклад організації HTML-файлів статичного сайту

У більшості вебсерверів файл головної сторінки сайту зазвичай має назву "index.html". Тобто для прикладу, якщо користувач зайде на сайт http://example.com, то у його браузері завантажиться сторінка по замовчуванню: https://example.com/index.html (можна перевірити обидва посилання).

Створення HTML-файлів відбувається за допомогою текстових редакторів. Створити HTML-файл можливо навіть за допомогою Блокноту Windows. Для цього потрібно ввести HTML-код у редактор та зберегти його у форматі HTML (рис. 5).

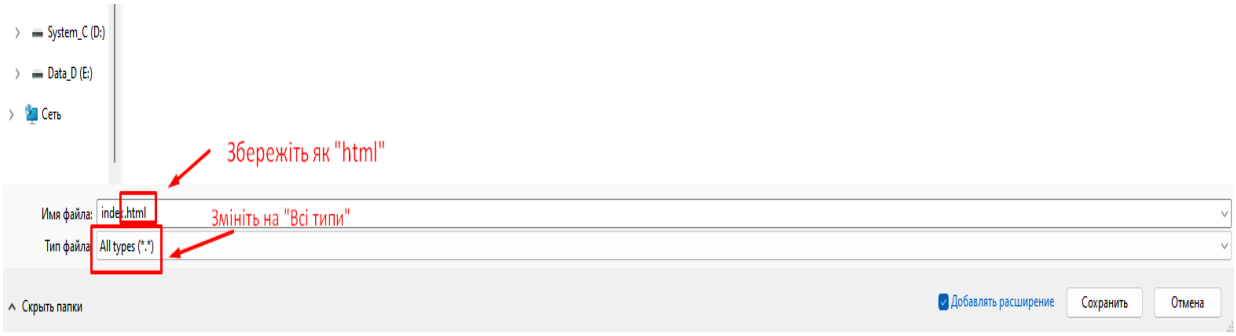

Рисунок 5 – Збереження тексту у форматі HTML-файлу

ЗАВДАННЯ

1. Ознайомитись з теоретичними відомостями щодо розробки структурної схеми сайту.

2. Описати розділи та підрозділи вебсайту.

3. Зобразити сторінки вебсайту та переходи між ними у вигляді направленого графу (типу як на рис. 3 та рис. 4). Кількість сторінок на сайті – не менше 6.

4. Зробити звіт за результатами виконання роботи та зробити висновки.

ЗМІСТ ЗВІТУ

1. Зазначити тему, мету і порядок роботи.

2. Описати список розділів вебсайту, що розробляється.

3. Граф сторінок та переходів між ними.

4. Висновки.

Практична робота №2

СТВОРЕННЯ ГРАФІЧНОГО ТА ТЕКСТОВОГО НАПОВНЕННЯ HTML-СТОРІНКИ

Мета роботи: отримати базові поняття про елементи мови розмітки HTML, навчитися створювати вебсторінки з контентом.

ЗАГАЛЬНІ ВІДОМОСТІ

HTML – це стандартна мова розмітки для вебсторінок в Інтернеті. Більшість вебсторінок створюються за допомогою мови HTML. HTML-сторінка обробляється браузером і відображається на екрані в звичному для користувачів вигляді.

Елементи HTML є основним компонентом мови розмітки HTML. Документ HTML складається з основних елементів html і доповнює його вміст іншими елементами. Кожен елемент має унікальну назву, вона пишеться латинськими літерами і не враховує регістр символів. В цілому, елемент складається з трьох компонентів:

1. Теги (початковий та кінцевий): `<p>TEXT</p>;`
2. Атрибути тегу (параметри): `<img src=”picture.jpg” />;`
3. Вміст (контент): `<center>Page middle</center>.`

Тег – це назва елемента, укладена в кутові дужки (< >). Атрибути задають технічну інформацію про елемент. Вміст елемента – це вся необхідна текстова і графічна інформація документа, яка відтворюється браузером на екрані.

Багато людей думають, що елемент – це тег (наприклад, "p-тег"). Однак варто пам'ятати, що елемент в основному складається з 2 компонентів (тегів і вмісту), а тег є складовою частиною елемента. Наприклад, елемент head завжди представлений у документі, навіть якщо обидва теги <head> та </head> у розмітці опущені.

Мова розмітки HTML є строго ієрархічною мовою, в якому кожен елемент має своє місце в ієрархії документа. Кожен елемент (крім тегу <html>) повинен бути розміщений всередині батьківського елемента. Давайте розглянемо приклад і кожен з його елементів:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
  </body>
```

```
</html>
```

Оголошення <!DOCTYPE html> визначає, що цей документ є документом HTML5. Елемент <html> є кореневим елементом HTML-сторінки. <head> елемент містить метадані про HTML-сторінку. Вони не видно користувачам, але пошукові системи та соціальні мережі можуть їх прочитати. Елемент <title> визначає заголовок HTML-сторінки (відображається в рядку заголовка або на вкладці сторінки браузера). <body> елементи визначають основну частину документа і є контейнерами всього видимого вмісту, такого як заголовки, абзаци, зображення, гіперпосилання, таблиці та списки. Елемент <h1> визначає заголовок. Елемент <p> визначає абзац.

Списки

Списки в HTML бувають трьох видів: упорядковані (ordered list) `<ol>`, невпорядковані (unordered list) `<ul>` і списки означень (definition list) `<dl>`. Елементи двох перших типів списків задаються тегом `<li>` (list item). Елементом списку може бути будь-який текст, картинки, абзаци, і, навіть, інші списки. Наприклад, впорядкований список задають так:

```
<ol>
  <li>Раз</li>
  <li>Два</li>
  <li>Три</li>
</ol>
```

Виглядає це таким чином:

1. Раз.
2. Два.
3. Три.

Таблиці

Таблиця (<table>) складається з рядків (<tr> – table row), кожен з яких також складається з клітинок (<td> – table data). А всередині клітинки може бути вже все що завгодно. Навіть ще одна таблиця. Виглядає це так:

Рядок 1 Стовпець 1	Рядок 1 Стовпець 2	Рядок 1 Стовпець 3
Рядок 2 Стовпець 1	Рядок 2 Стовпець 2	Рядок 2 Стовпець 3
Рядок 3 Стовпець 1	Рядок 3 Стовпець 2	Рядок 3 Стовпець 3

Код розмітки:

```
<table border="1">
  <tr>
    <td>Рядок 1 Стовпець 1</td>
    <td>Рядок 1 Стовпець 2</td>
    <td>Рядок 1 Стовпець 3</td>
  </tr>
  <tr>
```

```
    <td>Рядок 2 Стовпець 1</td>
    <td>Рядок 2 Стовпець 2</td>
    <td>Рядок 2 Стовпець 3</td>
  </tr>
  <tr>
    <td>Рядок 3 Стовпець 1</td>
    <td>Рядок 3 Стовпець 2</td>
    <td>Рядок 3 Стовпець 3</td>
  </tr>
</table>
```

По замовчуванню таблиці відображаються без меж, тобто межі невидимі. Це іноді корисно, але іноді потрібно, щоб межі було видно. Для цього задають значення атрибуту border. Він задає товщину меж таблиці. Якщо його значення нуль, то межі не відображаються.

Інколи, коли треба назвати стовпці, чи рядки, використовують клітинку заголовку.

Для цього замість тегу <td>пишуть <th>. Виглядає це так:

	Стовпець 1	**Стовпець 2**
Рядок 1	Рядок 1 Стовпець 1	Рядок 1 Стовпець 2
Рядок 2	Рядок 2 Стовпець 1	Рядок 2 Стовпець 2

Код розмітки наведеної таблиці:

```
<table border="1">
  <tr>
    <td></td>
    <th>Стовпець 1</th>
    <th>Стовпець 2</th>
  </tr>
  <tr>
    <th>Рядок 1</th>
    <td>Рядок 1 Стовпець 1</td>
    <td>Рядок 1 Стовпець 2</td>
  </tr>
  <tr>
    <th>Рядок 2</th>
    <td>Рядок 2 Стовпець 1</td>
    <td>Рядок 2 Стовпець 2</td>
  </tr>
</table>
```

Клітинки таблиці можна об'єднувати. Робиться це за допомогою атрибутів colspan і rowspan тегу <td>. Colspan вказує на скільки колонок буде пролягати дана клітинка, а rowspan на скільки рядків. Такий код:

```
<table border="1">
  <tr>
```

```
    <td colspan="2">Рядок 1 Стовпець 1 прос...ся на два
вправо</td>
    <td>Рядок 1 Стовпець 3</td>
  </tr>
  <tr>
    <td>Рядок 2 Стовпець 1</td>
    <td rowspan="3">Рядок 2 Стовпець 2 прос...ся на 2 вниз</td>
    <td>Рядок 2 Стовпець 3</td>
  </tr>
  <tr>
    <td>Рядок 3 Стовпець 1</td>
    <td>Рядок 3 Стовпець 3</td>
  </tr>
</table>
```

дає такий результат:

<table>
<tr><td colspan="2">Рядок 1 Стовпець 1 простягається на два вправо</td><td>Рядок 1
Стовпець 3</td></tr>
<tr><td>Рядок 2 Стовпець 1</td><td rowspan="2">Рядок 2 Стовпець 2 простягається на 2вниз</td><td>Рядок 2
Стовпець 3</td></tr>
<tr><td>Рядок 3 Стовпець 1</td><td>Рядок 3
Стовпець 3</td></tr>
</table>

Крім рядків таблиця може мати заголовок. Тег <caption> призначений для створення заголовка до таблиці і може розміщуватись тільки в середині тегу, причому зразу після відкриваючого тегу. Такий заголовок представляє собою текст по замовчуванню, відображений перед таблицею і описує її.

Також можна виділити рядки таблиці в групи з різним функціональним призначенням і призначити їм різні стилі. Можна виділити заголовкову частину <thead>, основну частину <tbody>і підсумок <tfoot>.

Зображення

Щоб вставити зображення в текст, використовуйте тег <img>. Його атрибут src задає джерело, тобто файл, що містить зображення. Тобто, коли ви пишете код, який відповідає останнім стандартам, він повинен мати такий вигляд: />.

Ви також можете змінити розмір зображення. Наприклад, якщо у вас невелике зображення, ви можете розтягнути його. Але воно виглядає дещо розмитим. Окрім зміни розміру вікна браузера, ви також можете змінити розмір зображення. Для цього вкажіть розмір у відсотках. Розміри задаються атрибутами width і height.

Приклад:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Картинки</title>
```

```
  </head>
  <body>
  <img      src="image.jpg"      alt="Маленька      картинка"
  title="Маленькакартинка" width="100" height="100" /><br/>
  <img      src="image.jpg"      alt="Велика      картинка"
  title="Великакартинка" width="400" height="400" /><br/>
  <img      src="image.jpg"      alt="Широка      картинка"
  title="Широка  картинка"    width="100%"    height="400"
  /><br/>
  </body>
</html>
```

Відео

HTML5 тег <video> дозволяє вставити відео, що програється засобами браузера. Шлях до відео файлу задається за допомогою вкладеного атрибуту <source>.

Приклад:

```
<video width="320px" height="240px" controls>
  <source src="videofile.mp4" type="video/mp4" />
</video>
```

При виконанні завдання з відео допускається вставляти відео з YouTube (замість використання тегу <video>). Для того, щоб вставити відео з YouTube, потрібно на сторінці ролику натиснути кнопку "Share" (поділитись), після чого обрати режим "Embed" (вставити).

ЗАВДАННЯ

1. Ознайомитися з тегами мови HTML.
2. Додати контент до вебсайту. Сайт повинен містити:
   - заголовки, параграфи, групування елементів, різні стилі тексту;
   - форму, кнопки;
   - список;
   - таблиці;
   - зображення та відео (можна вставити з YouTube).
3. Оформити звіт за результатами виконання роботи та зробити висновки.

БОНУСНІ ЗАВДАННЯ

Сторінка сайту має бути розміщена у Інтернеті та відкриватись за URL-посиланням.

ЗМІСТ ЗВІТУ

1. Тема, мета і порядок роботи.
2. Скріншот з однієї із сторінок сайту.
3. Висновки по роботі.

4. Вставити посилання на код. Код HTML можна розмістити наступним чином:

- розмістити на хостингу та розмістити посилання на нього (після висновків);
- залити на Git-репозиторій та розмістити посилання на нього (після висновків);
- якщо немає можливості зробити жоден варіант – приєднати код на наступній сторінці звіту після висновків.

Найбільш простий спосіб розмістити сайт (1) – скористатись платформою хмарної розробки Codiad у JetIQ. Потрапити туди можна із студентського персонального кабінету JetIQ: «Мої інструменти» – > «Web-проєкти». Доступ до Web-проєктів надається викладачем.

КОРИСНІ ПОСИЛАННЯ

1. HTML Tables – W3Schools. URL: https://www.w3schools.com/html/html_tables.asp (дата звернення: 21.08.2024).

2. HTML basics – Learn web development – MDN. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics (дата звернення: 21.08.2024).

3. HTML редактор-пісочниця TryIT – W3Schools. URL: https://www.w3schools.com/html/tryit.asp?filename=tryhtml_intro (дата звернення: 21.08.2024).

4. Introduction to HTML – W3Schools. URL: https://www.w3schools.com/html/html_intro.asp (дата звернення: 21.08.2024).

РОЗМІЩЕННЯ САЙТІВ

1. Tutorial: Configuring a static website using a custom domain registered with Route 53. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/website-hosting-custom-domain-walkthrough.html (дата звернення: 21.08.2024).

2. Github. URL: https://pages.github.com/ (дата звернення: 21.08.2024).

Практична робота №3

ВИКОРИСТАННЯ CSS НА ВЕБСАЙТІ

Мета роботи: отримати базові поняття про каскадні таблиці стилів CSS, навчитися створювати вебсторінки з використанням CSS.

ЗАГАЛЬНІ ВІДОМОСТІ

Каскадні таблиці стилів (англ. Cascading Style Sheets або css) – це спеціальна мова, яка використовується для відображення сторінок, написаних мовою розмітки даних. CSS найчастіше використовується для візуального відображення сторінок, написаних на HTML.

CSS використовується авторами та відвідувачами вебсторінок для визначення кольорів, шрифтів, макетів та інших аспектів зовнішнього вигляду сторінки. Однією з головних переваг є те, що можливість обміну вмістом сторінки (або вмістом) залежить від типу документа (описаного в CSS). CSS код може бути підключений до сторінки трьома способами:

1. Inline-стиль, за допомогою якого можна призначити стиль конкретному HTML-елементу безпосередньо в документі, за допомогою HTML-атрибуту style.

Приклад:

```
<img src="image.jpg" style="width: 400px; height: 350px;" />
```

2. Вбудувати стилі безпосередньо в HTML-документ (у вигляді блоку css коду) за допомогою елемента <style>. Даний елемент має розміщуватись в межах тегу head.

Приклад:

```
<style>
img {
width: 400px; height: 350px;
}
</style>
```

3. Застосувати зовнішні стилі (у вигляді окремого текстового css-файлу) за допомогою елемента <link>. Даний елемент має розміщуватись в межах тегу head.

Приклад:

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

Синтаксис CSS

Всі правила CSS складаються з селекторів і рекламних блоків (вони укладені в фігурні дужки (див. рисунок 6). У оголошеному блоці може бути один або більше оголошень (тобто всередині фігурних дужок), розділених крапкою з комою. Оголошення – це рядок, що складається з властивості css та його значення.

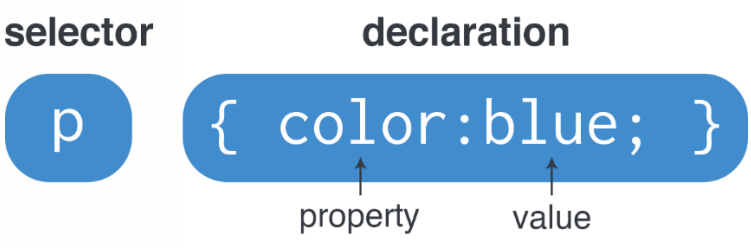

Рисунок 6 – Синтаксис CSS-правила

Різновиди селекторів

CSS-селектори можуть бути трьох типів. Основні з них такі:

1. *Селектори HTML-елементів*. Це найпростіший випадок – в якості селектора використовується ім'я html-елемента, який потрібно змінити. Наприклад, для тегу селектором буде strong. Відповідно, для тега <h1> селектором буде h1.

Приклад:

```
strong { font-weight: normal; color: red; }h1 { font: bold 10pt verdana; }
```

2. *ID селектори* – самий потужний тип селекторів. Це означає, що вони «витісняють» інші типи селекторів, а стилі, вказані з цим селектором, застосовуються першочергово. ID-селектор прив'язується до конкретного HTML-елементу з певним ID:

```
<div id="happy-lake">Елемент з ID</div>
```

Приклад CSS-коду:

```
#happy-lake { border: 1px solid blue; }
```

3. *Селектори класу*. Найбільш універсальний тип селектора. Можна додавати декілька класів (через пробіл) до HTML-елемента.

Приклад HTML-коду:

```
<div class="test-class">Елемент з класом</div>
```

Приклад CSS-коду:

```
.test-class { background-color: gray; }
```

ЗАВДАННЯ

1. Ознайомитися з основними стилями CSS та правилами застосування селекторів.
2. Додати стилі до вебсайту. Сайт повинен містити хоча б 3 класи з загальною кількістю характеристик не менше 10.
3. Оформити звіт за результатами виконання роботи та зробити висновки.
4. За допомогою стилів CSS застосувати стилі до спискового меню сайту таким чином:
   - прибрати маркери списку у меню;
   - розмістити всі елементи меню сайту у горизонтальному порядку;
   - зробити щоб елементи меню сайту реагували на наведення миші та натискання зміною кольору фону конкретного елементу.
5. Умова: код має бути розміщений онлайн і відкриватись за URL-посиланням.

ЗМІСТ ЗВІТУ

1. Тема, мета і порядок роботи.
2. Знімок екрану з однією із сторінок сайту.
3. Висновки по роботі.
4. URL-посилання на сторінку або HTML код (на наступній після висновків сторінці, якщо розмістити сторінку онлайн неможливо).

КОРИСНІ ПОСИЛАННЯ

1. Гайд по CSS селекторам. *DevZone*. URL: https://codeguida.com/post/69 (дата звернення: 21.08.2024).

2. Що таке CSS. *Ніжинський державний університет ім. М. Гоголя*. URL: http://www.ndu.edu.ua/liceum/html/teor_CSS.pdf (дата звернення: 21.08.2024).

Практична робота №4

ОСНОВИ FRONT-END ПРОГРАМУВАННЯ НА JAVASCRIPT

Мета роботи: ознайомитися з базовим синтаксисом та основними можливостями управління вмістом вебсторінки на стороні клієнта. Отримати практичні навички написання клієнтських скриптів з використанням мови JavaScript.

ЗАГАЛЬНІ ВІДОМОСТІ

JavaScript (також ECMAScript) — це мова програмування, яка стандартизована міжнародною організацією ECMA згідно із специфікацією ECMA-262, яка підтримується більшістю сучасних веббраузерів.

JavaScript має низку властивостей об'єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об'єктів в нім відрізняється від традиційних мов ООП. Крім того, JavaScript має ряд властивостей, властивих функціональним мовам: функції як об’єкти першого рівня, об’єкти як списки, каррування (currying), анонімні функції, замикання (closures), що додає мові додаткову гнучкість.

JavaScript має C-подібний синтаксис, але, в порівнянні з мовою Сі, має такі корінні відмінності:

1. Анонімні функції.
2. Автоматичне приведення типів.
3. Автоматичне прибирання сміття.
4. Функції як об'єкти першого класу.
5. Об'єкти з можливістю інтроспекції і динамічної зміни типу через механізм прототипів.
6. Обробка винятків.

Динамічний HTML

HTML – це лише мова розмітки, тому сам по собі він не передбачає можливості створення виконуваного коду. Під час розробки HTML я отримав розширення під назвою Dynamic HTML. Це дозволяє запускати на сторінці код JavaScript і VisualBasic (тільки в середовищі Internet Explorer). Таким чином, HTML-сторінка має можливість запускати програмні скрипти на стороні браузера.

Коли JavaScript-код інтегрований у вебсторінку, доступні можливості динамічного управління контентом. Це робиться за допомогою тегу <script>, який розміщується в потрібному місці на вебсторінці та виділяє початок та

кінець коду файл .js, завантажений для запуску із зовнішнього джерела. Якщо цей тег використовується в тексті документа (в тезі body), скрипт виконується при відображенні вебсторінки в браузері.

Приклад:

```
<!-- вставка скрипту в HTML-код -->
<script type="text/javascript">
// код скрипту
</script>
```

Приклад завантаження із зовнішнього джерела:

```
<!-- загрузка скрипту з файлу на сайті -->
<script type="text/javascript" src="http://example.com/script.js">
</script>
```

Приклади використання

Оскільки JavaScript у браузері насамперед задумувався як технологія динамічного оновлення вмісту сторінки, тому варто розглянути приклади, де динамічно змінюється контент.

Приклад виводу діалогу повідомлення:

```
alert("Hello JavaScript");
```

Приклад зміни вмісту HTML-елемента за його ID:

```
document.getElementById("demo").innerHTML = "Hello JavaScript";
```

Приклад зміни CSS-стилів елементу:

```
document.getElementById("demo").style.fontSize = "35px";
```

Приклад зміни видимості елементу:

```
document.getElementById("demo").style.display = "none"; // приховує
document.getElementById("demo").style.display = "block"; // показує
```

Приклади оголошення змінної:

```
var x = 5;var y = 6;
var z = "Hello JavaScript";
```

Приклад функції:

```
function myFunction(p1, p2) {
return p1 * p2; // Множить p1 на p2 і повертає результат
}
```

Приклад оголошення об’єкту:

```
const person = { firstName: "John",lastName: "Doe", age:   50, eyeColor: "blue"
};
```

ЗАВДАННЯ

1. Ознайомитися з особливостями синтаксису мови JavaScript.
2. Додати до сайту файл із скриптом, що змінює розмітку сторінки під час її перегляду у браузері (зміни при натисканні на кнопки, спливаючі вікна, перевірка завантажувальних файлів, онлайн конвертор, пасхалки, гра тощо).
3. Підключити створений JavaScript файл до існуючих сторінок сайту.
4. Оформити звіт за результатами виконання роботи та зробити висновки.

БОНУСНІ ЗАВДАННЯ

Умова: код має бути розміщений онлайн і відкриватись за URL-посиланням.

ЗМІСТ ЗВІТУ

1. Тема, мета і порядок роботи.
2. Знімок екрану з однією із сторінок сайту.
3. Висновки по роботі.
4. URL-посилання на сторінку або HTML код (на наступній сторінці після висновків, якщо розмістити сторінку онлайн неможливо).

КОРИСНІ ПОСИЛАННЯ

1. Синтаксис JavaScript. *W3Schools*. URL: https://www.w3schools.com/js/js_syntax.asp (дата звернення: 21.08.2024).

2. Змінні у JavaScript. *W3Schools*. URL: https://www.w3schools.com/js/js_variables.asp (дата звернення: 21.08.2024).

3. Оператори JavaScript. *W3Schools*. URL: https://www.w3schools.com/js/js_operators.asp (дата звернення: 21.08.2024).

4. Функції JavaScript. *W3Schools*. UPL: https://www.w3schools.com/js/js_functions.asp (дата звернення: 21.08.2024).

5. Арифметика у JavaScript. *W3Schools*. URL: https://www.w3schools.com/js/js_arithmetic.asp (дата звернення: 21.08.2024).

Практична робота №5

СТВОРЕННЯ ТА ПІДКЛЮЧЕННЯ ДО САЙТУ БАЗИ ДАНИХ

Мета роботи: отримання практичного досвіду створення та підключення бази даних до вебсайту, а також розуміння і обґрунтування вибору між SQL та NoSQL базами даних з урахуванням вимог конкретного проєкту. Крім того, робота спрямована на розвиток навичок програмування на мові JavaScript, розуміння архітектурних принципів веброзробки та вміння проводити тестування та аналіз різних технологій.

ЗАГАЛЬНІ ВІДОМОСТІ

Бази даних

База даних – це структурована колекція даних, яка зберігається та організовується таким чином, щоб забезпечити ефективний доступ, оновлення та аналіз даних.

Основні складові:

1. *Таблиці (Tables):* вони використовуються в реляційних базах даних для зберігання даних у вигляді рядків і стовпців.
2. *Структура (Schema):* визначає структуру бази даних, включаючи таблиці, поля та взаємозв'язки між ними.
3. *Запити (Queries):* використовуються для отримання, оновлення, видалення або вставки даних у базу даних.

Типи баз даних

Реляційні (SQL) бази даних:

1. Використовують табличну структуру для організації даних.
2. Запити здійснюються за допомогою мови SQL (Structured Query Language).
3. Приклади: PostgreSQL, MySQL, SQLite, Oracle.

NoSQL бази даних:

1. Використовують різні моделі для зберігання та організації даних, такі як документи, ключ-значення, колонки тощо.
2. Зазвичай масштабуються краще горизонтально та дозволяють більшу гнучкість у структурі даних.
3. Приклади: MongoDB, Cassandra, Redis, Couchbase.

Використання в веброзробці

Реляційні бази даних:

1. Зручні для використання в ситуаціях, коли структура даних стабільна та прогнозована.
2. Ідеально підходять для ситуацій, коли потрібна висока цілісність даних, наприклад, в банківській або фінансовій галузі.

NoSQL бази даних:

1. Зручні для ситуацій, коли структура даних може змінюватися та не є строго визначеною.
2. Ефективні для ситуацій, коли потрібно масштабувати систему горизонтально або коли вимоги до швидкодії великі.

Порівняння

Продуктивність:

1. Реляційні бази даних зазвичай мають більшу стабільність та доведену продуктивність.
2. NoSQL бази даних зазвичай дозволяють швидше масштабування та роботу з великими обсягами даних.

Гнучкість:

1. Реляційні бази даних вимагають строго визначеної структури даних.
2. NoSQL бази даних дають більшу гнучкість у відношенні до структури даних.

Цілісність:

1. Реляційні бази даних забезпечують високу цілісність даних через використання транзакцій та заборону дублювання даних.
2. NoSQL бази даних можуть мати меншу цілісність даних через відсутність транзакцій та меншу нормалізацію.

Враховуючи ці теоретичні відомості, вибір між реляційною та NoSQL базами даних залежить від конкретних вимог та характеристик проєкту.

ЗАВДАННЯ

1. Створення серверної частини з використанням Node.js:
   - створити вебсервер за допомогою Node.js;
   - підключити необхідні бібліотеки для роботи з базами даних (для PostgreSQL можна використовувати pg, а для MongoDB – mongodb).

- налаштувати маршрутизацію для роботи з базою даних (наприклад, CRUD операції);
- реалізувати методи для отримання, створення, оновлення та видалення даних на сервері.

2. Створення та підключення бази даних:
   - SQL (PostgreSQL):
     - створити базу даних PostgreSQL.
     - створити таблиці з відповідними полями для зберігання даних, які будуть використовуватися на вебсайті;
   - підключити серверну частину до бази даних PostgreSQL;
   - NoSQL (MongoDB):
     - встановити та налаштувати MongoDB;
     - створити колекції та документи для зберігання даних;
     - підключити серверну частину до MongoDB.

3. Тестування та порівняння:
   - написати тести для перевірки правильності роботи серверної частини;
   - порівняти працездатність, швидкість та легкість використання баз даних PostgreSQL та MongoDB;
   - скласти звіт із порівнянням обох баз даних, вказавши їх переваги та недоліки для даного проєкту.

ОБҐРУНТУВАННЯ ВИБОРУ БАЗИ ДАНИХ

Під час обґрунтування вибору між SQL та NoSQL базами даних, слід врахувати наступні фактори:

1. *Структура даних:* якщо дані мають чітку структуру з фіксованими полями, то реляційна база даних може бути кращим вибором. Якщо структура даних досить гнучка або неоднорідна, то NoSQL база даних може бути кращим рішенням.

2. *Масштабованість:* якщо передбачається значний обсяг даних та велика кількість операцій з ними, то NoSQL може бути кращим вибором, оскільки вона зазвичай краще масштабується горизонтально.

3. *Запити та операції:* якщо потрібно виконувати складні SQL-запити, такі як злиття або групування, то реляційна база даних може бути кращим варіантом. Якщо потрібно прості операції з даними, то NoSQL може бути ефективнішою альтернативою.

4. *Транзакції та цілісність даних:* реляційні бази даних зазвичай надають кращу підтримку для транзакцій та забезпечують цілісність даних. Якщо ці аспекти є критичними, то SQL може бути кращим варіантом.

КОРИСНІ ПОСИЛАННЯ

Основи баз даних

1. Безкоштовні курси SQL with TEEI+DataCamp. *DOU*. URL: https://dou.ua/forums/topic/41184/ (дата звернення: 21.08.2024).

2. Introduction to Databases. *Coursera*. URL: https://www.coursera.org (дата звернення: 21.08.2024).

Реляційні бази даних

1. Навчальний матеріал з SQLite легкого використання реляційної бази даних. *SQLite Tutorial*. URL: https://www.sqlitetutorial.net/ (дата звернення: 21.08.2024).

2. Підручник з MySQL. *W3Schools*. URL: https://www.w3schools.com/MySQL/default.asp (дата звернення: 21.08.2024).

3. PostgreSQL Tutorial. *W3Schools*. URL: https://www.w3schools.com/postgresql/index.php (дата звернення: 21.08.2024).

NoSQL бази даних

1. Безкоштовні курси від MongoDB, включаючи вступні та поглиблені матеріали. *MongoDB University*. URL: https://learn.mongodb.com/ (дата звернення: 21.08.2024).

Статті про порівняння та вибір баз даних

1. Саломан Дж. Як вибрати правильну базу даних для вашого проєкту. *DEV*. URL: https://dev.to/saloman_james/how-to-choose-the-right-database-for-your-project-1cl (дата звернення: 21.08.2024).

2. SQL проти NoSQL: 5 критичних відмінностей. *Integrate.io*. URL: https://www.integrate.io/blog/the-sql-vs-nosql-difference/ (дата звернення: 21.08.2024).

Практична робота №6

СТВОРЕННЯ САЙТУ З ПАРАЛАКС ЕФЕКТОМ

Мета роботи: полягає у розробці захопливого та інтерактивного вебсайту з використанням ефекту паралаксу для покращення користувацького досвіду та привертання уваги.

ЗАГАЛЬНІ ВІДОМОСТІ

Сайти з паралакс ефектом – це вебсайти, які використовують технологію паралаксу для створення враження глибини та руху об'єктів на екрані. Паралакс ефект відтворюється шляхом різниці у швидкості руху фонових та передніх об'єктів під час прокрутки вебсторінки. Це створює враження, ніби різні шари рухаються незалежно один від одного, що приносить новий рівень візуального враження для користувачів.

Основні технічні аспекти створення сайтів з паралакс ефектом включають використання CSS (Cascading Style Sheets), JavaScript і інших вебтехнологій. Для досягнення паралакс ефекту фонові та передні об'єкти, зазвичай, розділяються на різні шари, які рухаються з різною швидкістю під час прокрутки. Це може бути досягнуто шляхом зміни позиції об'єктів у момент прокрутки за допомогою CSS або JavaScript.

Одним з основних переваг використання паралакс ефекту на вебсайтах є здатність створювати захоплюючі та вражаючі візуальні ефекти, які залучають увагу користувачів і підвищують їхню взаємодію з вебсайтом. Це може бути особливо корисно для сайтів-портфоліо, рекламних сторінок, інтерактивних ігор та інших проєктів, де візуальний аспект має ключове значення.

Проте варто зазначити, що неправильне використання паралакс ефекту може призвести до погіршення досвіду користувача, особливо на мобільних пристроях або для користувачів з обмеженими можливостями. Тому важливо збалансувати ефективність і вражаючий вигляд з доступністю.

Для створення ефекту паралаксу за допомогою CSS можна використовувати різні підходи. Ось декілька основних CSS селекторів та властивостей, які можна використовувати:

1) *background-attachment*: fixed;: Цей селектор дозволяє закріпити фонове зображення на екрані, тобто воно залишатиметься на місці під час прокрутки сторінки.

```
.parallax-section {
    background-image: url('background.jpg');
    background-attachment: fixed;
    background-size: cover;
```

```
}
```

3) *transform*: translate();: Цей селектор дозволяє зміщувати елементи на екрані. Його можна використовувати для створення ефекту руху шарів під час прокрутки.

```
@keyframes parallax {
   from {
      transform: translate(0, -100%);
   }
   to {
      transform: translate(0, 100%);
   }
}
.parallax-layer {
   animation: parallax 10s infinite alternate;
}
```

3) *perspective*: Ця властивість визначає глибину перспективи елемента. Використовуйте її для створення тривимірного візуального ефекту.

```
.parallax-container {
   perspective: 100px;
}
.parallax-container {
   perspective-origin: 50% 50%;
}
```

4) *transform-style*: Ця властивість визначає, як дочірні елементи взаємодіють з перспективою відносно батьківського елемента.

```
.parallax-layer {
   transform-style: preserve-3d;
}
```

5) *opacity*: Ця властивість визначає прозорість елемента. Використовуйте її для створення ефекту прозорості шарів.

```
.parallax-layer {
   opacity: 0.8;
}
```

6) *z-index* визначає порядок накладання елементів на плоскості з від'ємним значенням, які знаходяться ближче до глядача, і елементів зі значенням більше за нуль, які знаходяться далі від глядача.

```
.parallax-layer-1 {
   z-index: 1;
}
```

```
.parallax-layer-2 {
   z-index: 2;
}
```

ЗАВДАННЯ

1. Вивчення основи HTML, CSS і JavaScript, які необхідні для створення ефекту паралакс.
2. Ознайомитися з різними способами реалізації ефекту паралаксу на вебсайті, включаючи використання CSS, JavaScript, бібліотек та фреймворків.
3. Створити вебсайт з ефектом паралаксу: слід розробити власний вебсайт з використанням ефекту паралаксу. Потрібно використовувати різні шари, анімації та зображення, щоб створити цікавий та захоплюючий дизайн. В кожного має бути індивідуальний сайт (банер, або сторінка).
4. Тестування та оптимізація: після створення вебсайту провести тестування, щоб переконатися, що ефект паралаксу працює на різних пристроях та браузерах.
5. Удосконалити швидкість завантаження та оптимізувати вебсайт для покращення користувацького досвіду.
6. Продемонструвати свій вебсайт та пояснити вибрані стратегії та техніки використання ефекту паралаксу.

КОРИСНІ ПОСИЛАННЯ

1. Creating a Parallax Website from Scratch. *Envato tuts+*. URL: https://webdesign.tutsplus.com/t/tutorials/search/Creating+a+Parallax+Website+from+Scratc (дата звернення: 21.08.2024).

2. Elementor Parallax Scrolling Effect Tutorial. URL: https://www.youtube.com/watch?v=gLO0X6yrKys (дата звернення: 21.08.2024).

3. How to Create a Parallax Scrolling Website. *W3Schools*. URL: https://www.w3schools.com/howto/howto_css_parallax.asp (дата звернення: 21.08.2024).

4. Pareh M. A Simple Parallax Scrolling Technique. *Envato tuts+*. URL: https://webdesign.tutsplus.com/a-simple-parallax-scrolling-technique--net-27641t (дата звернення: 21.08.2024).

5. Parallax Tutorial – Scrolling Effect using CSS and JavaScript. URL: https://www.youtube.com/watch?v=llv5kW4sz0U (дата звернення: 21.08.2024).

6. The Parallax Effect in Web Design: Create Depth and Dimension. *Envato tuts+*. URL: https://webdesign.tutsplus.com/t/tutorials/search/The+Parallax+Effect+in+Web+Design:+Create+Depth+and+Dimension (дата звернення: 21.08.2024).

Практична робота №7

АДАПТИВНИЙ ВЕБДИЗАЙН: ЗАБЕЗПЕЧЕННЯ ОПТИМАЛЬНОГО КОРИСТУВАЦЬКОГО ДОСВІДУ НА БУДЬ-ЯКИХ ПРИСТРОЯХ

Мета роботи: вивчення основ адаптивного вебдизайну та розробка вебсайту, який буде оптимально відображатися на різних пристроях (планшетах, смартфонах, настільних комп'ютерах тощо).

ЗАГАЛЬНІ ВІДОМОСТІ

Основні принципи адаптивного вебдизайну

Один із головних принципів адаптивного дизайну – це створення сайту таким чином, щоб він виглядав добре на будь-яких пристроях, незалежно від розміру їхнього екрану.

Використання відносних одиниць вимірювання (%, em, rem) для розмірів шрифтів, контейнерів, та інших елементів забезпечує гнучкість у розміщенні контенту.

Медіа-запити дозволяють змінювати стилі елементів в залежності від розміру екрану та інших параметрів.

Приклад:

```
/* Стилі для елементу при розширенні екрану більше 768px */
@media screen and (min-width: 768px) {
  .container {
    width: 80%;
  }
}
```

Використання медіа-запитів для адаптації до різних розмірів екранів

Медіа-запити дозволяють змінювати стилі елементів на основі характеристик екрану, таких як ширина та висота.

Це дозволяє стилізувати сайт для різних пристроїв, наприклад, смартфонів, планшетів, настільних комп'ютерів тощо.

Приклад:

```
/* Стилі для елементів при розширенні екрану менше 768px */
@media screen and (max-width: 768px) {
  .menu {
    display: none;
  }
}
```

Гнучкі сітки (flexbox та grid) для створення резинового макету

Flexbox та CSS Grid – це потужні інструменти для створення гнучких сіток, які адаптуються до різних розмірів екрану.

Вони дозволяють легко контролювати розташування та розміщення елементів у вебдокументі.

Приклад:

```
.container {
  display: flex;
  justify-content: space-between;
}
```

Зображення з різними роздільними здатностями та ретинізація

Для оптимального відображення зображень на різних пристроях із різною роздільною здатністю, рекомендується використовувати атрибут srcset для вказівки зображень різної роздільності.

Ретинізація дозволяє підвищити якість зображень на пристроях з високою роздільною здатністю (наприклад, Retina дисплеях).

Приклад:

<img src="small.jpg" srcset="medium.jpg 1000w, large.jpg 2000w" alt="Опис зображення">

ЗАВДАННЯ

1. Навігаційне меню, яке буде автоматично адаптуватися для зручного використання на різних пристроях.
2. Гнучкий сітковий макет, який буде автоматично змінювати свою структуру в залежності від розміру екрану.
3. Зображення, що будуть адаптовані для різних пристроїв, забезпечуючи оптимальне їх відображення.
4. Використання медіа-запитів для створення різних стилів для різних розмірів екранів.

КОРИСНІ ПОСИЛАННЯ

1. Can I Use. URL: https://caniuse.com/ (дата звернення: 21.08.2024).

2. Coyier C. CSS Flexbox Layout. *CSS-TRICKS*. URL: https://css-tricks.com/snippets/css/a-guide-to-flexbox/ (дата звернення: 21.08.2024).

3. LePage P., Andrew R. Responsive web design basics. *Web.dev*. URL: https://developers.google.com/web/fundamentals/design-and-ux/responsive (дата звернення: 21.08.2024).

4. Responsive Web Design Basics. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Responsive_Design (дата звернення: 21.08.2024).

5. Srcset & Sizes Explained. *Responsive Images*. URL: https://www.responsivebreakpoints.com/ (дата звернення: 21.08.2024).

6. Responsive Web Design – The Viewport. *W3schools. URL*: https://www.w3schools.com/css/css_rwd_viewport.asp (дата звернення: 21.08.2024).

Практична робота №8

ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ ПРОДУКТИВНОСТІ САЙТУ

Мета роботи: практична робота має на меті дослідити процес тестування та оптимізації продуктивності програмного забезпечення. Метою є оцінка ефективності програмного продукту, виявлення його слабких місць та впровадження заходів для покращення продуктивності.

ЗАГАЛЬНІ ВІДОМОСТІ

Тестування продуктивності – це процес визначення та оцінки різних аспектів продуктивності вебдодатку або вебсайту. Він включає перевірку швидкості завантаження сторінок, часу відповіді сервера, масштабованості системи під навантаження та інших важливих метрик продуктивності. Розглянемо кожен аспект детальніше:

1. *Швидкість завантаження сторінок*: це один з найважливіших аспектів продуктивності вебдодатку. Вимірюється час, необхідний для завантаження всіх елементів сторінки, включаючи HTML, CSS, JavaScript, зображення та інші ресурси. Для цього використовуються різні інструменти, такі як Chrome DevTools або Lighthouse.
2. *Час відповіді сервера*: це час, який сервер потребує для обробки запиту та надсилання відповіді назад клієнту. Повільна відповідь сервера може бути викликана перевантаженням сервера, неефективними запитами до бази даних або іншими причинами.
3. *Масштабованість*: це здатність вебдодатку або вебсайту працювати ефективно при збільшенні обсягу трафіку або навантаження. Масштабованість може бути протестована шляхом збільшення кількості одночасних користувачів або об'єму даних, які обробляються системою.
4. *Стійкість до навантаження*: це здатність вебдодатку чи вебсайту продовжувати працювати ефективно при високому рівні навантаження. Під час тестування продуктивності проводяться імітації великого потоку користувачів або інтенсивного використання додатку для визначення його меж продуктивності.
5. *Кешування та оптимізація ресурсів*: це стратегії, спрямовані на зменшення часу завантаження сторінок шляхом зберігання та використання кешованих копій ресурсів, таких як CSS, JavaScript, зображення тощо. Також включає в себе компресію ресурсів для зменшення їх розміру та мінімізацію кількості запитів до сервера.

Для ефективного тестування продуктивності вебдодатків використовуються спеціалізовані інструменти, такі як Apache JMeter, Gatling,

Siege, а також моніторинг реального часу для виявлення проблем при навантаженні в реальних умовах експлуатації. Після тестування результати аналізуються і виявлені проблеми виправляються для забезпечення оптимальної продуктивності вебдодатку.

У тестуванні продуктивності вебдодатків використовуються різні методики та підходи для забезпечення ефективності, надійності та оптимальної продуктивності. Найпоширеніші методи тестування продуктивності:

1. *Навантажувальне тестування (Load Testing*): цей метод використовується для оцінки роботи вебдодатку під навантаженням. Він імітує реальний трафік та одночасних користувачів, щоб визначити, як система веде себе при різних рівнях навантаження. Під час навантажувального тестування вимірюються час відповіді сервера, пропускна здатність мережі та інші метрики.
2. *Стрес-тестування (Stress Testing)*: цей метод оцінює межі та стійкість вебдодатку при надмірному навантаженні. Системі надається надзвичайно велика кількість запитів або інтенсивне навантаження для виявлення проблем, таких як перевантаження сервера, вичерпання ресурсів або падіння продуктивності.
3. *Шкальованість тестування (Scalability Testing)*: цей метод досліджує, як вебдодаток масштабується при збільшенні обсягу трафіку або користувачів. Тестується здатність системи ефективно працювати при зміні масштабу інфраструктури, такої як додавання нових серверів або збільшення ресурсів.
4. *Стійкість до витривалості (Endurance Testing)*: цей метод випробовує стабільність вебдодатку протягом тривалого періоду часу. Системі надається постійне навантаження протягом довгого часу для виявлення проблем, таких як витоки ресурсів або накопичення помилок з часом.
5. *Конфігураційне тестування (Configuration Testing)*: цей метод тестує продуктивність вебдодатку при різних конфігураціях серверів, мереж та програмного забезпечення. Це допомагає виявити, які конфігурації є оптимальними для певного вебдодатку та його вимог.
6. *Тестування резервного копіювання (Failover Testing)*: цей метод перевіряє, як швидко та ефективно вебдодаток може відновити свою працездатність після виникнення аварійної ситуації. Випробовується реакція системи на відмову серверів або інших системних неполадок.

Ці методи часто використовуються в поєднанні один з одним для створення повноцінного тестового покриття, яке допомагає забезпечити оптимальну продуктивність та надійність вебдодатків.

Оптимізація продуктивності вебдодатків є критичним аспектом розробки, оскільки вона впливає на користувацький досвід, конверсію та загальний успіх проєкту.

Ключові аспекти оптимізації продуктивності вебдодатків:

1. *Швидкість завантаження сторінок*: одним з найважливіших аспектів оптимізації є зменшення часу завантаження сторінок. Це можна досягти за допомогою різних технік, таких як компресія ресурсів (зображення, CSS, JavaScript), кешування, мінімізація та об'єднання файлів, використання Content Delivery Network (CDN) та оптимізація серверного та клієнтського коду.
2. *Оптимізація запитів до сервера*: зменшення кількості та розміру запитів до сервера може значно підвищити продуктивність. Це включає в себе використання кешування на стороні клієнта, зменшення кількості HTTP-запитів, асинхронне завантаження ресурсів та оптимізацію запитів до бази даних.
3. *Кешування*: ефективне використання кешування дозволяє зберігати копії ресурсів на клієнтських та серверних рівнях для швидкого доступу. Це може бути досягнуто за допомогою HTTP-кешування, кешування на рівні даних (наприклад, кешування результатів запитів до бази даних), а також використання проміжних кешів, таких як Redis або Memcached.
4. *Оптимізація зображень та мультимедіа*: зменшення розміру та оптимізація зображень і мультимедійних файлів може значно підвищити швидкість завантаження сторінок. Це може бути досягнуто за допомогою компресії зображень, видалення непотрібних метаданих, використання форматів зображень з втратами або без них в залежності від контексту, а також використання lazy loading для великих мультимедійних файлів.
5. *Оптимізація CSS та JavaScript*: мінімізація та об'єднання CSS- та JavaScript-файлів, видалення непотрібного коду, використання асинхронного завантаження та видалення блокуючих скриптів можуть значно покращити продуктивність вебдодатків.
6. *Моніторинг та оптимізація бази даних*: ефективне використання бази даних є важливим аспектом оптимізації продуктивності. Це включає в себе використання індексів, оптимізацію запитів, видалення зайвих даних, реплікацію та шардування для розподілу навантаження.
7. *Використання кешування на рівні клієнта*: можливість кешування даних на браузері клієнта може значно покращити продуктивність вебдодатків. Це може бути досягнуто за допомогою технік, таких як локальне сховище, sessionStorage, IndexedDB та використання Service Workers для офлайн доступу.
8. *Оптимізація для мобільних пристроїв*: з огляду на зростаючу кількість користувачів мобільних пристроїв, важливо оптимізувати вебдодатки для мобільних платформ. Це включає в себе використання респонсивного дизайну, оптимізацію для швидкості мобільних мереж, обмеження використання ресурсів та оптимізацію для різних браузерів.

ЗАВДАННЯ

1. Збір вихідних даних: почніть з аналізу вебсайту для збору вихідних даних про його продуктивність. Це може включати час завантаження сторінок, кількість запитів до сервера, розмір сторінок тощо.
2. Використання інструментів тестування: використовуйте різноманітні інструменти для тестування продуктивності, такі як Google PageSpeed Insights, GTmetrix або Pingdom. Ці інструменти надають детальну інформацію про різні аспекти продуктивності сайту.
3. Виконання тестів продуктивності: запустіть навантажувальні тести згідно з розробленим набором сценаріїв та збережіть результати.
4. Аналіз результатів тестування: ретельно проаналізуйте результати тестування, звертаючи особливу увагу на області, де є можливість для поліпшення. Це може бути оптимізація розміру зображень, зменшення кількості запитів до сервера, кешування ресурсів та інші.
5. Впровадження оптимізацій: на основі аналізу результатів тестування розробіть план оптимізації, який включатиме конкретні кроки для поліпшення продуктивності. Це може включати зміну коду, налаштування сервера, використання кешування та інші техніки.
6. Перевірка ефективності оптимізацій: після впровадження оптимізацій проведіть знову тестування, щоб переконатися, що вони дійсно покращили продуктивність сайту. Порівняйте нові результати з початковими, щоб визначити ефективність ваших зусиль.
7. Постійний моніторинг: встановіть постійний моніторинг та періодично виконуйте тестування продуктивності для забезпечення оптимальної роботи сайту. Продуктивність вебсайту може змінюватися з часом через зростання вмісту, зміни в програмному забезпеченні або інші фактори.

КОРИСНІ ПОСИЛАННЯ

1. Killelea P. Web Performance Tuning. 2-ге вид. O'Reilly Media, 2002. 480 с.

2. Molyneaux I. The Art of Application Performance Testing. 2-ге вид. O'Reilly Media, 2014. 257 с.

3. Performance Testing Guidance for Web Applications: patterns and practices. Microsoft Corporation, 2007. P. 221. URL: http://download.51testing.com/ddimg/uploadsoft/20101206/PerfTestGuide.pdf (дата звернення: 21.08.2024).

Практична робота №9

ОСНОВИ SEO

Мета роботи: ознайомитись із базовими поняттями та процесами SEO (пошукова оптимізація), створити семантичне ядро, обрати ключові слова, та налаштувати базові SEO параметри для вебсайту.

ЗАГАЛЬНІ ВІДОМОСТІ

SEO (Search Engine Optimization, пошукова оптимізація) – це комплекс заходів, спрямованих на покращення позицій сайту у результатах пошукових систем за певними запитами. Мета SEO – збільшити органічний трафік на сайт, покращивши його видимість у пошукових системах.

Види SEO

On-page SEO: оптимізація внутрішніх елементів сайту, таких як контент, HTML-теги, структура URL та інші технічні параметри.

Off-page SEO: оптимізація зовнішніх факторів, які впливають на ранжування сайту, таких як зворотні посилання (backlinks), соціальні сигнали та інші зовнішні джерела.

Семантичне ядро – це список ключових слів та фраз, які відображають тематику сайту і за якими сайт буде оптимізований. Воно є основою для створення та структурування контенту на сайті.

Важливість семантичного ядра досить суттєва.

Релевантність контенту: допомагає створити контент, який відповідає запитам користувачів.

Структура сайту: сприяє побудові логічної структури сайту з урахуванням ключових тем.

Підвищення видимості: забезпечує покращення видимості сайту у пошукових системах за різними запитами.

Ключові слова

Ключові слова (keywords) – це слова або фрази, які користувачі вводять у пошукові системи для знаходження потрібної інформації. Вони є основою для створення семантичного ядра.

Типи ключових слів

Головні ключові слова (head keywords): короткі загальні слова або фрази з високою частотністю запитів і конкуренцією (наприклад, «купити телефон»).

Довгі хвости (long-tail keywords): більш конкретні фрази з меншою частотністю запитів і конкуренцією, але більш цільові (наприклад, «купити смартфон Samsung Galaxy S21 в Києві»).

Середні ключові слова (middle-tail keywords): фрази, що знаходяться між головними ключовими словами і довгими хвостами за довжиною і специфічністю.

Вибір ключових слів

Частотність запитів: як часто користувачі шукають це слово або фразу.

Конкуренція: скільки інших сайтів оптимізовано за цим словом.

Релевантність: наскільки ключове слово відповідає змісту вашого сайту та очікуванням користувачів.

Мета-теги

Мета-теги – це елементи HTML-коду сторінки, які надають інформацію про сторінку пошуковим системам і користувачам.

Основні мета-теги:

Title (заголовок): короткий опис змісту сторінки (до 60 символів), що відображається у заголовку вікна браузера та у результатах пошуку.

Meta description (опис): короткий опис змісту сторінки (до 160 символів), що відображається у результатах пошуку під заголовком.

Meta keywords: список ключових слів (не використовується більшістю сучасних пошукових систем, але може бути корисним для внутрішнього SEO).

Базові SEO параметри

Robots.txt: файл, який інструктує пошукові системи, які сторінки сайту вони можуть індексувати.

Sitemap.xml: XML-файл, що містить список всіх сторінок сайту, які повинні бути індексовані пошуковими системами.

Alt-теги для зображень: текстові описи зображень, які допомагають пошуковим системам зрозуміти вміст зображень та покращити SEO.

Запуск первинної SEO-кампанії

Індексація сайту: процес додавання сайту до бази даних пошукових систем. Перший крок після запуску сайту.

Моніторинг позицій та трафіку: регулярний аналіз позицій сайту у результатах пошуку та обсягу органічного трафіку з метою коригування SEO-стратегії.

Оновлення контенту: регулярне додавання нових статей і оновлення існуючого контенту для підтримання актуальності та релевантності сайту.

SEO – це постійний процес, що вимагає регулярного моніторингу та коригування стратегії у відповідь на зміни алгоритмів пошукових систем та поведінки користувачів.

ЗАВДАННЯ

1. Визначення цільової аудиторії.

Перед початком роботи з SEO важливо зрозуміти, хто ваша цільова аудиторія. Для цього визначте:

- вік, стать, місце проживання, інтереси;
- які проблеми вирішує ваш продукт/послуга для цієї аудиторії.

2. Аналіз конкурентів.

Аналіз конкурентів допоможе зрозуміти, які ключові слова використовують вони і які стратегії SEO є ефективними у вашій ніші:

- виберіть 3-5 основних конкурентів;
- використовуйте інструменти для аналізу SEO (наприклад, Ahrefs, SEMrush, Moz) для збору інформації про їхні ключові слова та трафік;
- проаналізуйте, які з їхніх ключових слів мають найбільшу кількість трафіку та які сторінки є найбільш популярними.

3. Створення семантичного ядра:

- використовуйте інструменти для підбору ключових слів (Google Keyword Planner, Ahrefs, SEMrush, Ubersuggest);
- зберіть всі можливі ключові слова, що відносяться до вашого бізнесу;
- розділіть ключові слова на тематичні групи (кластеризація).

4. Вибір ключових слів.

З великого списку ключових слів оберіть ті, які будуть найефективнішими для вашого сайту:

- враховуйте такі критерії: частотність пошуку, конкуренція, релевантність;
- виберіть основні ключові слова для головних сторінок сайту;
- виберіть довгі хвости (long-tail keywords) для блогу чи новинних сторінок.

5. Написання мета-тегів для сторінок.

Мета-теги допомагають пошуковим системам зрозуміти зміст сторінки:

- створіть унікальний та описовий title (заголовок) для кожної сторінки (до 60 символів);
- напишіть meta description (опис) для кожної сторінки (до 160 символів), що включає ключові слова;
- використовуйте ключові слова у заголовках (h1, h2, h3) та тексті сторінок.

6. Налаштування базових SEO параметрів на вебсайті:

- перевірте налаштування Robots.txt:
  - Robots.txt - це файл, який інструктує пошукові системи, які сторінки сайту вони можуть або не можуть індексувати;
- створіть файл Robots.txt у кореневій директорії вашого сайту;
- приклад файлу Robots.txt:

```
User-agent: *;
Disallow: /admin/;
Disallow: /login/;
Allow: /;
Sitemap: https://www.yoursite.com/sitemap.xml;
```

- створіть та налаштуйте Sitemap.xml:
  - Sitemap.xml - це XML-файл, що містить список всіх сторінок сайту, які повинні бути індексовані пошуковими системами;
- використовуйте інструменти для генерації Sitemap.xml (наприклад, XML Sitemap Generator);
- приклад файлу Sitemap.xml:

```
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>https://www.yoursite.com/</loc>
    <lastmod>2024-05-22</lastmod>
    <changefreq>monthly</changefreq>
    <priority>1.0</priority>
  </url>
  <url>
    <loc>https://www.yoursite.com/about</loc>
    <lastmod>2024-05-20</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.8</priority>
  </url>
</urlset>
```

7. Створення та налаштування Robots.txt та Sitemap.xml.

Створення файлу Robots.txt:

- створіть файл Robots.txt у текстовому редакторі (наприклад, Notepad);
- внесіть необхідні інструкції:

- User-agent означає, що правила застосовуються до всіх пошукових систем;

- Disallow: /admin/ забороняє індексацію директорії /admin/;

- Allow: / дозволяє індексацію всіх інших сторінок;

- Sitemap: https://www.yoursite.com/sitemap.xml вказує на розташування файлу Sitemap.xml;

- збережіть файл і завантажте його в кореневу директорію вашого сайту (наприклад, https://ww;

- w.yoursite.com/robots.txt).

Створення файлу Sitemap.xml:

- використовуйте інструменти для генерації Sitemap.xml (наприклад, XML Sitemap Generator, Yoast SEO для WordPress);

- створіть структуру файлу:

- <urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"> початок файлу Sitemap.xml;

- <url> блок для кожної URL сторінки;

- <loc> містить повну URL сторінки;

- <lastmod> вказує дату останньої модифікації сторінки;

- <changefreq> вказує частоту зміни сторінки (наприклад, monthly, weekly);

- <priority> встановлює пріоритет сторінки (від 0.0 до 1.0);

- збережіть файл і завантажте його в кореневу директорію вашого сайту (наприклад, https://www.yoursite.com/sitemap.xml).

8. Запуск первинної SEO-кампанії

- індексація сайту:

- відправте сайт для індексації у Google Search Console;

- додайте URL вашого сайту в розділ «Інструменти для вебмайстрів» та завантажте файл Sitemap.xml;

- моніторинг позицій та трафіку:

- використовуйте Google Analytics для відстеження трафіку;

- використовуйте Google Search Console для моніторингу індексації та позицій у пошукових результатах;

- використовуйте інструменти для моніторингу позицій (наприклад, Ahrefs, SEMrush) для відстеження змін позицій;

- оновлення контенту:

- регулярно додавайте нові статті з ключовими словами;

- оновлюйте існуючий контент, щоб зберегти його актуальність;

- використовуйте аналітичні дані для оптимізації контенту відповідно до поведінки користувачів.

КОРИСНІ ПОСИЛАННЯ

1. Bailyn E. SEO Made Easy: Everything You Need to Know About SEO and Nothing More. 2-nd edition. Pearson Education, 2013. P. 312.

2. Clarke A. SEO 2022: Learn Search Engine Optimization with Smart Internet Marketing Strategies. CreateSpace Independent Publishing Platform, 2022. P. 298.

3. Clarke A. SEO 2023: Learn Search Engine Optimization with Smart Internet Marketing Strategies. CreateSpace Independent Publishing Platform, 2023. P. 322.

4. Clay B. Search Engine Optimization All-in-One For Dummies. 4-nd edition. For Dummies, 2015. P. 800.

5. Ducharme R. Advanced SEO: Search Engine Optimization Secrets That Will Increase Your Search Engine Traffic. CreateSpace Independent Publishing Platform, 2015. P. 152.

6. Enge E., Spencer S., Stricchiola J. The Art of SEO: Mastering Search Engine Optimization. 3-nd edition. O'Reilly Media, 2015. P. 994.

7. French G., Ward E. Ultimate Guide to Link Building: How to Build Website Authority, Increase Traffic and Search Ranking with Backlinks. 2-nd edition. Entrepreneur Press, 2020. P. 304.

8. Kent P. SEO for Dummies. 7-nd edition. For Dummies, 2019. P. 512.

9. McDonald J. SEO Fitness Workbook: 2023 Edition: The Seven Steps to Search Engine Optimization Success on Google. CreateSpace Independent Publishing Platform, 2023. P. 450.

10. The Beginner's Guide to SEO. *MOZ*. URL: https://moz.com/beginners-guide-to-seo (дата звернення: 21.08.2024).

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Безкоштовні курси SQL with TEEL+DataCamp. *DOU*. URL: https://dou.ua/forums/topic/41184/ (дата звернення: 21.08.2024).

2. Гайд по CSS селекторам. *DevZone*. URL: https://codeguida.com/post/69 (дата звернення: 21.08.2024).

3. Найкращі курси з теми Databases – вивчайте Databases онлайн. *Coursera*. URL: https://www.coursera.org (дата звернення: 21.08.2024).

4. Що таке CSS. Ніжинський державний університет ім. М. Гоголя. URL: http://www.ndu.edu.ua/liceum/html/teor_CSS.pdf (дата звернення: 21.08.2024).

5. A Simple Parallax Scrolling Technique. *Envato tuts+*. URL: https://webdesign.tutsplus.com/a-simple-parallax-scrolling-technique--net-27641t (дата звернення: 21.08.2024).

6. Bailyn E. SEO Made Easy: Everything You Need to Know About SEO and Nothing More. 2-nd edition. Pearson Education, 2013. P. 312.

7. Can I Use : офіц. вебсайт. URL: https://caniuse.com/ (дата звернення: 21.08.2024).

8. Cardello J. Parallax effects create depth and dimensionality, making static pixels come to life. *Webflow*. URL: https://webflow.com/blog/parallax-scrolling (дата звернення: 21.08.2024).

9. Clarke A. SEO 2023: Learn Search Engine Optimization with Smart Internet Marketing Strategies. CreateSpace Independent Publishing Platform, 2023. P. 322.

10. Clarke A. SEO 2022: Learn Search Engine Optimization with Smart Internet Marketing Strategies. CreateSpace Independent Publishing Platform, 2022. P. 298.

11. Clay B. Search Engine Optimization All-in-One For Dummies. – 4-nd edition. For Dummies, 2015. P. 800.

12. Creating a Parallax Website from Scratch. *Envato tuts+*. URL: https://webdesign.tutsplus.com/t/tutorials/search/Creatsng+a+Parallax+Website+from+Scratch (дата звернення: 21.08.2024).

13. CSS Flexbox Layout Guide. *CSS-Tricks*. URL: https://css-tricks.com/snippets/css/a-guide-to-flexbox/ (дата звернення: 21.08.2024).

14. Ducharme R. Advanced SEO: Search Engine Optimization Secrets That Will Increase Your Search Engine Traffic. CreateSpace Independent Publishing Platform, 2015. P. 152 с.

15. Enge E., Spencer S., Stricchiola J. The Art of SEO: Mastering Search Engine Optimization. 3-nd edition. O'Reilly Media, 2015. P. 994.

16. French G., Ward E. Ultimate Guide to Link Building: How to Build Website Authority, Increase Traffic and Search Ranking with Backlinks. 2-nd edition. Entrepreneur Press, 2020. P. 304.

17. How To – Parallax Scrolling. *W3school*. URL: https://www.w3schools.com/howto/howto_css_parallax.asp (дата звернення: 21.08.2024).

18. HTML Introduction. *W3Schools*. URL: https://www.w3schools.com/html/html_intro.asp (дата звернення: 21.08.2024).

19. HTML Tables. *W3Schools*. URL: https://www.w3schools.com/html/html_tables.asp (дата звернення: 21.08.2024).

20. HTML: Creating the cjntent. *W3Schools*. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics (дата звернення: 21.08.2024).

21. JavaScript Syntax. *W3Schools*. URL: https://www.w3schools.com/js/js_syntax.asp (дата звернення: 21.08.2024).

22. JavaScript Variables. *W3Schools*. *URL*: https://www.w3schools.com/js/js_variables.asp (дата звернення: 21.08.2024).

23. JavaScript Operators. *W3Schools*. *URL*: https://www.w3schools.com/js/js_operators.asp (дата звернення: 21.08.2024).

24. JavaScript Functions. *W3Schools*. *URL*: https://www.w3schools.com/js/js_functions.asp (дата звернення: 21.08.2024).

25. JavaScript Arithmetic. *W3Schools*. URL: https://www.w3schools.com/js/js_arithmetic.asp (дата звернення: 21.08.2024).

26. Kent P. SEO for Dummies. 7-nd edition. For Dummies, 2019. P. 512.

27. Killelea, P. Web Performance Tuning. 2-nd edition. O'Reilly Media, 2002. P. 480.

28. LePage P., Andrew R. Responsive Web Design Basics. *Web.dev*. URL: https://web.dev/articles/responsive-web-design-basics (дата звернення: 21.08.2024).

29. McDonald J. SEO Fitness Workbook: 2023 Edition: The Seven Steps to Search Engine Optimization Success on Google. CreateSpace Independent Publishing Platform, 2023. P. 450.

30. Molyneaux I. The Art of Application Performance Testing. 2-nd edition. O'Reilly Media, 2014. P. 257.

31. MongoDB University : офіц. вебсайт. URL: https://learn.mongodb.com/ (дата звернення: 21.08.2024).

32. MySQL Tutorial. *W3Schools*. URL: https://www.w3schools.com/MySQL/default.asp (дата звернення: 21.08.2024).

33. Parallax Tutorial – Scrolling Effect using CSS and JavaScript. *YouTube*. URL: https://www.youtube.com/watch?v=llv5kW4sz0U (дата звернення: 21.08.2024).

34. Performance Testing Guidance for Web Applications. *Learn*. URL: https://learn.microsoft.com/en-us/previous-versions/msp-n-p/bb924375(v=pandp.10) (дата звернення: 21.08.2024).

35. PostgreSQL Tutorial. *W3Schools*. URL: https://www.w3schools.com/postgresql/index.php (дата звернення: 21.08.2024).

36. Responsive Images Breakpoints Generator : офіц. вебсайт. URL: https://www.responsivebreakpoints.com/ (дата звернення: 21.08.2024).

37. Responsive Web Design – The Viewport. *W3school*. URL: https://www.w3schools.com/css/css_rwd_viewport.asp (дата звернення: 21.08.2024).

38. Responsive Web Design Fundamentals. *Class Central*. URL: https://www.classcentral.com/course/udacity-responsive-web-design-fundamentals-3255 (дата звернення: 21.08.2024).

39. Saloman J. How to Choose the Right Database for Your Project. *DEV*. URL: https://dev.to/saloman_james/how-to-choose-the-right-database-for-your-project-1cl (дата звернення: 21.08.2024).

40. SQLite Tutorial : офіц. вебсайт. URL: https://www.sqlitetutorial.net/ (дата звернення: 21.08.2024).

41. SQL vs NoSQL: 5 Critical Differences. *Integrate.io*. URL: https://www.integrate.io/blog/the-sql-vs-nosql-difference/ (дата звернення: 21.08.2024).

42. The Beginner's Guide to SEO. *MOZ*. URL: https://moz.com/beginners-guide-to-seo (дата звернення: 21.08.2024).

43. TryIT: HTML Introduction. *W3Schools*. URL: https://www.w3schools.com/html/tryit.asp?filename=tryhtml_intro (дата звернення: 21.08.2024).

Електронне навчальне видання

Чехместрук Роман Юрійович

Методичні вказівки до виконання практичних робіт з дисципліни «Основи WEB-дизайну»

Рукопис оформив *Р. Чехместрук*

Редактор *О. Малетіна*

Оригінал-макет виготовлено в РВВ ВНТУ

Підписано до видання 31.01.2025
Гарнітура Times New Roman.
Зам. № Р2025-029

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
press.vntu.edu.ua;
Email: rvv.vntu@gmail.com
Свідоцтво суб'єкта видавничої справи серія
ДК № 3516 від 01.07.2009 р.