**Практична робота 2**

**Тема:** **Підключення зовнішніх скриптів JavaScript та послідовність їх виконання в HTML-документі.**

**Мета:** Навчитися підключати зовнішні скрипти JavaScript. Ознайомитися з поняттям асинхронних скриптів JavaScript. Навчитися використовувати атрибути defer/async. Вміти використовувати спливаючі діалогові вікна введення/виведення та виведення даних у консоль браузера.

**Питання для повторення:**

1. Абсолютні та відносні адреси в HTML-документі.
2. Призначення елементу <base> в HTML-документі.
3. Призначення управляючого метасимвола (metacharacter) ASCII \t та \n в документі HTML.
4. Призначення атрибуту мета-тегу – name = 'viewport' та meta name = "apple-mobile-web-app-capable" content = "yes"

**Завдання:**

Підключити зовнішні скрипти JavaScript до HTML-документа.

**Хід роботи:**

1. Підключення зовнішніх скриптів JavaScript до HTML-документа
   1. В довільному текстовому редакторі створіть HTML-документ код якого наведено нижче та збережіть його в паці з ім’ям 02-JS01:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name='viewport' content='width=device-width, initial-scale=1.0'>

<meta name="apple-mobile-web-app-capable" content="yes"/>

<!-- <link href='http://fonts.googleapis.com/css?family=Open+Sans' rel='stylesheet' type='text/css'> -->

<link href="styles/style.css" rel="stylesheet" type="text/css">

<title>Лабораторна\_02-JS01</title>

</head>

<body>

<h1 class="myclass n1">Mozilla - це класно!!!</h1>

<p>Важлива інформація не буде відображена, поки не завантажиться скрипт.</p>

<p>...Важлива інформація!</p>

<div class="temp m1">Скількі буде у Java Script 0.1 + 0.2 = ?</div>

<script src="https://js.cx/hello/ads.js?speed=0"></script>

<h1 class="myclass n2"></h1>

<img src="images/firefox-icon.png" alt="The Firefox logo: a flaming fox surrounding the Earth." style="width:200px">

<div class="temp m0"></div>

<script >

console.log(0.1 + 0.2);

var sum = 0.1 + 0.2;

var result = document.getElementsByClassName("m1")[0];

result.innerHTML = "В Java Script " + "<br>"+ " 0.1 + 0.2 = " + sum;

result.style.color = '#2d264a';

</script>

<p> Mozilla являє собою глобальне товариство:</p>

<ul>

<li>розробників;</li>

<li>дизайнерів;</li>

<li>мислителів.</li>

</ul>

<p>Всі вони працють разом, щоб зробити Інтернет інтерактивним та доступним для всіх людей. </p>

<p> Прочитайте маніфест розробників <a href="https://www.mozilla.org/en-US/about/manifesto/">Mozilla</a>,

щоб дізнатися про цінності та принципи, які погладені в основу цієї місії.</p>

<script src="scripts/prompt.js"></script>

<script src="scripts/alert.js"></script>

<script src="scripts/confirm.js"></script>

</body>

</html>

* 1. В паці з ім’ям 02-JS01 створіть вкладену папку з ім’ям scripts з скриптами prompt.js, alert.js та confirm.js .

**prompt.js:**

//----- Об’єкту myTxt присвоїти значення селекторів 'h1.n2'

var myTxt = document.querySelector('h1.n2');

//----- Об’єкту myTxt присвоїти значення селекторів '.myclass'

var myFont = document.querySelectorAll('.myclass');

// ---- Завантажуємо цикл для зміни розміру шрифту об’єкту myFont[i]

for (var i = 0; i < myFont.length; i++) {

myFont[i].style.fontSize = '35px';

}

// ---- Записуємо в об’єкт myTxt значення 'Привіт, студенте!'

myTxt.innerHTML = 'Привіт, cтуденте!';

// ---- Присвоюємо змінній person значення яке введе користувач з клавіатури

var person = prompt('Як Вас звати');

if (person != null) {

myTxt.innerHTML = "Я вітаю Вас " + person + "!" + "<br>" + " ";

}

// ---- Змінємо колір обєкту myTxt за допомогою DOM

myTxt.style.color = 'red';

**alert.js:**

Метод alert дозволяє виводити діалогове вікно з заданим повідомленням і кнопкою Оk.

//----- Метод alert () виводить на екран модальне вікно з повідомленням

alert("Як Ваші справи, " + person + "? \n Ви готові для нашої подальшої роботи?");

**confirm.js:**

Метод confirm дозволяє вивести діалогове вікно з повідомленням і двома кнопками – Оk і Відміна (Cancel). На відміну від методу alert цей метод повертає логічну величину, значення якої залежить від того, на якій з двох кнопок клацнув користувач. Якщо він клацнув на кнопці Оk, то повертається значення true (істина, так); якщо він клацнув на кнопці Скасування, то повертається значення false:

var r = confirm("Готові гризти граніт науки?");

if (r == true) {

myTxt.innerHTML = "ВПЕРЕД ДО ЗНАНЬ!";

} else {

myTxt.innerHTML = "Мені дуже сумно!";

}

var myImage = document.querySelector('img');

var j = 0;

myImage.onclick = function () {

j++;

var mySrc = myImage.getAttribute('src');

if (mySrc === 'images/firefox-icon.png') {

myImage.setAttribute ('src', 'images/firefox-icon\_bw.png');

} else {

myImage.setAttribute ('src', 'images/firefox-icon.png');

}

if (j == 3) {

myImage.remove();

document.write("<h1>Вітаю!</h1><h2>Завдання виконано!</h2>");

}

}

myTxt.innerHTML += "<br>" + " Клацніть мишею по емблемі FireFox!";

Скрипти (папка scripts) і рисунки (папка images) з файлом CSS (папка styles) можна завантажити з наступного ресурсу Інтернет: <https://drive.google.com/open?id=0B8IblPvaTQxFcXlwVUZnOTA1dEk>

* 1. Проаналізуйте послідовність виконання скриптів JavaScript.

1. Асинхронні скрипти: **defer/async**

Браузер завантажує і відображає HTML-документ поступово. Особливо це помітно при повільному Інтернет-з’єднанні: браузер показує ту частину, яку встиг завантажити. Якщо браузер бачить тег <script>, то він за стандартом зобов'язаний спочатку виконати його, а потім показати решту сторінки.

Така поведінка називається «синхронною». Але, якщо скрипт – зовнішній, то поки браузер не виконає його, він не покаже частину сторінки під ним.

<p>Важлива інформація не буде відображена, поки не завантажиться скрипт.</p>

<**script src="https://js.cx/hello/ads.js?speed=0.1"**></**script**>

<p>...Важлива інформація!</p>

Додайте цей уривок коду до Вашого HTML-документу для розуміння процесів, які відбуваються при синхронному виконанні коду JavaScript.

Для управління послідовністю виконання скриптів використовують атрибути defer та async.

Якщо скрипт має атрибут async:

<**script async src="…"**></**script**>

браузер не зупиняє обробку сторінки, а опрацьовує код далі. Після завантаження скрипт буде виконаний.

Якщо скрипт має атрибут defer, то буде виконуватися строга послідовність відповідно потоку в HTML-документі, навіть, якщо наступний скрипт буде завантажено у кеш-память браузера.

Друга відмінність – скрипт з атрибутом defer спрацює, коли весь HTML-документ буде оброблений браузером. Атрибути async/defer використовують тільки зовнішні скрипти, тобто посилання на скрипти містять атрибут src.

* 1. Поміняйте місцями скрипти в HTML-документі та додайте до них атрибути async/defer. Зробіть аналіз змін до яких це призвело.

1. Виведення інформації засобами JavaScript

Існують чотири способи виведення інформації засобами JavaScript:

1. За допомогою методу innerHTML :

<**div class="temp m1"**>…/**div**>

<**script**>  
 **var *sum*** = 0.1+0.2;  
 **var *result*** = **document**.getElementsByClassName(**"m1"**)[0];  
 ***result***.**innerHTML** = **"В Java Script "** + **"<br>"**+ **" 0.1 + 0.2 = "** + ***sum***;  
 ***result***.style.**color** = **'#2d264a'**;  
</**script**>

1. Запис (перезапис) сторінки за допомогою методу document.write( ) :

**document.write(9 + 10);**

1. Виведення в модальному вікні за допомогою window.alert( ) :

**window**.alert(“Важлива інформація!”);

1. Виведення в консоль браузера за допомогою console.log( ):

**console**.log(20 + 17);

* 1. Створіть HTML-документ який буде містити JavaScript код з усіма наведеними вище прикладами. Отриманий результат у формі Web-сторінки покажіть викладачу.

1. Спливаючі вікна в JavaScript

У JavaScript є три типи спливаючих вікон: вікно сповіщення, поле підтвердження та вікно запиту:

* вікно alert, для продовження роботи скрипта необхідно обов’язково натиснути кнопку Ok (alert("УВАГА!");)
* вікно підтвердження window.confirm( ), яке повертає два значення true або false;
* вікно запрошення prompt, яке повертає введене значення або null.

Наприклад:

**var** person = prompt(**"Прошу ввести Ваше імя"**, **"Петро"**);

* 1. Проаналізуйте код першої сторінки лабораторної роботи з спливаючими діалоговими вікнами. Змініть їх та перегляньте результат у вікні браузера.

1. Створення HTML-документу
   1. Створити HTML-документ на основі прикладу першого HTML-документу лабораторної роботи який буде містити дві кнопки та зображення лампочки. При натисненні кнопки включити – лампочка буде міняти колір з білого на жовтий, а при вимкнути – навпаки.
   2. Створити HTML-документ який буде симулятором роботи світлофору. Для зміни кольору застосуйте CSS та JavaScript.

Рисунки зображень можна завантажити з мережі Інтернет: <https://drive.google.com/open?id=0B8IblPvaTQxFZ3RfOVBPNXJ6dXM>

* 1. Результат роботи розмістити на безкоштовному сервері в мережі Інтернету (наприклад: сервер uCoz – <https://www.ucoz.ua/>).
  2. Дати відповіді на контрольні питання.

**Контрольні питання**

1. Який метод введення даних користувачем із використанням модальних вікон повертає значення true або false?
2. Поясніть який результат ми отримаємо, якщо перенесемо скрипт confirm.js у заголовок HTML-документа?
3. Яке призначення методу prompt( )?
4. Пояснити призначення та відмінності методів textContent та innerHTML.
5. Яке призначення методу confirm( )?