

Лекція 6. Основи JavaScript

1. Характеристика та властивості **JavaScript**

- **JavaScript** — назва реалізації стандарту мови програмування ECMAScript компанії Netscape, базується на принципах прототипного програмування. Найпоширеніше і найвідоміше застосування мови — написання сценаріїв для веб-сторінок, але, також, використовується для впровадження сценаріїв керування об'єктами вбудованими в інші програми.

Дата 12.1995р

Компанія Netscape.

Автори: **Брендан Айх**, Марк Андрісін, Білл Джой:

Коротка характеристика

- **Основні архітектурні риси: динамічна типізація, слабка типізація, автоматичне керування пам'яттю, прототипне програмування, функції як об'єкти першого класу.**

Об'єкт називають «об'єктом першого класу» коли він:

1. може бути збережений у змінній
2. може бути переданий у функцію як параметр
3. може бути повернутий з функції як результат
4. може бути створений під час виконання програми внутрішньо самоідентифікуємий (незалежний від іменування)

- **Динамічна типізація** — прийом, широко використовуваний у мовах програмування й мовах специфікації, при якому змінна зв'язується з типом у момент присвоювання значення, а не в момент оголошення змінної, таким чином, у різних ділянках програми ті самі змінні може надаватися значення різних типів.

Приклади мов, де є динамічна типізація — Smalltalk, Python, Objective-C, Ruby, PHP, Perl, Javascript, Lisp, xbase.

- **Прототипне програмування** — стиль об'єктно-орієнтованого програмування, при якому відсутнє поняття класу, а повторне використання (спадкування) проводиться шляхом клонування існуючого екземпляра об'єкта — прототипу.

Основні властивості

Javascript не потрібно компілювати, він підключається до HTML-сторінки й працює "як є".

Javascript може:

- змінювати сторінку,
- писати на ній текст,
- додавати й видаляти теги,
- міняти стилі елементів.
- реагувати на події: (скрипт може чекати, коли щонебудь трапиться (клік миші, закінчення завантаження сторінки) і реагувати на це виконанням функції.)

- виконувати запити до сервера й завантажувати дані без перезавантаження сторінки. (Це іноді називають "AJAX".)
- встановлювати й зчитувати cookie,
- валідувати дані (перевіряти їх коректність),
- виводити повідомлення (й багато чого іншого.)

Унікальність javascript:

- Повна інтеграція із браузером
- Прості речі робляться просто
- Підтримується майже скрізь

Структура мови

Структурно Javascript можна представити у вигляді об'єднання трьох чітко помітних друг від друга частин:

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

Якщо розглядати Javascript у відмінних від браузера застосуваннях, то об'єктна модель браузера й об'єктна модель документа можуть не підтримуватися.

2. Підключення та виконання JavaScript

Існує три основні можливості підключення до HTML-файла:

- Підключення в будь-якому місці
- Винесення скриптів у заголовок HEAD
- Зовнішні скрипти

Підключення в будь-якому місці

```
<html>
<body>
  <h1>Рахуємо кроликів</h1>
  <script type="text/javascript">
    for(var i=1; i<=3; i++)
    {
      alert("З капелюха дістали "+i+" кролика!")
    }
  </script>
  <h1>...Порахували</h1>
</body>
</html>
```

У цьому прикладі використали наступні елементи.

- `<script type="text/javascript"> ... </script>`
- Тег `<script>` повідомляє браузеру про те, що усередині перебуває скрипт. Атрибут `type` говорить про те, що це `javascript`.
- Функція `alert` - виводить повідомлення на екран і чекає, поки відвідувач не натисне ОК

Винесення скриптів у заголовок HEAD

```
<html>
  <head>
    <script type="text/javascript">
      function count_rabbits() {
        for(var i=1; i<=3; i++) {
          // оператор + з'єднує рядки
          alert("З капелюха дістали "+i+" кролика!")} }
    </script>
  </head>
  <body>
    <input type="button" onclick="count_rabbits()" value="
    Рахувати кролів!"/>
  </body>
</html>
```

JavaScript-код тільки описує функцію `count_rabbits`, а її виклик здійснюється по натисканню на кнопку `input`.

Зовнішні скрипти

Підключення окремого файлу зі скриптом:

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

- *де /my/script.js файл що містить javascript.*

- Щоб підключити декілька скриптів - використайте кілька таких тегів:

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

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

При наявності атрибута src зміст тегу ігнорується.

- **Файл Rax.js**

```
function count_rabbits() {  
    for(var i=1; i<=3; i++) {  
        alert("З капелюха дістали "+i+" кролика!");  
    }  
}
```

Файл "Рахувати кролів!"

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0  
    Transitional//EN">
```

```
<html>
```

```
    <head>
```

```
        <script language="JavaScript" src="Rax1.js"></script>
```

```
    </head>
```

```
<body>
```

```
    <input type="button" onclick="count_rabbits()  
    value="Рахувати кролів!"/>
```

```
</body>
```

```
</html>
```

3. Структура JavaScript

В Javascript:

- усі ідентифікатори регістрозалежні,
- у назвах змінних можна використовувати букви, підкреслення, символ долара, арабські цифри,
- назви змінних не можуть починатися із цифри,
- для оформлення однорядкових коментарів використовуються //, багаторядкові всередині рядка коментарі починаються з /* і закінчуються */.

Крапка з комою

Стрічки (речення), змінні мовою javascript можна розділяти крапкою з комою.

Але при переході на новий рядок її можна не ставити.

- `a = 5`

- `a = 5;`

- От так не буде працювати: `var a = "довгий
рядок "`

- А так буде `var a = "довгий\
рядок "`

Змінні та оператори можуть бути написані з пропусками або без них

- `x=3`
- `x = 3`

Але `X++` тільки підряд.

Дужки.

Все що в круглих дужках має бути в одні стрічці

Фігурні дужки, та все що в них, можна розташовувати довільно.

Літерали – значення вказане в тексті програми

- 12 // Число дванадцать
- 1.2 // Число одна целая две десятых
- "hello world" // Строка текста
- 'Hi' // Другая строка
- true // Логическое значение
- false // Другое логическое значение
- /javascript/gi // Регулярное выражение (для поиска по шаблону)
- null // Отсутствие объекта

Ідентифікатори

В Javascript:

- усі ідентифікатори регістрозалежні,
- у назвах змінних можна використовувати букви, підкреслення, символ долара, арабські цифри,
- назви змінних не можуть починатися із цифри,

- **i**

- **my_variable_name**

- **v13**

- **_dummy**

- **\$str**

Зарезервовані ключові слова JavaScript

- Break do if switch typeof
- Case else in this var
- Catch false instanceof throw void
- Continue finally new true while
- Default for null trywith
- Delete functionreturn

Типи даних

- **Стрічкові- String** - визначається подвійними або одинарними лапками й використовується для символічних даних,
- **Логічні - Boolean**- визначається відсутністю лапок і використовується для значень true=1 або false=0,
- **Цифрові - Number**- визначається відсутністю лапок і використовується для чисел (не символів).

Також існують спеціальні типи даних:

- **null** - відсутність даних,
- **об'єкт (object)** - програмний об'єкт (посилання на нього),
- **функція (function)** - визначення функції.

Змінні

- Змінні в javascript слабо типізовані. Це означає, що при оголошенні не потрібно вказувати тип. Можна надати змінній будь-яке значення.

Однак при цьому інтерпретатор javascript (браузер) завжди знає, якого типу значення містить дана змінна, а значить - які операції до неї застосовні.

- Змінну не обов'язково оголошувати. Досить їй просто надати значення:

$x = 5$

В іменах змінних можуть використовуватися латинські букви (a...z, A...Z), цифри (0...9), знак долара (\$) і знак підкреслення (_), при цьому не можна використовувати цифру першою.

Змінні в Javascript призначаються двома способами:

- 1. За допомогою оператора «=»: змінна = значення. Приклад:

```
x = 1
```

- 2. За допомогою ключового слова var і оператора «=»: var змінна або var змінна = значення.

Приклад:

```
var x
```

```
x = 1
```

```
//або
```

```
var x = 1
```

- Другий спосіб відрізняється від першого тим, що можна призначати відразу декілька змінних:

```
var x = 1, y = 2
```

- Другий спосіб визначає значення змінної тільки у своєму блоці коду.

Якщо надати змінні значення, яка не оголошена за допомогою інструкції `var`, Javascript неявно оголосить цю змінну за вас, але завжди як глобальну.

Всі "глобальні" змінні прив'язані до свого вікна. Різні вікна й фрейми мають різні "глобальні" змінні, котрими можуть обмінюватися один з одним.

Оператори

- **Арифметичні**

Арифметичні операції проводяться тільки з даними одного типу.

+ - Додавання - $(x + y)$ - Додавання двох чисел або склеювання двох рядків

- Віднімання - $(x - y)$ - Віднімання y з x або видалення рядка y з рядка x

-Множення - $(x * y)$ - Перемножування двох чисел

/ ділення - (x / y) - ділення числа x на число y

%- відсоток (залишок) - $(x \% y)$ - Залишок від ділення числа x на число y

++ Інкремент (збільшення на 1) - **x++** - Еквівалентно - **x+1**

-- Декремент (зменшення на 1) - **(x--)** - Еквівалентно - **x-1**

- **5 + true // результат 6**
- **5 + false // результат 5**
- **4 * true // результат 4**
- **4 * false // результат 0**
- **true + true // результат 2**
- **true + false // результат 1**
- **"Java" + "Script" // результат "JavaScript"**
- **a = "Java"**
- **b = "Script"**
- **a + b // результат "JavaScript"**
- **"Уровень" + 5 // результат "Уровень5"**
- **"Уровень" + true // результат "Уровеньtrue"**
- **"1" + "2" // результат "12"**

4. Керуючі інструкції

Послідовність символів яка призводить до певних дій над змінними (об'єктами) називають - керуючою інструкцією.

Прості інструкції:

Присвоювання - $s = \text{"Привіт"} + \text{ім'я}$;

Інкремент(декремент) - $i++$;

Складені інструкції - {

$x = \text{Math.PI}$;

$cx = \text{Math.cos}(x)$;

$\text{alert}(\text{"cos("} + x + \text{"}) = " + cx)$;

}

Базові керуючі інструкції

- Інструкція **if**

Інструкція має дві форми.

Перша: `if (вирази) інструкція`

У цій формі інструкції `if` спочатку обчислюється вираз. Якщо отриманий результат рівний `true` або може бути перетворений в `true`, то виконується інструкція. Якщо вираз рівний `false` або перетвориться в `false`, то інструкція не виконується.

- Наприклад:

```
if ((address == null) || (address == "")) {  
    address = "undefined";  
    alert(" Будь ласка, укажіть поштову адресу.");}
```

- Друга форма інструкції if вводить конструкцію else, що виконується в тих випадках, коли вираз рівний false. Її синтаксис:

```
if (вираз)
```

```
    інструкція1
```

```
else
```

```
    інструкція2
```

```
if (username != null)
```

```
    alert("Привіт " + username + "\n Ласкаво прошу на  
    мою домашню сторінку.");
```

```
else {
```

```
    username = prompt(" Ласкаво просимо!\n Як вас  
    кличуть?");
```

```
    alert("Привіт " + username);
```

```
}
```

Якщо потрібно виконати один з багатьох фрагментів коду то можливим способом зробити це полягає в застосуванні інструкції else if.

```
if (n == 1) {  
    // Виконуємо блок коду 1  
}  
else if (n == 2) {  
    // Виконуємо блок коду 2  
}  
else if (n == 3) {  
    // Виконуємо блок коду 3  
}  
else {  
    // Якщо всі решта умов else не виконуються, виконуємо блок 4  
}
```

Інструкція switch

switch(вираз) { інструкції }

Коли виконується інструкція `switch`, вона обчислює значення виразу, а потім шукає мітку `case`, відповідну цьому значення. Якщо мітка знайдена, виконується блок коду, починаючи з інструкції, що за міткою `case`. Якщо мітка `case` з відповідним значенням не знайдена, виконання починається з першої інструкції, що за спеціальної міткою `default`:. Якщо мітки `default`: нема, блок коду пропускається цілком.

```
function convert(x) {
```

```
  switch(typeof x) {
```

```
    case 'number': // Перетворимо число в шіснадцяткове ціле  
      return x.toString(16);
```

```
    case 'string': // Повертаємо рядок, взятий в лапки  
      return "" + x + "";
```

```
    case 'boolean': // Перетворимо в TRUE або FALSE, у верхньому регістрі  
      return x.toString().toUpperCase();
```

```
  default: return x.toString() } } // Будь-який інший тип перетворимо  
звичайним способом
```

Інструкція `while`

Інструкція `while` – це базова інструкція, що дозволяє JavaScript виконувати повторювані дії. Вона має наступний синтаксис:

while (вираз)
інструкція

Інструкція `while` починає роботу з обчислення виразу. Якщо він рівний `false`, інтерпретатор JavaScript переходить до наступної інструкції програми, а якщо `true`, то виконується інструкція, що утворює тіло циклу, і вираз обчислюється знову.

Приклад циклу while:

```
var count = 0; while (count < 10) {  
    document.write(count + "<br>");    count++; }
```

Цикл **do/while**

- Цикл **do/while** багато в чому схожий на цикл **while**, за винятком того, що умова циклу перевіряється наприкінці, а не на початку циклу. Це значить, що тіло циклу завжди виконується хоча б один раз. Синтаксис такий:

do

інструкція

while (вираз);

Інструкція for

Синтаксис циклу for:

for(ініціалізація; перевірка; інкремент)

інструкція

Вираз ініціалізація обчислюється один раз перед початком циклу. Цей вираз, як правило, є виразом з побічними ефектами (за звичай присвоюванням), допускається щоб у виразі ініціалізація була інструкція оголошення змінної var, тому можна одночасно оголосити та ініціалізувати лічильник циклу. Вираз перевірка обчислюється перед кожною ітерацією й визначає, чи буде виконуватися тіло циклу. Якщо результат перевірки рівний true, виконується інструкція, що є тілом циклу. Наприкінці циклу обчислюється вираз інкремент. Звичайно це або вираз присвоювання, або вираз, що використовує оператор ++ або --.

```
for(var count = 0; count < 10; count++)
```

```
document.write(count + "<br>");
```

Інструкція for/in

Має наступний синтаксис:

for (змінна *in* об'єкт)

інструкція

Тут змінна повинна бути або іменем змінної, або інструкцією `var`, що оголошує змінну, або елементом масиву, або властивістю об'єкта (тобто повинна бути тим, що може перебувати в ліві частини виразу присвоювання). Параметр об'єкт – це ім'я об'єкта або вираження, результатом якого є об'єкт. Інструкція – це інструкція або блок інструкцій, що утворюють тіло циклу.

Наприклад цикл, `for/in` друкує імена й значення всіх властивостей об'єкта:

```
for (var prop in my_object) { document.write("ім'я: " +  
prop + "; значення: " + my_object[prop], "<br>");  
}
```

Функції перевірки і перетворення типів

- `isNaN(значення)` - перевірка, чи являється значення
- числом `parseInt(значення)` перетворення до цілого числа
- `parseFloat(значення)` перетворення до дійсного числа
- `eval` обчислює рядок, що представляє будь-які JavaScript літерали або змінні, перетворюючи її в число.
- Приклади.
- `x=parseInt(F1.T1.value)`

Деякі константи і методи(функції) об'єкту Math

Math.min(<i>число, число</i>)	менше / більше з двох
Math.max(<i>число, число</i>)	
Math.round(<i>число</i>)	округлення до цілого
Math.sqrt(<i>число</i>)	квадратний корінь
Math.pow(<i>основа, показник</i>)	піднесення до степені
Math.log(<i>число</i>)	натуральний логарифм
Math.abs(<i>число</i>)	абсолютне значення