Тема. Основи JavaScript

План:

1. Ключові терміни, рядковий літерал
2. Лексичні структрури JavaScript
3. Типи даних JavaScript
4. Вирази
5. Операції JavaScript
6. Оператори JavaScript
7. Змінні
8. Ключові терміни, рядковий літерал

JavaScript - це інтерпретована мова програмування з об'єктно-орієнтованими можливостями.

Зазвичай JavaScript застосовується у веб-браузерах, а розширення його можливостей за рахунок введення об'єктів дозволяє організувати взаємодію з користувачем, управляти веб-броузером і змінювати вміст документа, що відображається в межах вікна веб-браузера. Ця вбудована версія JavaScript запускає сценарії, впроваджені в HTML-код веб-сторінок.

В основі мови JavaScript і підтримуваних ним типів даних лежать международні стандарти, завдяки чому забезпечується прекрасна сумісність між реалізаціями. Деякі частини клієнтського JavaScript формально стандартизовані, інші частини стали стандартом де-факто, але є частини, які є специфічними розширеннями конкретної версії браузера.

Клієнтський JavaScript включає в себе інтерпретатор JavaScript і об'єктну модель документа (Document Object Model, DOM), яка визначається веб-браузером.

Документи можуть містити JavaScript-сценарії, які в свою чергу можуть використовувати модель DOM для модифікації документа або управління способом його відображення. Іншими словами, можна сказати, що клієнтський JavaScript дозволяє визначити поведінку статичного вмісту веб-сторінок.

JavaScript забезпечує можливість управління не тільки вмістом HTML документів, але і їх поведінкою. Іншими словами, JavaScript-програма може реагувати на дії користувача: введення значення в текстове поле або клацання мишею в області зображення в документі. Це досягається шляхом визначення “обробників подій” для документа - фрагментів JavaScript-коду, виконуваних при виникненні певної події, наприклад клацання на кнопці.

Можливості відображення JavaScript

JavaScript може «відображати» дані різними способами:

- Запис в елемент HTML за допомогою `innerHTML`.
- Запис у вивід HTML за допомогою `document.write()`.
- Запис у вікні сповіщень за допомогою `window.alert()`.
- Запис у консоль браузера за допомогою `console.log()`

Лексичні структрури JavaScript

При написанні програм на JavaScript використовується набір символів Unicode. JavaScript - це мова, чутлива до регістру.

Багато JavaScript-об'єктів та їх властивості мають ті ж імена, що і теги і атрибути мови HTML, які вони позначають.

JavaScript ігнорує пробіли, табуляції і переведення рядків, присутні між лексемами в програмі.

Прості JavaScript-інструкції зазвичай завершуються символами крапки з комою (;), крапка з комою служить для відділення інструкцій один від одного.

Будь-який текст, присутній між символами // і кінцем рядка, розглядається як коментар і ігнорується JavaScript.

Будь-який текст між символами / * і * / також розглядається як коментар.

Типи даних JavaScript

JavaScript дозволяє працювати з:

- трьома елементарними типами даних:
- числами;
- рядками тексту (або просто рядками);
- значеннями логічної істинності (або просто логічними значеннями).
- двома тривіальними типами даних (кожен з яких визначає тільки одне значення):
- null
- undefined

складений тип даних: об'єкт (object) (або член об'єктного типу даних) - колекція значень (або елементарних, таких як числа і рядки, або складних, наприклад **інших об'єктів).**

Об'єкти в JavaScript мають подвійну природу:

об'єкт може бути представлений як невпорядкована колекція іменованих значень;

як упорядкована колекція пронумерованих значень. В останньому випадку об'єкт називається масивом (array).

В JavaScript визначений ще один спеціальний тип об'єкта, відомий як функція (function) - це об'єкт, з яким пов'язаний виконуваний код.

Функція може викликатися (invoked) для виконання певної операції.

Крім функцій і масивів у базовому мові JavaScript визначено ще декілька спеціальних видів об'єктів.

Ці об'єкти являють собою не нові типи даних, а лише нові класи (classes) об'єктів.

Синтаксис запису цифр:

[цифри] [. цифри] [(E | e) [(+ |?)] цифри]

Наприклад:

3.14

2345.789

.333333333333333333

6.02e23 // 6.02 X 1023

1.4738223-? 32 // 1.4738223 X 10-32

дійсних чисел існує нескінченно багато, але формат подання дійсних чисел в JavaScript дозволяє точно задати лише обмежену їх кількість (точніше +18437736874454810627).
будь-якому числовому літералу може передувати знак «мінус» (-), що робить числа від'ємними.

Рядок являє собою послідовність букв, цифр, знаків пунктуації та інших Unicode-символів і є типом даних JavaScript для представлення тексту.

Рядковий літерал - це послідовність з нуля або більше Unicode-символів, укладена в одинарні або подвійні лапки (‘ або ").

Самі символи подвійних лапок можуть міститися в рядках, обмежених символами одиуарних лапок, а символи одинарних лапок - у рядках, обмежених символами подвійних лапок.

Рядкові літерали повинні записуватися в одному рядку програми і не можуть розбиватися на два рядки.

Щоб включити в строковий літерал символ перекладу рядка, слід використовувати последовність символів \ n

Оскільки апостроф і одинарні лапки - це одне і те ж, необхідно за допомогою символу зворотного слеша (\) екранувати апострофи, розташовані всередині одиночних лапок.

Основні управляючі послідовності JavaScript:

\0 - символ NUL (\u0000), весь текст, що слідує після нього, не буде відображатися
\b – еквівалент клавіши Backspace (\u0008)
\t –горизонтальна табуляція (\u0009)
\n – перевод строки (\u000A)
\f - перевод сторінки (\u000C)
\r - повернення каретки (\u000D)
\" - подвійні лапки (\u0022)
\' - одинарні лапки(\u0027)
\\ - обернений слеш (\u005C)
\xXX – Тут XX – дві шістнадцятирічні цифри
\uXXXX – Юнікод символ, який задано чотирма шістнадцятирічними цифрами XXXX

Якщо оператор + застосовується до чисел, вони додаються, а якщо до рядків, вони об'єднуються, при цьому другий рядок додається в кінець першого.

Перетворення чисел в рядки проводиться автоматично, у міру необхідності.

Наприклад, якщо число використовується в операції конкатенації рядків, воно буде перетворено в рядок:

```
var n = 100;
var s = n + "стаканів води";
```

Щоб перетворити число в рядок, достатньо просто скласти його з порожнім рядком:

```
var n_as_string = n + "";
```

Для явного перетворення числа в рядок використовується функція String ():

```
var string_value = String (number);
```

Коли рядок використовується в числовому контексті, вона автоматично перетворюється на число.

При необхідності перетворити рядок в число достатньо просто відняти з рядка значення 0:

```
var number = string_value - 0;
```

Прямолінійний спосіб перетворення рядка в число полягає у зверненні до конструктора Number () як до звичайної функції:

var number = Number (string_value);
Більш гнучкий спосіб перетворення забезпечується функціями parseInt () і parseFloat(). Ці функції перетворять і повертають довільні числа, що стоять на початку рядка, ігноруючи будь-які нецифрові символи, розташовані слідом за числом.
Функція parseInt () виконує тільки цілочисельне перетворення, тоді як parseFloat () може перетворювати як цілі, так і дійсні числа.
Приклад використання функції parseInt ()
Виклик функцій поверне 15:
parseInt("15abc");
parseInt("15*3", 10);
Виклик функцій поверне NaN:
parseInt("Hello", 8); // не є числом
parseInt("546", 2); // цифри не входять у двійкову систему
Виклик функцій поверне 3.14: Приклад використання функції parseFloat ()
parseFloat("3.14");
parseFloat("314e-2");
parseFloat("0.0314E+2");
Коли в якості логічного значення використовується число, воно перетвориться в значення true, якщо воно не дорівнює значенням 0 або NaN, які перетворюються в логічне значення false.
Коли в якості логічного значення використовується рядок, вона перетворюється в значення true, якщо це не порожній рядок, в протилежному випадку в результаті перетворення виходить значення false.
Спеціальні значення null і undefined перетворюються в false, а будь-які функція, об'єкт або масив, значення яких відмінні від null, перетворюються в true.

Вирази

Вираз - це фраза мови JavaScript, яка може бути обчислена інтерпретатором для отримання значення. Найпростіші вирази - це літерали або імена змінних. Шляхом об'єднання простих виразів можуть створюватися більш складні вираження.
Найпростіші вирази - це літерали або імена змінних, наприклад:
1.7 // Числовий літерал
"JavaScript is fun!" // Рядковий літерал
true // Логічний літерал
null // Літерал значення null
{ x:2, y:2 } // Об'єктний літерал
[2,3,5,7,11,13,17,19] // Літерал масива
function(x){return x*x;} // Функціональний літерал
i // Змінна i
sum // Змінна sum
Значення виразу литерала - це просто значення самого літерала.
Значення виразу змінної - це значення, що міститься в змінній, або значення, на яке змінна посилається.
Шляхом об'єднання простих виразів можуть створюватися більш складні вирази. Наступний приклад теж являє собою вираз:
i + 1.7

5. Операції JavaScript

1. Арифметичні.

Операція	Значення
+	додавання
-	віднімання
*	множення
/	ділення
%	ділення по модулю
++	інкремент
--	декремент

Операція "+" виконує конкатенацію, якщо хоча б один операнд - рядок, причому, не обов'язково перший.

2. Порівняння.

Операція	Значення
==	дорівнює
!=	недорівнює
<	менше
<=	менше або дорівнює
>	ділення
>=	більше або дорівнює
===	ідентично
!==	не ідентично

3. Побітові.

Операція	Значення
&	Побітове (AND)
|	Побітовое АБО (OR)
^	Побітове виключаюче АБО XOR)
~	Побітовое НЕ (NOT)
<<	Зсув вліво
>>	Зсув вправо, який переносить знак
>>>	Зсув вправо із заповненням нулями

4. Логічні.

Оператор	Значення
&&	Логічне II
||	Логічне АБО
!	Логічне НЕ

Оператори JavaScript

Умовний оператор.

1. Неповне розгалудження

if (умова)
оператор

Приклад.

```
if (username == null)
username = "Jonn";
```

2. Повне розгалудження

```
if (умова)
оператор 1
else
```

оператор 2
```
var num = 15;
if (num > 10)
alert("число " + num + " більше 10");
else
alert("число " + num + " менше 10");
```
Для виконання одного із фрагментів програмного коду у випадку, якщо умова не істинна, використовується else if.
Приклад.
```
if (n == 1) {
// Виконуємо блок кода 1
}
else if (n == 2) {
// Виконуємо блок кода 2
}
else if (n == 3) {
// Виконуємо блок кода 3
}
else {
// Якщо всі інші else не виконуємося, виконуємо блок кода 4
}
```
Замість умовного оператора можна використовувати тернарну операцію. Її синтаксис:
умова ? оператор 1 : оператор 2
Приклад.
```
if (a==true) then b=1 else b=0; //умовний оператор
var b=a?1:0; //тернарна операція
```
Оператор Switch
```
switch(умова)
{
оператори
}
```
Приклад.
```
switch(n) {
case 1: //Виконується, якщо n == 1
// Виконуємо блок кода 1.
break; // Зупиняємося
case 2: // Виконується, якщо n == 2
// Виконуємо блок кода 2.
break; // Зупиняємося
case 3: // Виконується, якщо n == 3
// Виконуємо блок кода 3.
break; // Зупиняємося
default: // Якщо n не дорівнює 1,2 або 3
// Виконуємо блок кода 4.
break; //Зупиняємося
}
```
Оператори циклу:
Дві форми оператора while:

1. while
/ умова перевірється перед виконанням тіла циклу
while(i<5) { … }
2. do while
// умова перевіряється після виконання тіла циклу
do { … } while (i<5)
Дві форми оператора for.
1. Цикл for
for (var i=0;i<10;i++)
{
...
}
2. Цикл for/in
for(key in obj)
{
... obj[key]
}
Приклад. Цикл for/in друкує імена і значення всіх властивостей об'єкта:
for (var prop in my_object) {
document.write("имя: " + prop + "; значение: " + my_object[prop], "
");
}
Оператор break:
Виконує негайний вихід із циклу
Оператор continue:
Запускає наступну ітерацію циклу
Оператор return:
Розташовується лише в тілі функції. Повертає результат виконання функції.

Змінні

Змінні оголошуються за допомогою ключового слова var:
var i;
var sum;
Можна оголосити кілька змінних:
var i, sum;
Крім того, оголошення змінної можна поєднувати з її ініціалізацією:
var message = "hello";
var i = 0, j = 0, k = 0;
Якщо початкове значення не задано в інструкції var, то змінна оголошується, але її початкове значення залишається невизначеним (undefined), поки не буде змінено програмою.