JavaScript. Основи
Лекція 1

JavaScript. Основи
Лекція 1

- **ЕЛЕМЕНТИ МОВИ**
  - Вступ
  - Впровадження в документ
  - Лексична структура (Lexical structure)
  - Коментарі
  - Змінні
  - Ідентифікатри
- **ТИПИ ДАНИХ**
  - Типи даних
  - Перетворення типів
- **ОПЕРАЦІЇ НАД ОПЕРАНДАМИ**
  - Арифметичні операції
  - Операції присвоювання
  - Операції відношення
  - Логічні операції

Вступ

JavaScript (ECMAScript) - прототипно-орієнтована скриптова мова програмування.

- Користувальницький інтерфейс
  - Створення візуальних ефектів (анімація)
  - Виконання нескладних обчислень.
  - Перевірка даних введених користувачем.
  - Маніпуляція даними введеними користувачем у форми.
  - Пошук за даними вбудованим в завантажується сторінку.
  - Збереження даних в cookies.
- Динамічна зміна вмісту сторінки (DHTML).
- Пряма взаємодія з сервером без перезавантаження всієї сторінки (AJAX).

Вступ

... JS був зобов'язаний «виглядати як Java», тільки трохи менше, бути таким собі молодшим братом-простаком для Java ...

... Крім того, він повинен був бути написаний за 10 днів, а інакше ми б мали щось гірше JS ... У той час ми повинні були рухатися дуже швидко, тому що знали, що Microsoft йде за нами ...

Брендан Айк (*Brendan Eich*)

Вступ

Java

- написати код
- скомпілювати
- упакувати в аплет
- підключити

JavaScript

- написати код
- ~~скомпілювати~~
- ~~упакувати в аплет~~
- ~~підключити~~

Синтаксис → C, C ++, Java

Наслідування через прототипи → Self

Динамічна типізація → Perl

Посилання на функції → Lisp

Впровадження в документ

```
<script>
  document.getElementById("demo").innerHTML = "Привіт світ";
</script>
<script type="text/javascript">
<!DOCTYPE html>
<html><head>
<script>
function myFunction() {
 document.getElementById("demo").innerHTML="Абзац змінено";
}
</script>
</head>
<body>
<h1>Моя Веб-сторінка</h1>
<p id="demo">Абзац</p>
<button type="button"onclick="myFunction()">Спробуй</button>
</body>
</html>
```

Впровадження в документ

Зовнішній JavaScript

Сценарії можуть бути розміщені в зовнішніх файлах.

Зовнішні скрипти практичні, коли той же код використовується в різних веб-сторінках.

JavaScript-файли мають розширення `.js`.

Приклад:

```
<!DOCTYPE html>
<html>
<body>
<script src="myScript.js"></script>
</body>
</html>
```

Тег `<script>` можна розміщувати як в секції `<body>`так і в `<head>`

Лексична структура (Lexical structure)

Лексична структура мови програмування - це набір елементарних правил, що визначають, як пишуться програми на мові.

Лексична структура JavaScript:

- Чутливий до регістру
- Інструкції повинні закінчуватися крапкою з комою або починатися з нового рядка
- Ігнорує пробіли та табуляції

```
var a = 0;

function add (a, b) {
  return a + b;
}

while (true) {

}
```

Коментарі

Однорядковий коментар

```
// Однорядковий коментар
```

Багаторядковий

```
/ *
  багато
  рядковий
  коментар
* /
```

Змінні

Змінна складається з імені та виділеної області пам'яті, яка їй відповідає.

Для оголошення або, іншими словами, створення змінної використовується ключове слово **var**:

```
var message;
message = 'Hello';

var message = 'Hello';

var user = 'John', age = 25, message = 'Hello';

MyCount = 22;
```

JavaScript - динамічно типізована, а не строго типізована мова програмування.

Іменування змінних

Ім’я може складатися з;

A-z 0-9 _ $

Ім'я не може починатися з цифри:

element21 21element

Першим символом імені може бути буква або символ '$' або '_':

$Str _name

Ідентифікатор повинен бути одним словом:

myName My Name

Ідентифікатор не повинен збігатися з зарезервованими словами

Myfunction function

При іменуванні часто використовуються правила camelCase та PascalCase

Типи даних (Data Types)

Елементарні типи

- Числовий (Number)
- Рядковий (String)
- Логічний (Boolean)

Тривіальні типи

- null - відсутність значення
- undefined - невизначене значення

Складові типи

- Об'єкт (Object)
- Масив (Array)

Спеціальний тип

- Функція (function)

Числовий формат (Number)

```
0, 117, -345 // десяткове, основа 10
015, 0001, -0o77 // вісімкове, основа 8
0x1123, 0X00111, -0xF1A7//шістнадцяткове,"Hex" або основа 16
0b11, 0b0011, -0B11 // бінарне, основа 2
0.5 // десяткове, з плаваючою крапкою
.1e-23 // науковий запис
```

Всі числа зберігаються в форматі **`float64`** - 8 байт з плаваючою точкою.
У цьому форматі не завжди можливі точні обчислення.

```
0.1 + 0.2 == 0.3; // false
0.1 + 0.2; // 0.30000000000000004

(0.1 * 10 + 0.2 * 10) / 10 // 0.3

9999999999999999; //10000000000000000
```

Числовий формат (Number)

Спеціальні значення

NaN

Якщо математична операція не може бути здійснена, то повертається спеціальне значення NaN (Not-A-Number).

```
0/0; // NaN
NaN === NaN; // false
NaN == NaN; //false
```

infinity

Що повинно відбуватися при спробі ділення на нуль?

```
1/0; // Infinity
-1/0; // -Infinity
var a = Number.MAX_VALUE; // 1.7976931348623157e+308
a + a; // Infinity
Infinity / Infinity; // NaN
```

Типи даних

Логічний тип даних.

Дані зберігаються в змінної булевого типу можуть приймати два значення: істина (`true`) та ложь (`false`).

Часто істина представляється одиницею, а ложь - нулем.

Рядки - тип даних для подання тексту (в рядок може входити будь-UNICODE символ)

```
var string1 = "Hello ";
var string2 = 'World';
var resString = string1 + string2; // Hello World
```

\n – новий рядок \r – повернення каретки \t – табуляція \uXXXX – юнікод символ

Типи даних

undefined - невизначене значення

```
var a;
console.log (a); // undefined
```

Null - спеціальне значення, яке має сенс «нічого» або «значення невідомо».

```
var age = null;
```

Оператор typeof

Оператор `typeof` повертає тип аргументу. Формат:

```
typeof x        typeof(x)

typeof undefined // "undefined"  typeof 0 // "number"
typeof true // "boolean"   typeof {} // "object"
typeof "foo" // "string"
```

Арифметичні операції

Операція	Знак	Приклад	Результат (x = 10)
Додавання	+	`y = x + 2`	y = 12
Віднімання	–	`y = x - 5`	y = 5
Добуток	*	`y = x * 2`	y = 20
Ділення	/	`y = x / 4`	y = 2.5
Ділення по модулю	%	`y = x % 3`	y = 1
Інкремент	++	`++x` `x++`	x = 11
Декремент	--	`--x` `x--`	x = 9

Пріоритети арифметичних операцій:

- добуток, ділення, ділення по модулю;
- додавання, віднімання.

Операції одного рівня виконуються зліва направо.

Інкремент і декремент

Існує два способи запису цих операторів: префіксна (знак операції ставиться перед операндом) і постфіксна (знак операції ставиться після операнда).

При префіксной формі запису спочатку над операндом виконується дія, а потім він використовується.

При постфіксной формі – спочатку операнд використовується, а потім виконується дія.

Приклад

```
var i = 2;
alert (++i);          //виводиться 3
alert (i++);          //виводиться 3
alert (i);            //виводиться 4
alert (i--);          //виводиться 4
alert (i);            //виводиться 3
alert (--i);          //виводиться 2
```

Перетворення типів

Важливо пам'ятати, що в операції додавання, якщо хоча б один з операндів є рядком, то другий буде також перетворений до рядка! Причому не важливо, праворуч або ліворуч знаходиться операнд-рядок. наприклад:

```
console.log ('1' + 2); // "12"
console.log (2 + '1'); // "21"
```

В інших арифметичних операціях (віднімання, множення, ділення), операнди-рядки навпаки будуть перетворені до числам:

```
console.log (2 - '1'); // 1
console.log (6 / '2'); // 3
console.log ('4' - '1'); // 3
```

```
var apples = "2"; oranges = "3";
console.log( apples + oranges );
console.log( +apples + +oranges );
```

```
//виводиться 23
//виводиться 5
```

Перетворення типів

Перетворення в числовий тип **Number** здійснюється:

- `Number()` `Number("1") // => 1`
- `parseInt()` або `parseFloat()` `parseInt("1") // => 1`
- **Оператор** `унарний +` `+"1" // => 1`

не рекомендується для використання

parseInt() або **parseFloat()** більш гнучкий ніж **Number()**

```
Number("10 apples") // => NaN
parseInt("10 apples") // => 10
```

Вони можуть отримати значення, тільки якщо рядок починається з нього

```
parseInt("apples 10") // => NaN
```

Перетворення типів

Перетворення в рядковий тип **String**

- Функція String () `String(10) // => "10"`
- Оператор + з операндом рядкового типу `10 + "" // => "10"`

(вважається поганою практикою)

Перетворення в логічний тип **Boolean**

- Функція Boolean () `Boolean(12) //=> true`
- Подвійне заперечення !! `!!12 // => true`

Арифметичні операції із присвоюванням

Назва	Знак	Приклад	Альтернативний запис
Присвоювання	=	y = x	y = x
Додавання з заміщенням	+=	y += 2	y = y + 2
Віднімання з заміщенням	-=	y -= 5	y = y - 5
Добуток із заміщенням	*=	y *= 7	y = y * 7
Ділення з заміщенням	/=	y /= 4	y = y / 4
Ділення по модулю з заміщенням	%=	y %= 3	y = y % 3

```
var x = 2 * 2 + 1; // 5
var a, b, c;
a = b = c = 2 + 2;
alert( a, b, c ); // 4 4 4
```

```
var a = 1;
var b = 2;
var c = 3 - (a = b + 1);
alert(a); // 3
alert(c); // 0
```

Операції відношення. Логічні операції

Назва	Знак
Рівність	==
Ідентичність	===
Нерівність	!=
Не ідентичність	!==
Менше	<
Більше	>
Менше або рівне	<=
Більше або рівне	>=

Назва	Знак
Інверсія (заперечення)	!
Кон'юнкція (логічний добуток)	&&
Диз'юнкція (логічне додавання)	||

Приклад

```
2 > 5 // => false
2 == 2 // => true
2 != 2 // => false
'string' == 'string' // => true
```

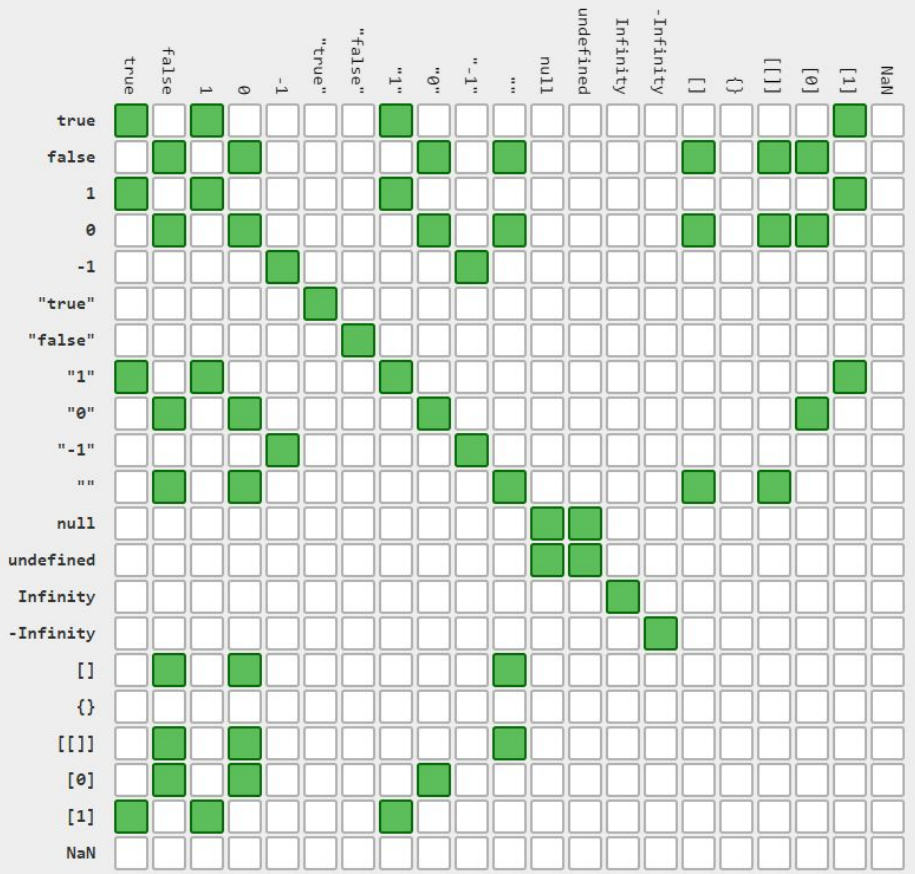
Приклад

```
!true // => false
true && false // => false
true || false // => true
```

"==" vs "==="

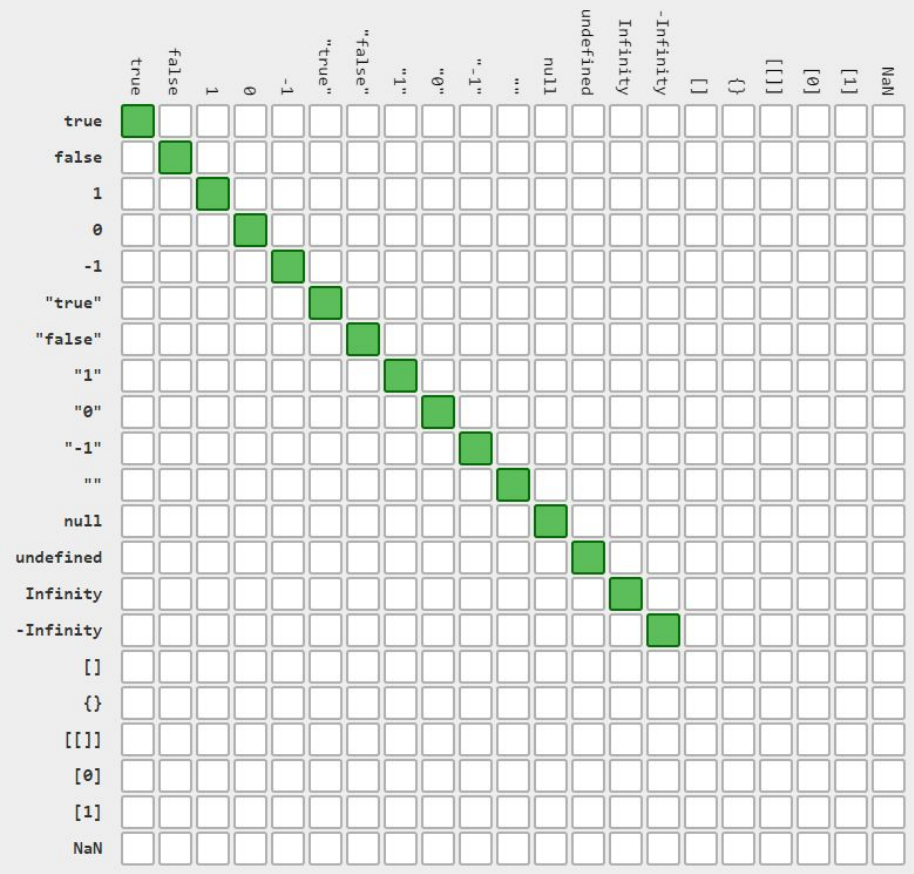
== (negated: !=)

When using two equals signs for JavaScript equality testing, some funky conversions take place.



=== (negated: !==)

When using three equals signs for JavaScript equality testing, everything is as is. Nothing gets converted before being evaluated.



Котрольні питання

Дякую!