Введение в JavaScript

Лекция 5

Как использовать JS?

Встраивание в страницу

```
<script>
  console.log("Hello, world!");
</script>
```

Отдельный файл

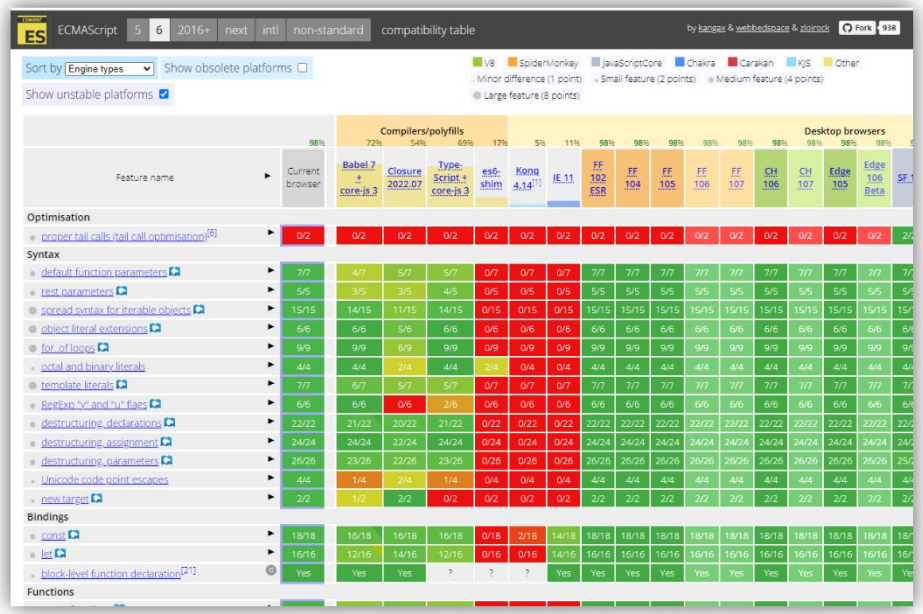
myScript.js:

```
function helloWorld() {
  console.log("Hello, world!");
}
```

```
<script src="myScript.js"></script>
<script>
  helloWorld();
</script>
```

Немного о языке

Современный стандарт ES-2015 (ES6)



Переменные

- **const** x = 3;
- **var** y = 5.0;
- **let** z;

```
y = 2;
z = x + y;
console.log(z);  // 5

z = “test” + z;
console.log(z);  // “test5”
```

- ***var*** – устаревший формат, обладает своими особенностями
- ***let*** – современный формат (рекомендованный)
- ***const*** – константы (но необычные)

Именование переменных

```
let  x
const X
var  _x
let  x123
const  $x321
var  varInLowerCamelCase
```

A-Z a-z _ $ 0-9
Unicode

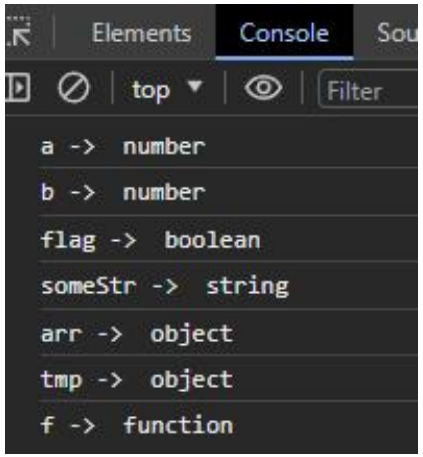
abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

Типы данных в JS

- **Целые числа:** let a = **123**
- **Дробные числа:** let b = **12.3**
- **Логические:** const flag = **true | false**
- **Строки:** let someStr = **“123”**
- **Массивы:** const arr = **[123, “123”]**
- **Объекты:** const tmp = **{x: 123, y: “123”}**
- **Функции:** let f = **(x) => { return x*x; }**

Чтобы узнать тип используем ***typeof a***

```
let a = 123;
let b = 12.3;
const flag = true;
let someStr = "123";
const arr = [123, "123"];
const tmp = {x: 123, y: "123"};
let f = (x) => {  return x*x; };

console.log("a -> ", typeof a);
console.log("b -> ", typeof b);
console.log("flag -> ", typeof flag);
console.log("someStr -> ", typeof someStr);
console.log("arr -> ", typeof arr);
console.log("tmp -> ", typeof tmp);
console.log("f -> ", typeof f);
```



Операторы сравнения

```
let x = 5;


x == 8  // false
x == 5  // true
x == “5“  // true
x === 5  // true
x === “5“  // true
```

- = – это оператор присваивания;
- == – сравнивает значения, не смотря на тип данных;
- === – сравнивает, учитывая различия в типах данных;
- ==== – такого оператора нет;

Ветвления

- **Простой if...else**

```
if (x == 5) {
        y = 7;
} else {
        y = 10;
}
```

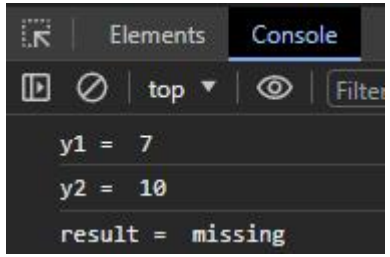
- **Тернарный оператор**

```
y = (x == 5 ? 7 : 10)
```

- **Оператор нулевого слияния**

```
let name = null, text = "missing";
let result = name ?? text;
```

```
let x = 5;
let y = 0;

if (x == 5) {
  y = 7;
} else {
  y = 10;
}

console.log("y1 = ", y);

x = 15;

y = x == 5 ? 7 : 10;

console.log("y2 = ", y);

let name = null,
  text = "missing";
let result = name ?? text;

console.log("result = ", result);
```



Ветвления

- **switch { case; case; }**

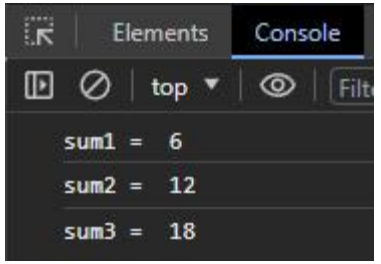
```
switch (x)  // ===
{
  case 1:
  case 2:
     y = 5;
     break;
  default:
     y = 10;
}
```

```
let x = 1;
let y = 0;

switch (x) {
  case 1:
  case 2:
    y = 5;
    break;
  default:
    y = 10;
}

console.log("x = ", x);
console.log("y = ", y);
```

Циклы

```
for (let i = 0; i < array.length; i++) {
   sum += array[i];
}

for (let i in array) {
  sum += array[i];
}

for (let x of array) {
  sum += x;
}
```

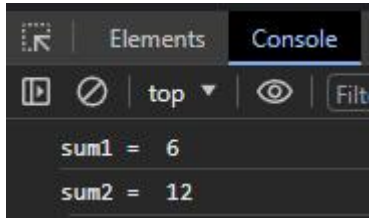
```
const array = [1, 2, 3];
let sum = 0;

for (let i = 0; i < array.length; i++) {
  sum += array[i];
}

console.log("sum1 = ", sum);

for (let i in array) {
  sum += array[i];
}

console.log("sum2 = ", sum);

for (let x of array) {
  sum += x;
}

console.log("sum3 = ", sum);
```



Циклы

```
while (i < array.length) {
  sum += array[i];
  i++;
}

do {
  // code block to be executed
}
while (condition);
```

```
let i = 0;
const array = [1, 2, 3];
let sum = 0;

while (i < array.length) {
  sum += array[i];
  i++;
}

console.log("sum1 = ", sum);

i = 0;

do {
  sum += array[i];
  i++;
} while (i < array.length);

console.log("sum2 = ", sum);
```



break, continue, label

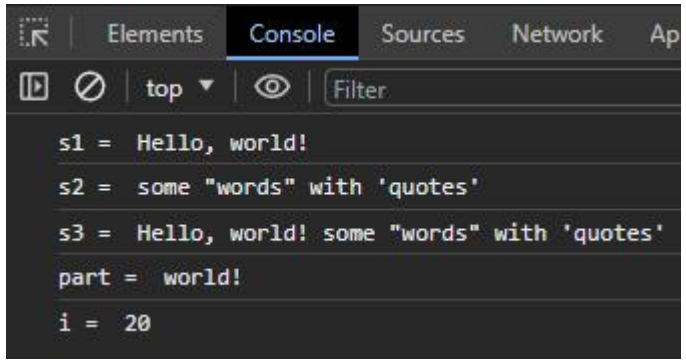
```
for (let i = 0; i < 10; i++) {
  if (i === 3) { break; }
  text += "The number is " + i + "<br>";
}
```

- **continue** – выход из одной итерации;
- **break** – из всего цикла (блока);
- **label** – не используйте никогда!

Подробнее про типы: строки

```
let s1 = "Hello, world!";
let s2 = 'some "words" with \'quotes\'';
let s3 = s1 + " " + s2;

let part = s3.slice(7, 13);
let i = s3.indexOf("word");
```

```
let s1 = "Hello, world!";
let s2 = "some \"words\" with 'quotes'";
let s3 = s1 + " " + s2;

let part = s3.slice(7, 13);
let i = s3.indexOf("word");

console.log("s1 = ", s1);
console.log("s2 = ", s2);
console.log("s3 = ", s3);
console.log("part = ", part);
console.log("i = ", i);
```



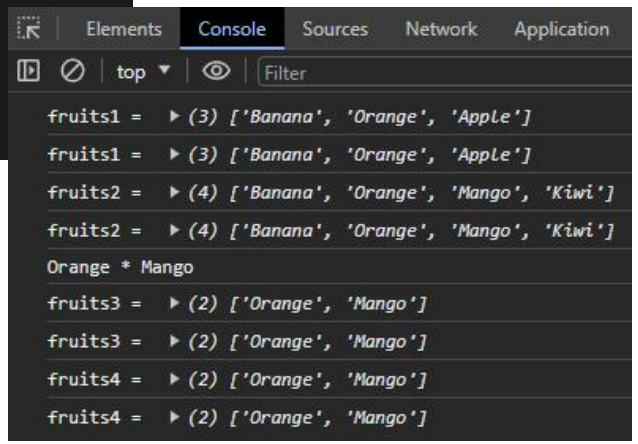
Подробнее про типы: массивы

```
const fruits = ["Banana", "Orange", "Apple"];
fruits[2] = "Mango";
fruits.push("Kiwi");
let fruit = fruits.pop();
fruits.shift();
fruits.join(" * ");
```

- **concat**()
- **slice**()
- **sort**()
- **reverse**()

```
fruits.sort(function(a, b){return a - b});
```

```
const fruits = ["Banana", "Orange", "Apple"];

console.log("fruits1 = ", fruits);

fruits[2] = "Mango";
fruits.push("Kiwi");

console.log("fruits2 = ", fruits);

let fruit = fruits.pop();
fruits.shift();
console.log(fruits.join(" * "));

console.log("fruits3 = ", fruits);

fruits.sort(function (a, b) {
  return a - b;
});

console.log("fruits4 = ", fruits);
```

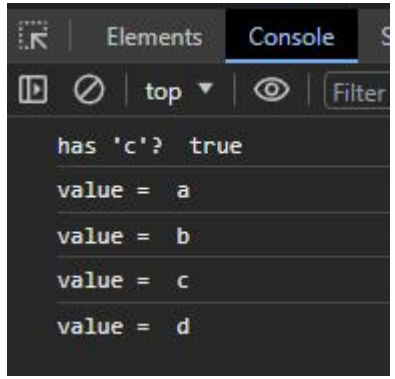


Подробнее про типы: массивы

- Особый класс **Set**

```
const letters = new Set(["a","b","c"]);
letters.add("d");
letters.has("c")  // true
letters.forEach( (value) => { … })
```

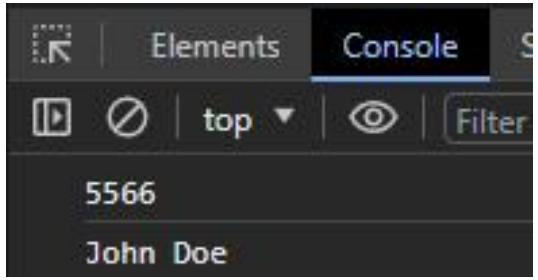
```
const letters = new Set(["a", "b", "c"]);

letters.add("d");
console.log("has 'c'? ", letters.has("c"));
letters.forEach((value) => {
  console.log("value = ", value);
});
```



Подробнее про типы: объекты

```
const person = {
  firstName: "John",
  lastName : "Doe",
  id: 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

```
const person = {
  firstName: "John",
  lastName: "Doe",
  id: 5566,
  fullName: function () {
    return this.firstName + " " + this.lastName;
  },
};

console.log(person["id"]);
console.log(person.fullName());
```

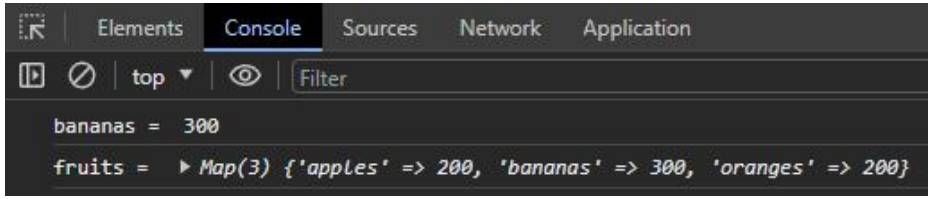


Подробнее про типы: объекты

- Особый класс **Map**

```
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);
fruits.set("apples", 200);
fruits.get("bananas");    // 300
```

```
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200],
]);
fruits.set("apples", 200);
console.log("bananas = ", fruits.get("bananas"));
console.log("fruits = ", fruits);
```



Подробнее про типы: функции

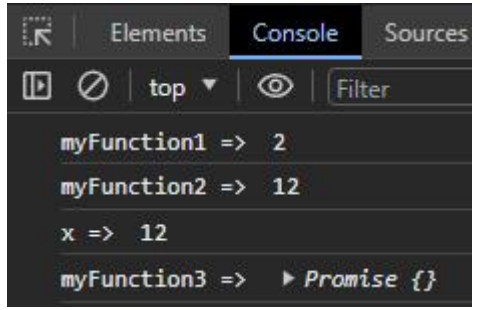
- ***Function declaration***

```
function myFunction(p1, p2) {
  return p1 * p2;
}
```

- ***Function expression***

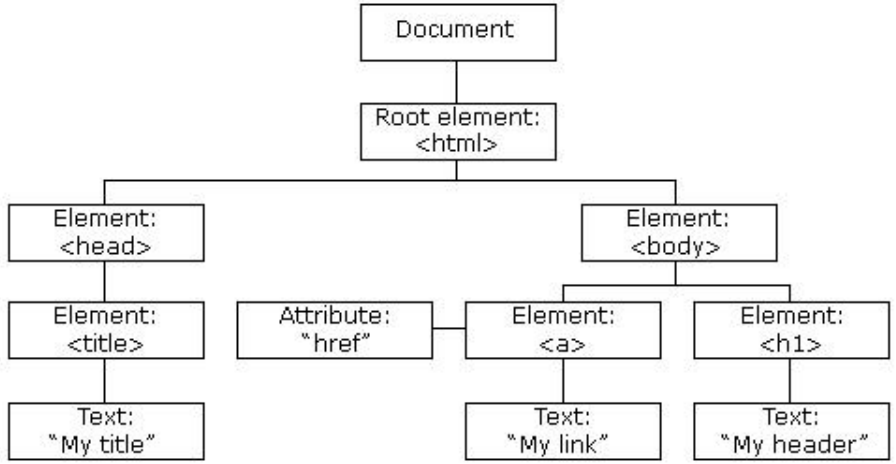
```
myFunction = (p1, p2) => { return p1 * p2; }

let x = myFunction(4, 3);

async function myFunction() {
  return "Hello";
}
```

```
function myFunction1(p1, p2) {
  return p1 * p2;
}

const myFunction2 = (p1, p2) => {
  return p1 * p2;
};

let x = myFunction2(4, 3);

async function myFunction3() {
  return "Hello";
}

console.log("myFunction1 => ", myFunction1(1, 2));
console.log("myFunction2 => ", myFunction2(3, 4));
console.log("x => ", x);
console.log("myFunction3 => ", myFunction3());
```



Взаимодействие с DOM

- **document.getElementById**("demo").innerHTML = "Hello World!";



Взаимодействие с DOM

Обращение к элементу:

- **getElementById**(“someElemID”);
- **getElementsByTagName**(“div”);
- **getElementsByClassName**(“someElemClassName”);

Добавление и удаление:

- document.**createElement**(*element*);
- document.**appendChild**(*element*);
- document.**replaceChild**(*new, old*);
- document.**removeChild**(*element*);
- document.**write**(*text*);

Переходы по DOM

- document.body
- parentNode
- childNodes[nodenumber]
- firstChild
- lastChild
- nextSibling
- previousSibling

Обработка событий

```
<h1 id=“hdr1” onclick="changeText(this)">Click!</h1>

<script>
function changeText(element) {
  element.innerHTML = "Ooops!";
}
</script>

<script>
document.getElementById("hdr1 ").onclick = changeText;
</script>
```

Основные события

- **onload, onunload**
- **onchange**
- **onmouseover, onmouseout**
- **onclick**

element.**addEventListener**(“**click**”, myFunction);

Глобальные объекты

- **window** – объект окна браузера
- **document** – корень DOM
- **location** – адресная строка
- **history** – история браузера
- **console** – консолька браузера для отладки
- **screen** – экран

И это ещё не все!

Дополнительная литература

1. What is Frontend Development?

1. JavaScript Roadmap