Введение в JS

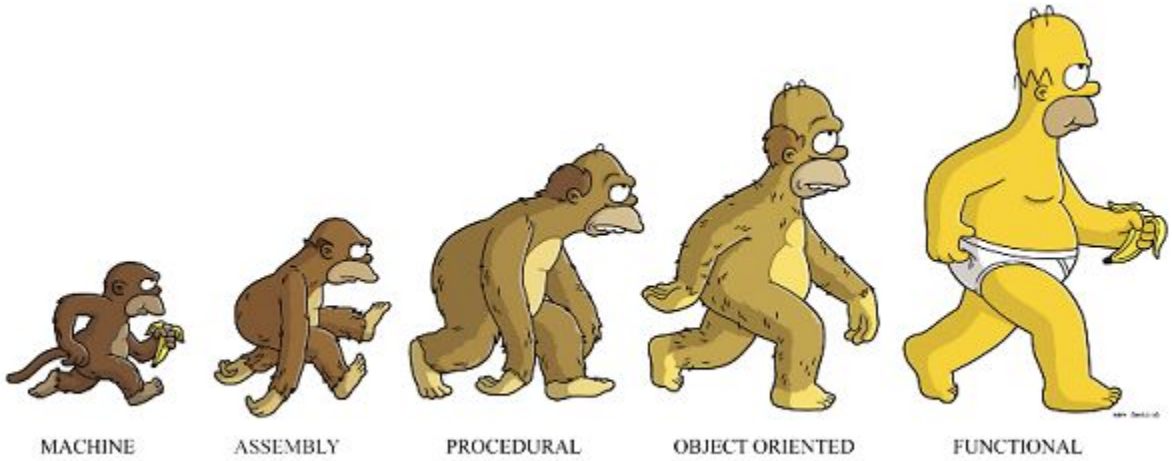
План лекции:

- Описание Javascript
- Стандарт ECMAScript
- Синтаксис языка
  - Типы данных
  - Переменные и константы
  - Условные и циклические выражения
  - Функция и область видимости
  - Подключение скриптов
  - Pascal vs Javascript

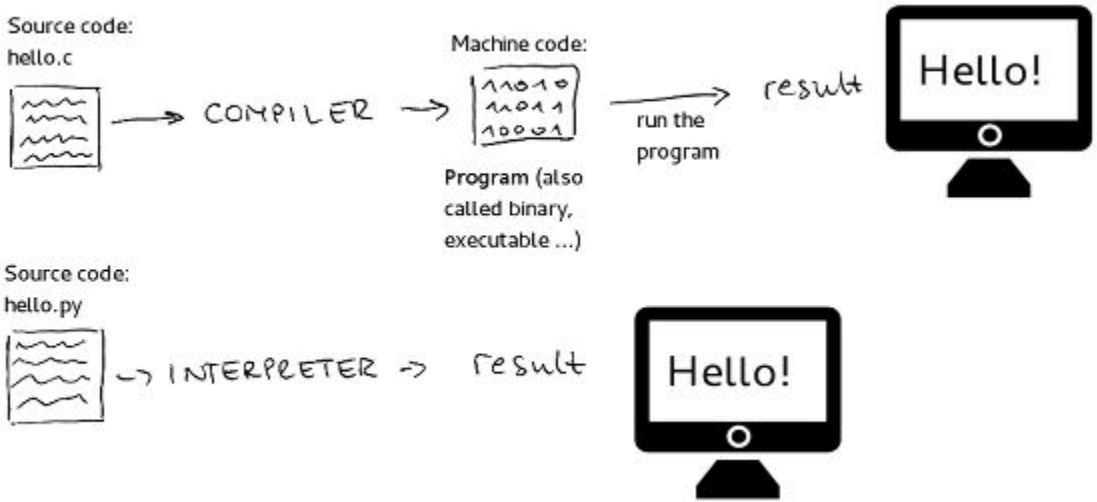
Javascript

<footer>3</footer>

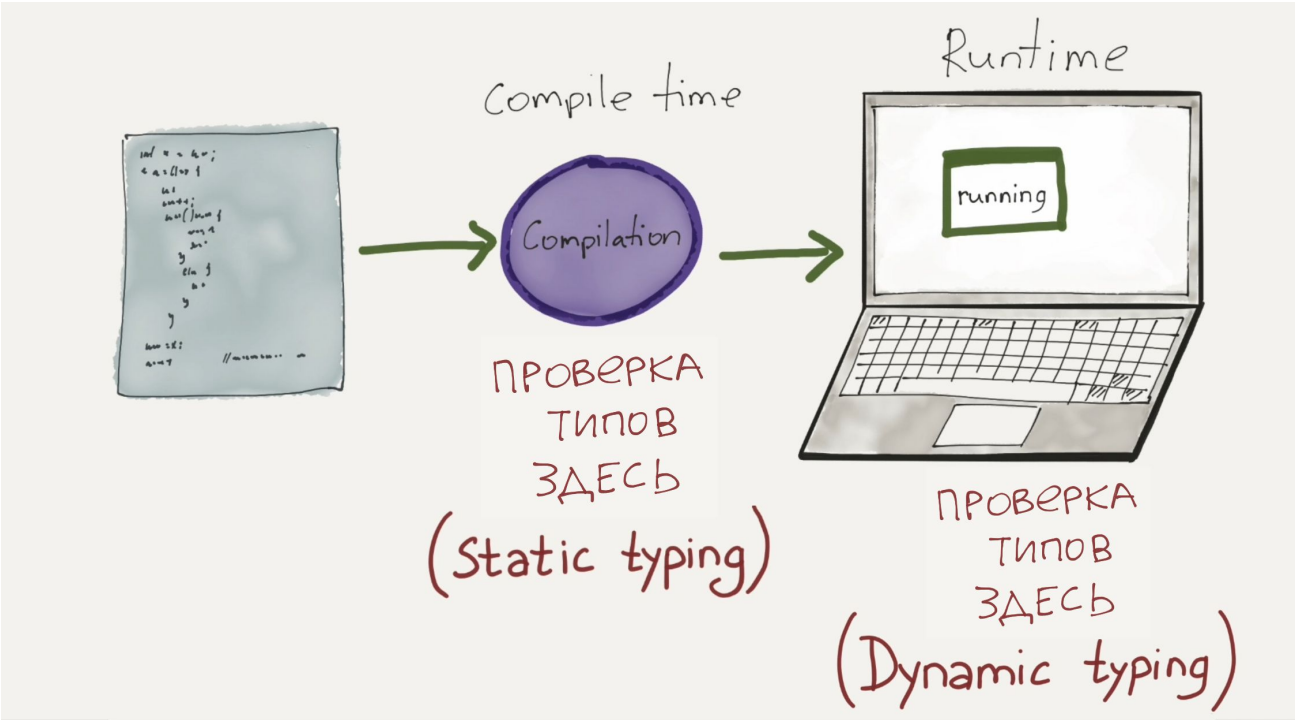
JS - Мультипарадигменный



JS - Интерпретируемый



JS - динамически типизированный



JS - динамически типизированный

```
/* Pascal */
Program StaticTyping;
Var
    name: string;
Begin
    name := "John";
    Writeln(name);
    name := 34;
End.
```

```
/* Javascript */
let name = 'John'
console.log(name)
name = 34
```

Javascript - это реализация стандарта ECMAScript

Зачем нужен стандарт ECMAScript?

Движок Javascript

- V8
- Rhino
- SpiderMonkey
- Javascript Core

Разные браузеры - разные движки Javascript

	Browser	Scripting Engine
1	Chrome	V8 (C++)
2	Mozilla Firefox	SpiderMonkey (C/C++)
3	IE Edge	Chakra JavaScript engine (C++)
4	Opera	V8 (C++)
5	Internet Explorer	Chakra JScript engine (C++)
6	Apple Safari	JavaScript Core (Nitro)

Node.js

Пример приложений на JS

https://www.hellomonday.com/

http://www.narrowdesign.com/

https://drive.google.com/drive/my-drive

https://www.pinterest.ru/

https://www.figma.com/

https://techrocks.ru/2018/05/20/web-sites-and-apps-built-with-node-js/

Типы данных

- Число “number”
- Число “bigint”
- Строка “string”
- Булевый (логический) тип “boolean”
- Специальное значение “null”
- Специальное значение “undefined”
- Объекты “object”

Оператор typeof

```
> typeof undefined
"undefined"
> typeof 0
"number"
> typeof 1n
"bigint"
> typeof true
"boolean"
> typeof 'hello'
"string"
>
```

```
> typeof Symbol()
"symbol"
> typeof {}
"object"
> typeof null
"object"
> typeof function() {}
"function"
> typeof []
"object"
>
```

Пример конкатенации разных типов

```
> 'hello' + ' world'
"hello world"
> 'hello' + 123
"hello123"
> 'hello' + 123n
"hello123"
> 'hello' + true
"hellotrue"
> 'hello' + Symbol()
▶ Uncaught TypeError: Cannot convert a Symbol value to a string
    at <anonymous>:1:9
```

Пример конкатенации разных типов

```
> 'hello' + function() {}
< "hellofunction() {}"
> 'hello' + []
< "hello"
> 'hello' + [1, 2, 3]
< "hello1,2,3"
> 'hello' + {}
< "hello[object Object]"
> 'hello' + null
< "hellonull"
> 'hello' + undefined
< "helloundefined"
```

Переменные - *var*

```
console.log(str)  // undefined
var str = 'hello'
console.log(str)  // 'hello'
```

Переменные - *let*

```
console.log(str)  // Uncaught ReferenceError: str is not defined
let str = 'hello'
console.log(str)
```

Константы - ***const***

```
const str = 'hello'
console.log(str)  // 'hello'

-------------------------------------------------------------

console.log(str)  // Uncaught ReferenceError: Cannot access 'str'
before initialization
const str = 'hello'
console.log(str)
```

Числовые типы данных - ***number | bigint***

```
/* number */
// 2.5x^2 + 5x - 7.2 = 0
const a = 2.5
const b = 5
const c = -7.2
const D = b * b - 4 * a * c
console.log('D =', D) // 97
```

```
/* bigint */
// 2x^2 + 5x - 7 = 0
const a = 2n
const b = 5n
const c = -7n
const D = b * b - 4n * a * c
console.log('D =', D) // 81n
```

bigint

```
> const num = 1.5n
Uncaught SyntaxError: Invalid or unexpected token
```

```
> const bigint = 15n
undefined
> bigint
15n
> bigint + 5
Uncaught TypeError: Cannot mix BigInt and other types, use explicit conversions
    at <anonymous>:1:8
```

Строковый тип данных - ***string***

```
const hello = 'Hello,'
const world = " World!"
console.log(hello + world) // 'Hello, World!'

hello[0] // 'H'
```

Булевый (логический) тип - ***boolean***

```
const disabled = true
const checked = false

let a = 1
let b = 2
console.log(a == b) // равно
console.log(a < b)  // меньше
console.log(a <= b) // меньше или равно
console.log(a > b)  // больше
console.log(a >= b) // больше или равно
```

Специальные значения - *null* | *undefined*

```
const nullable = null
```

```
// явная инициализация
undefined-ом
const notInitialized = undefined

// получение несуществующего
элемента
const arr = []
console.log(arr[0]) // undefined
```

Массив - ***Array***

```
const arr = [1, 2, 3]
console.log('arr:', arr, 'arr[0]:', arr[0])

arr.push(4)
arr.push(5, 6, 7)
console.log('arr:', arr, 'length:', arr.length)

arr.pop()
console.log('arr:', arr)
```

Способы объявления массива

```
const array = [];
const array2 = [0, false, ''];
const array3 = new Array();
const array4 = new Array(1,2);
const array5 = new Array(3);
const array6 = Array.from('text'); //array = ['t','e','x','t']
const array7 = array6.slice();
const array8 = [...array6];
const array9 = (...rest) => rest;
const array10 = Array.from([1, 2, 3, 4], x => x ** 2);
const array11 = array6.map(elem =>elem ** 2);
const array12 = array6.filter(char => char.length < 2);
const array13 = array6.concat();
```

Объекты - ***Object***

```
const obj = {
    name: 'John',
    age: 25,
}
console.log(Object.keys(obj))   // ['name', 'age']
console.log(Object.values(obj)) // ['John', 25]

obj['surname'] = 'Smith'
console.log(Object.keys(obj))   // ['name', 'age', 'surname']
console.log(Object.values(obj)) // ['John', 25, 'Smith']
```

Условные выражения

```
if (a > 0) {
  console.log('positive')
} else {
  console.log('NOT positive')
}
```

```
if (a < 0) {
  console.log('negative')
} else if (a == 0) {
  console.log('zero')
} else {
  console.log('positive')
}
```

Тернарный оператор

```
(a > 0)
  ? console.log('positive')
  : console.log('NOT positive')
```

```
if (a > 0) {
  console.log('positive')
} else {
  console.log('NOT positive')
}
```

Циклические выражения - *while / do..while*

```
let n = 0
let x = 0

while (n < 3) {
  n++
  x += n
}
```

```
let n = 0
let x = 0

do {
  n++
  x += n
} while (n < 3)
```

Циклические выражения - *for*

```
for (let i = 0; i < 9; i++) {
  console.log(i)
}
```

```
for (;;) {
  // будет выполняться вечно
}
```

Цикл *for...in*

```
const obj = {
    name: 'John',
    age: '25',
}

for (let key in obj) {
    console.log('obj.' + key, '=', obj[key])
}
// obj.name = John
// obj.age = 25
```

Цикл *for...of*

```
const obj = {
    name: 'John',
    age: 25,
}

for (let value of Object.values(obj)) {
    console.log(value) // John, 25
}
```

break и *continue*

```
let i = 0
while (i < 6) {
  if (i === 3) {
    break
  }
  i++
}
console.log(i) // 3
```

```
let i = 0
let n = 0
while (i < 5) {
   i++
   if (i === 3) {
    continue
   }
   n += i
} // 0 + 1 + 2 + 3 + 4 + 5 = 15
console.log('i: ', i, 'n: ', n) // 12
```

Функция

```
function factorial(n) {
  if (n < 2) {
      return 1
  } else {
      return n * factorial(n - 1)
  }
}

factorial(5) // 120
```

Область видимости

```
<!DOCTYPE html>
<html>
<head>
 <script src="init.js"></script>
 <script src="concat.js"></script>
</head>
</html>
```

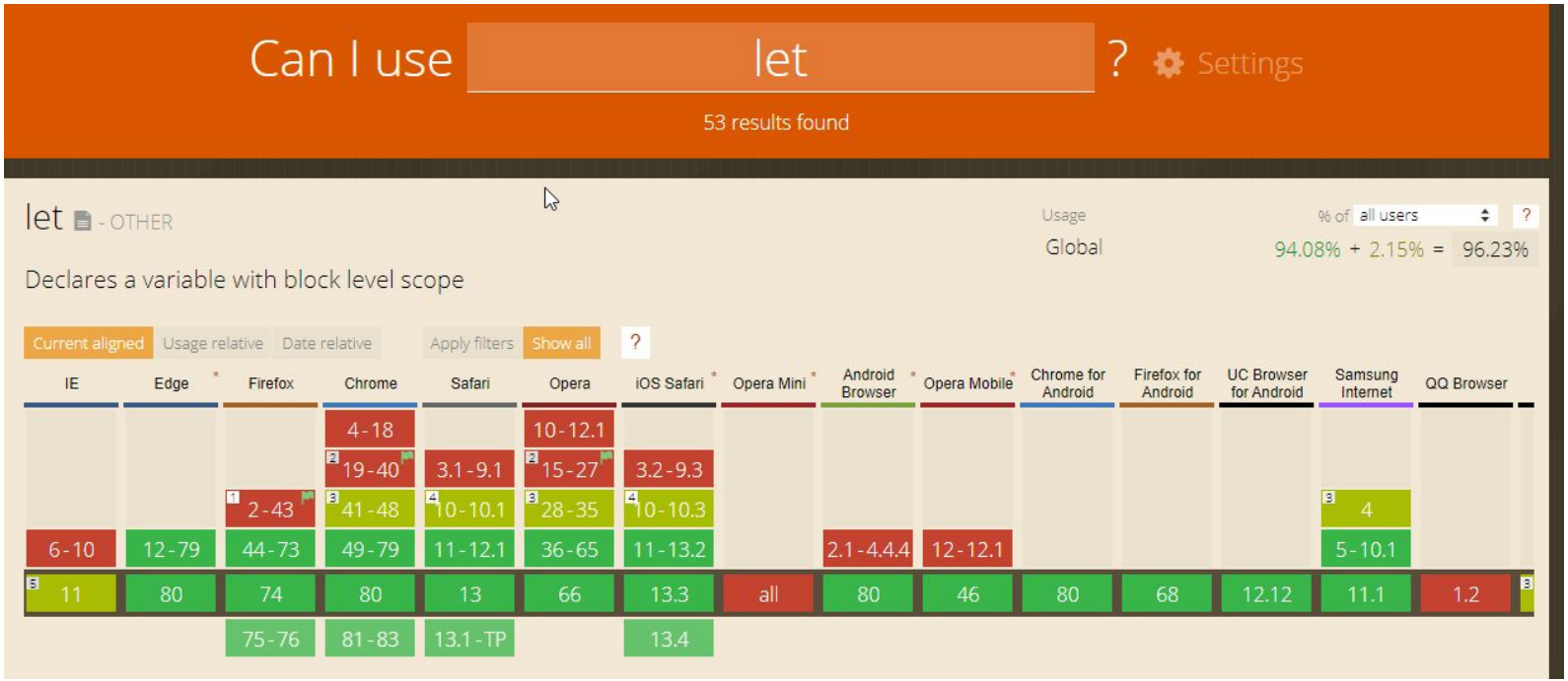
```
/* init.js */
let str = 'Hello,'
console.log('script1 str:', str)
// Hello,
----------------------------------
/* concat.js */
str += ' World!'
console.log('script2 str: ', str)
// Hello, World!
```

Область видимости

```
<!DOCTYPE html>
<html>
<head>
 <script src="init.js"></script>
 <script src="concat.js"></script>
</head>
</html>
```

```
/* init.js */
{
  let str = 'Hello,'
  console.log('script1 str:', str)
}
// Hello,
-----------------------------------
/* concat.js */
str += ' World!'
// Uncaught ReferenceError: str is
not defined
```

Can I use


Can I use
let
?
Settings
53 results found
let - OTHER
Declares a variable with block level scope
Usage
Global
% of all users
94.08% + 2.15% = 96.23%
Current aligned
Usage relative
Date relative
Apply filters
Show all
IE
Edge
Firefox
Chrome
Safari
Opera
iOS Safari
Opera Mini
Android Browser
Opera Mobile
Chrome for Android
Firefox for Android
UC Browser for Android
Samsung Internet
QQ Browser
6-10
11
12-79
80
2-43
44-73
75-76
4-18
19-40
41-48
49-79
81-83
3.1-9.1
10-10.1
11-12.1
13.1-TP
10-12.1
15-27
28-35
36-65
3.2-9.3
10-10.3
11-13.2
13.4
all
2.1-4.4.4
80
12-12.1
46
80
68
12.12
4
5-10.1
11.1
1.2

Подключение внешних скриптов

```
<!DOCTYPE html>
<html>
  <head>
      <script src="init.js"></script>
      <script src="concat.js"></script>
  </head>
</html>
```

Подключение внешних скриптов - ***async*** | ***defer***

```
<!DOCTYPE html>
<html>
  <head>
      <script src="init.js" async></script>
      <script src="concat.js" defer></script>
  </head>
</html>
```

Pascal vs Javascript

```
program Primes;
var
    isPrime: boolean;
    i, j, n: integer;
begin
    n := 10;
    for i := 2 to n do
    begin
        isPrime := true;
        for j := 2 to i - 1 do
            if i mod j = 0
            then
                begin
                    isPrime := false;
                    break;
                end;
        if isPrime = true
        then
            writeln(i);
    end
end.
```

```
let n = 10
let isPrime
for (let i = 2; i <= n; i++) {
    isPrime = true
    for (let j = 2; j < i; j++) {
        if (i % j == 0) {
            isPrime = false
            break
        }
    }
    if (isPrime) {
        console.log(i)
    }
}
```