ES6

我们将迎来最新的JavaScript核心语言标准。

ES6是一次重大的版本升级，与此同时，由于ES6秉承着最大化兼容已有代码的设计理念，你过去编写的JS代码将继续正常运行。

Block-Scoped Variables

```
function varTest() {
  var x = 1;
  if (true) {
    var x = 2;  // same variable!
    console.log(x);  // 2
  }
  console.log(x);  // 2
}
```

```
function letTest() {
  let x = 1;
  if (true) {
    const PI = 3.141593
    let x = 2;
    console.log(x);  // 2
  }
  console.log(x);  // 1
  console.log(PI); //PI is not defined
}
```

ES6 用 let 代替 var

```
var a = [];
(function () {
   'use strict';
   for (let i = 0; i < 5; ++i) { // *** `let` works as expected ***
     a.push( function() {return i;} );
   }
} ());
console.log(a.map( function(f) {f()} ));
// prints [0, 1, 2, 3, 4]

// Start over, but change `let` to `var`.
// prints [5, 5, 5, 5, 5]
```

Default Parameter Values

```
function f (x, y = 7, z = 42) {
    return x + y + z
}
f(1) === 50
```

Rest Parameter

```
function f (x, y, ...a) {
    return (x + y) * a.length
}
f(1, 2, "hello", true, 7) === 9
```

Spread Operator

```
// 单个Spread Operator
function myFunction(x, y, z) { }
var args = [0, 1, 2];
myFunction(...args);

//多个Spread Operator
function myFunction(v, w, x, y, z) { }
var args = [0, 1];
myFunction(-1, ...args, 2, …[3]);

//Copy an array
var arr = [1,2,3];
var arr2 = [...arr]; // like arr.slice()
arr2.push(4); // arr2 becomes [1,2,3,4], arr stays unaffected
```

二者区别

```
// Spread Operator
var abc = ['a', 'b', 'c'];
var def = ['d', 'e', 'f'];
var alpha = [ ...abc, ...def ];
// alpha == ['a', 'b', 'c', 'd', 'e', ‘f'];


// Rest Parameter
function sum( first, ...others ) {
    for ( var i = 0; i < others.length; i++ )
        first += others[i];
    return first;
}
// sum(1, 2, 3, 4) == 10;
```

Value Export/Import

```
//  lib/math.js
export function sum (x, y) { return x + y }
export var pi = 3.141593

//  someApp.js
import * as math from "lib/math"
console.log("2π = " + math.sum(math.pi, math.pi))

//  otherApp.js
import { sum, pi } from "lib/math"
console.log("2π = " + sum(pi, pi))
```

Default & Wildcard

```
//  lib/mathplusplus.js
export * from "lib/math"
export var e = 2.71828182846
export default (x) => Math.exp(x)

//  someApp.js
import exp, { pi, e } from "lib/mathplusplus"
console.log("e^{π} = " + exp(pi))
```

Class Definition

```
class Shape {
    constructor (id, x, y) {
        this.id = id
        this.move(x, y)
    }
    move (x, y) {
        this.x = x
        this.y = y
    }
}
```

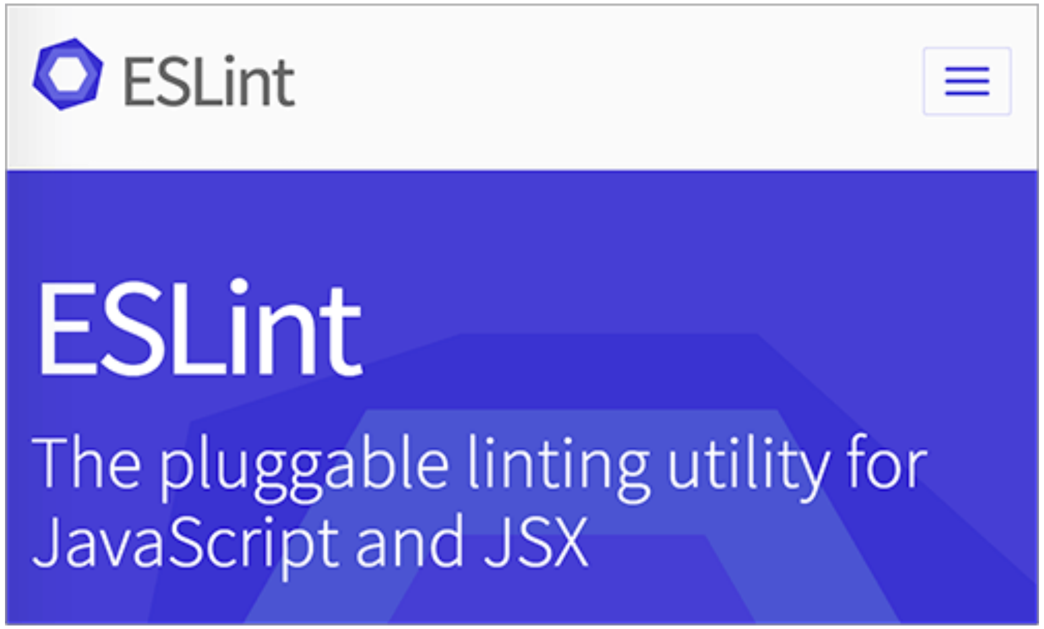
Class Inheritance

```
class Rectangle extends Shape {
    constructor (id, x, y, width, height) {
        super(id, x, y)
        this.width  = width
        this.height = height
    }
}
class Circle extends Shape {
    constructor (id, x, y, radius) {
        super(id, x, y)
        this.radius = radius
    }
}
```

Getter/Setter

```
class Rectangle {
    constructor (width, height) {
        this._width  = width
        this._height = height
    }
    set width  (width)  { this._width = width                }
    get width  ()       { return this._width                 }
    set height (height) { this._height = height              }
    get height ()       { return this._height                }
    get area   ()       { return this._width * this._height }
}
var r = new Rectangle(50, 20)
r.area === 1000
```

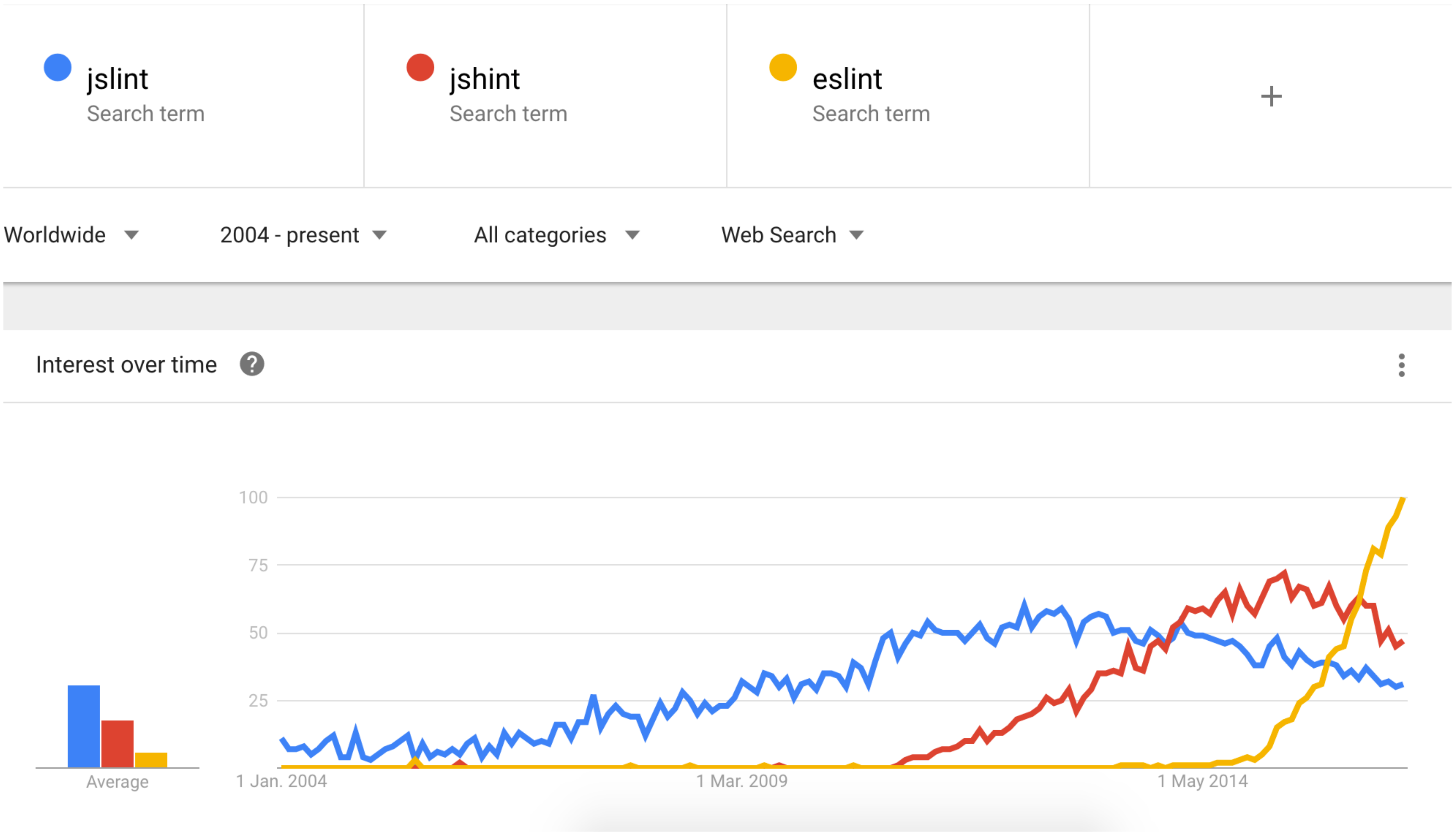
JSLint / JSHint / ESLint



ESLint

ESLint

The pluggable linting utility for JavaScript and JSX



● jslint
Search term

● jshint
Search term

● eslint
Search term

Worldwide ▼ 2004 - present ▼ All categories ▼ Web Search ▼

Interest over time

ESLint配置文件

1 .eslintrc.js
2 .eslintrc.yaml
3 .eslintrc.yml
4 .eslintrc.json
5 .eslintrc
6 package.json

严格模式

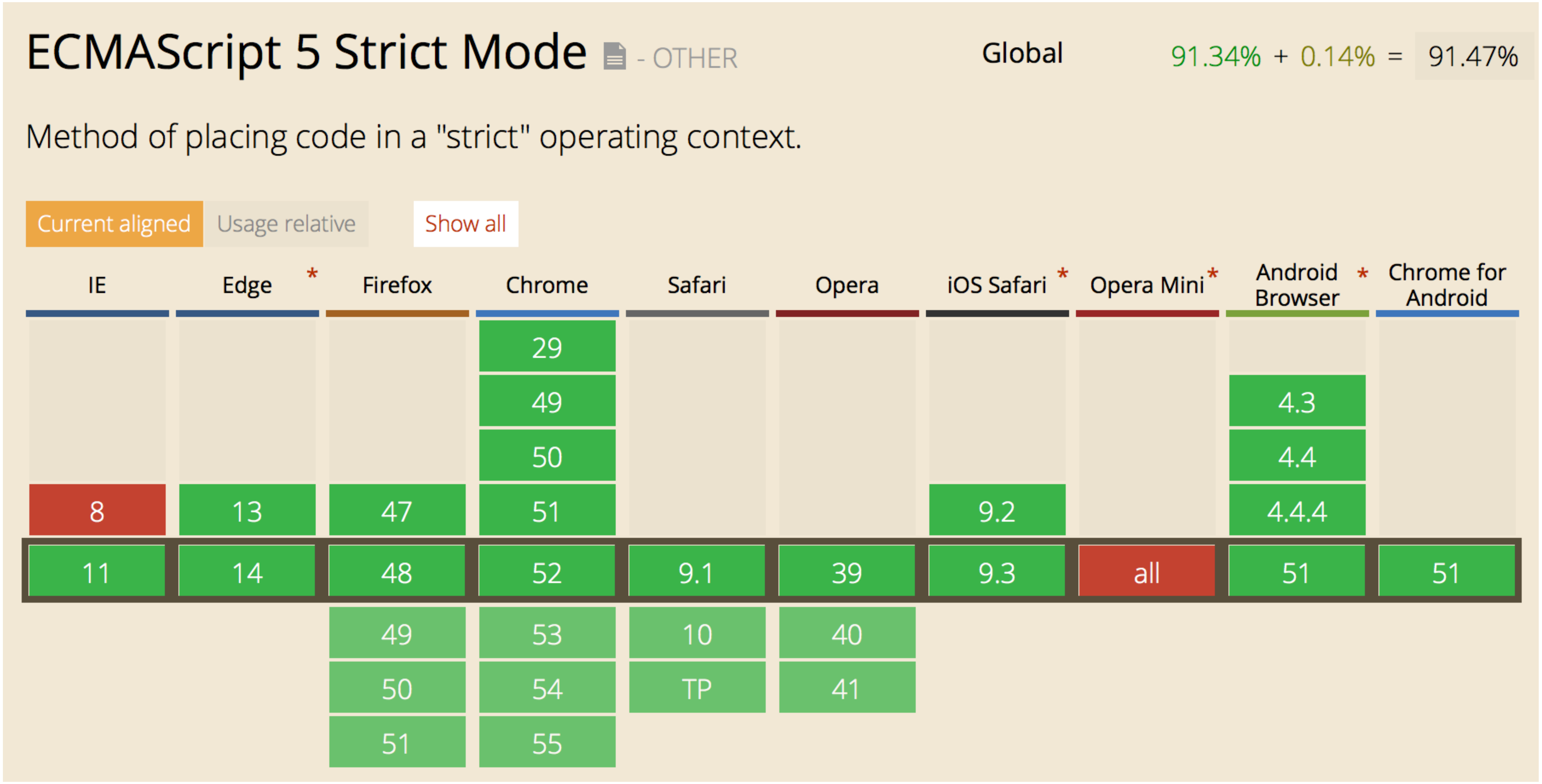
- 从ES5最早引入
- 禁止了一些不安全操作，也可以说会抛出更多异常
- 会禁用一些容易引起混淆的操作
- Fail fast and fail loudly.
- “use strict”;

```
// Non-strict code...

(function(){
  "use strict";

  // Define your library strictly...
})();

// Non-strict code...
```



ECMAScript 5 Strict Mode - OTHER

Global 91.34% + 0.14% = 91.47%

Method of placing code in a "strict" operating context.

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			29						
			49					4.3	
			50					4.4	
8	13	47	51			9.2		4.4.4	
11	14	48	52	9.1	39	9.3	all	51	51
		49	53	10	40				
		50	54	TP	41				
		51	55						

=>

```
(param1, param2, …, paramN) => { statements }
(param1, param2, …, paramN) => expression
         // equivalent to:  => { return expression; }

// 如果只有一个参数，圆括号是可选的:
(singleParam) => { statements }
singleParam => { statements }

// 无参数的函数需要使用圆括号:
() => { statements }
```

更短的函数

```
var a = [
  "Hydrogen",
  "Helium",
  "Lithium",
  "Beryllium"
];

var a2 = a.map(function(s){ return s.length });

var a3 = a.map( s => s.length );
```