ECMASCRIPT

mcapelo@bsolus.pt

... ECMASCRIPT 6

ECMA-WHAT?

ECMAScript - The language of the web

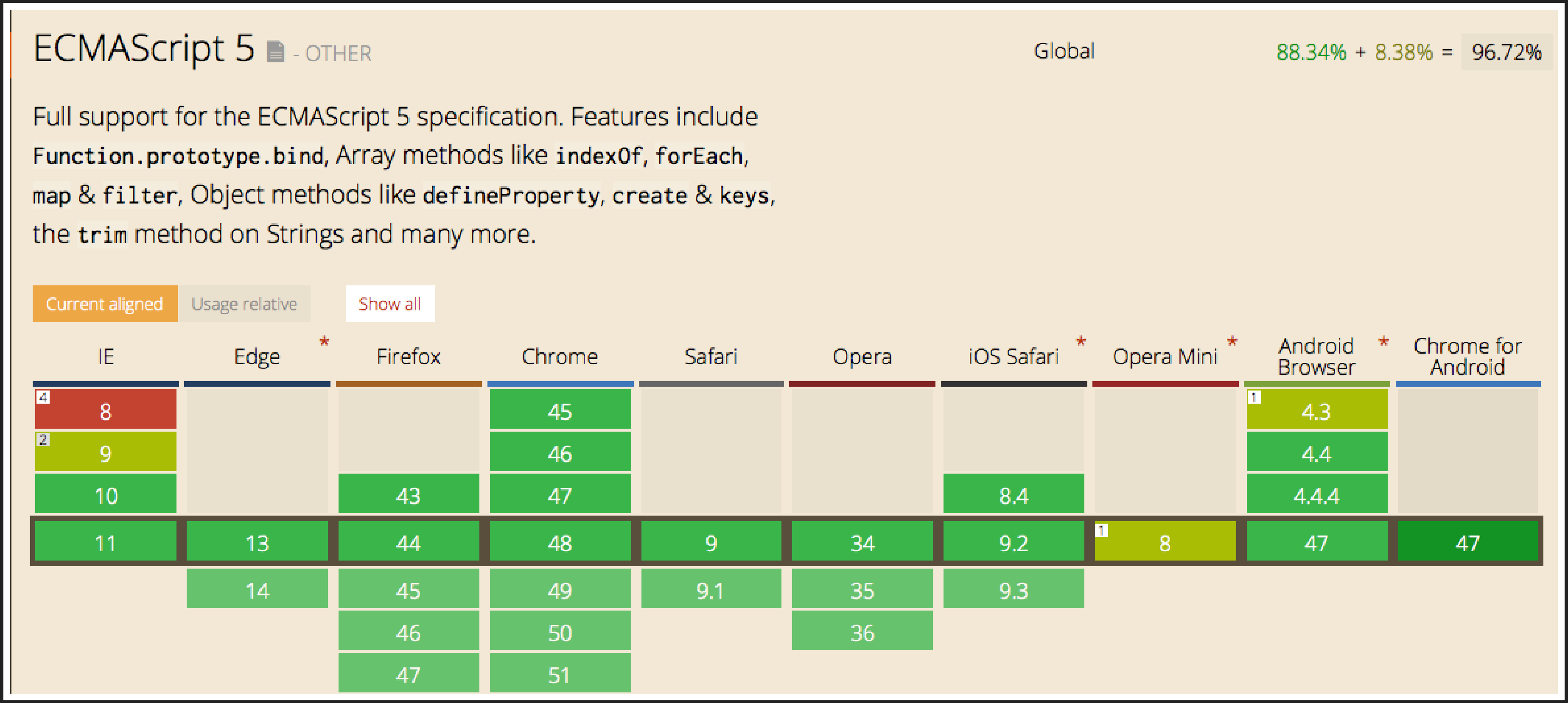
SPECIFICATION THAT DEFINES *SEMANTICS*, *SYNTAX*, AND *BEHAVIOUR* OF THE JAVASCRIPT PROGRAMMING LANGUAGE.

ECMAScript is the scripting language that forms the basis of JavaScript.

WHAT IS GOING ON?

WE ARE VIRTUALLY IN ~~2011~~ 2009 UNDER ES5.1 UMBRELLA

The current "running" version of JavaScript is based on a modification of the original ES5 specification, called ES5.1. ES 5.1 is dated 2011 but it's just a typo/fixing version of ES5 dated 2009. It's the first specification that has been officially adopted in IE ditching the historically slightly different JScript engine used in IE8 and lower.



ECMAScript 5 - OTHER

Global 88.34% + 8.38% = 96.72%

Full support for the ECMAScript 5 specification. Features include `Function.prototype.bind`, Array methods like `indexOf`, `forEach`, `map` & `filter`, Object methods like `defineProperty`, `create` & `keys`, the `trim` method on Strings and many more.

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
[4] 8			45					[1] 4.3	
[2] 9			46					4.4	
10		43	47			8.4		4.4.4	
11	13	44	48	9	34	9.2	[1] 8	47	47
	14	45	49	9.1	35	9.3			
		46	50		36				
		47	51						

More on http://kangax.github.io/compat-table/es5/

SO ...6?

ECMASCRIPT 6 IS THE LATEST STANDARDIZED VERSION OF JAVASCRIPT

(June, 2015)

The ES6 specification introduces a large number of language improvements to Javascript that make it easier to write and understand. There are already a number of features supported in the latest versions of Chrome, Firefox, Safari, and Opera.

SHOW ME

- ## ARROW FUNCTIONS

Lexical scoping of the this keyword + less ceremony when defining an anonymous function

```
function Car() {
  var self = this; //locally assign this that can be closed over
  self.speed = 0;
  setInterval(function goFaster() {
    //this has a different scope
    self.speed += 5;
  }, 1000);
}
```

```
function Car() {
  this.speed = 0;
  setInterval(() => {
    this.speed += 5; //this is from Car
  }, 1000);
}
```

```
var x = [0,1,2];
x.map(function (x) { //anonymous function
  console.log(x * x);
});
```

```
let x = [0,1,2];
x.map(x => console.log(x * x)); //arrow function
```

- ## CLASSES

ES6 introduces language support for *class*, *constructor* and *extends* keywords for inheritance.

Check details here, here, or anywhere else :)

```
function Car( make ) { //approximate a class/constructor
  this.make = make;
  this.currentSpeed = 25;
  this.printCurrentSpeed = function(){ //expose a function
    console.log(this.make + ' is going at ' + this.currentSpeed);
  }
}
```

```
class Car {
  constructor(make) { //constructors!
    this.make = make;
    this.currentSpeed = 25;
  }
  printCurrentSpeed(){
    console.log(this.make + ' is going at ' + this.currentSpeed);
  }
}

class RaceCar extends Car { //inheritance
  constructor(make, topSpeed) {
    super(make); //call the parent constructor with super
    this.topSpeed = topSpeed;
  }
  goFast(){
```

- **MODULES**

Provide a way to load and manage dependencies via the new import and export keywords

One module per file, one file per module.

Named exports

```
//------ lib.js ------
export const sqrt = Math.sqrt;
export function square(x) {
    return x * x;
}
export function diag(x, y) {
    return sqrt(square(x) + square(y));
}

//------ main.js ------
import { square, diag } from 'lib'; // or import all: import * as lik
console.log(square(11)); // 121
```

Single default export

```
//------ MyClass.js ------
export default class { ··· } // no semicolon!

//------ main2.js ------
import MyClass from 'MyClass';
const inst = new MyClass();
```

For a Module:

Top-level variables are ~~global~~ local to module.
Value of this at top level is ~~window~~ undefined.
Executed ~~synchronously~~ asynchronously.

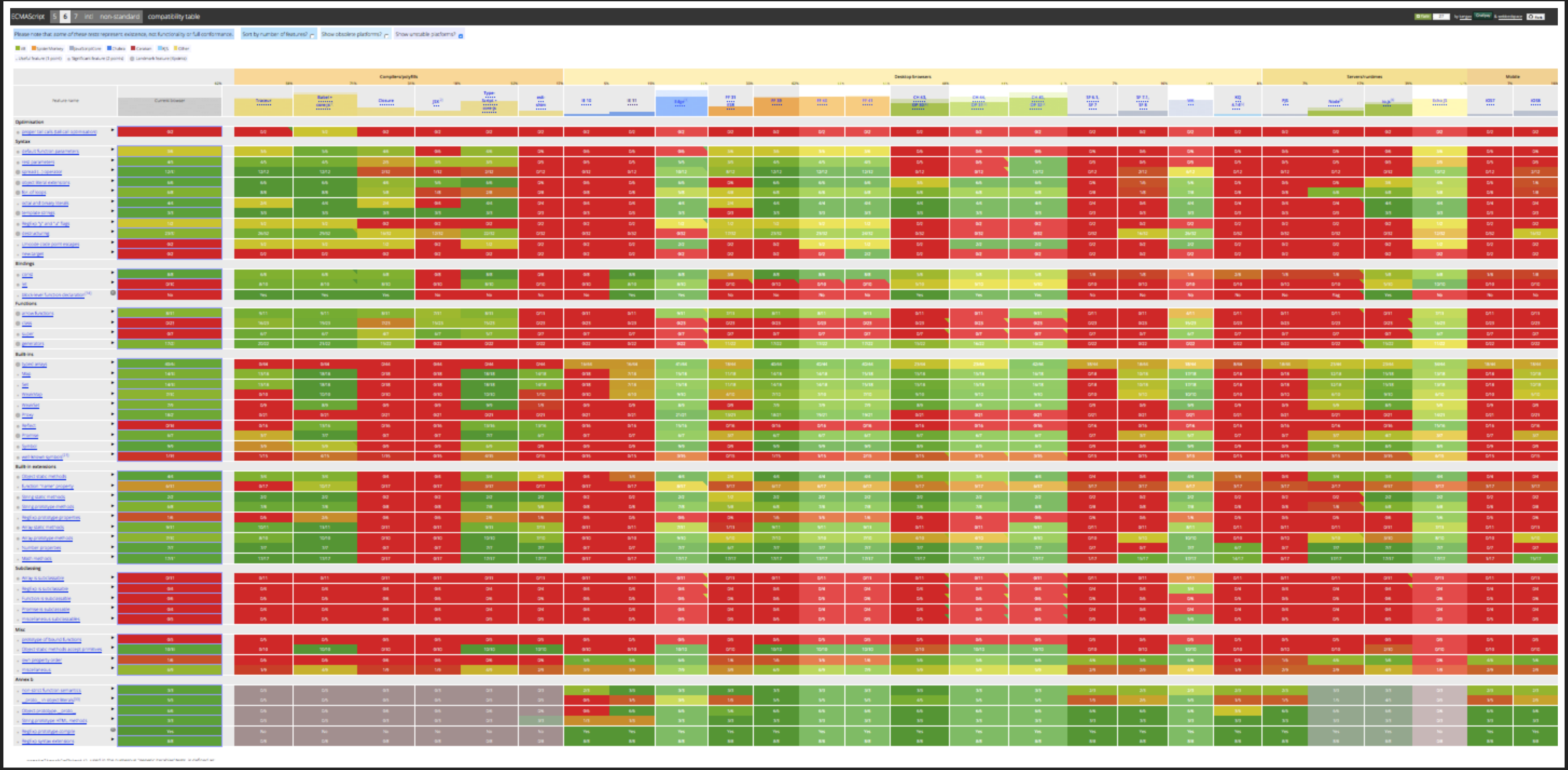
- ## PROMISES

Provide a mechanism to handle the results and errors from asynchronous operations - improved readability via method chaining and succinct error handling.

```
getJSON("/api/employee/1").then(function(post) {
  return getJSON(post.commentURL);
}).then(function(comments) {
  // proceed with access to employee
}).catch(function(error) { //succinct error handling
  // handle errors in either of the two requests
});
```

- DESTRUCTURING
- ITERATORS
- GENERATORS
- ...

BUT



ECMAScript 5 6 7 intl non-standard compatibility table

Please note that some of these tests represent existence, not functionality or full conformance. Sort by number of features? Show obsolete platforms? Show unstable platforms?

Compilers/polyfills

Desktop browsers

Servers/runtimes

Mobile

More on http://kangax.github.io/compat-table/es6/

WHAT TO DO?

TRANSPILERS

ECMAScript 6 to ECMAScript 5

Babel, Traceur

Not perfect

- Most cases lacks feature detection – the ES6 code gets fully converted to ES5.
- Does not allow testing the performance and validity of the ES6 implementation in the browser.
- Debugging made harder - transpiled code is optimised for performance, not for readability. We need source maps.
- May generate bigger files than needed.

NEVERTHELESS, IT'S HERE.

OR IT IS COMING.

DID I HEAR ECMASCRIPT7?

working draft

THANK YOU