ECMAWat?!

Paul Verbeek

Booking.com

workingatbooking.com

nlhtml5.org

ECMAScript

ECMAScript?

"ECMAScript was always an unwanted trade name that sounds like a skin disease.

Brendan Eich,
uitvinder van JavaScript

https://mail.mozilla.org/pipermail/es-discuss/2006-October/000133.html

ECMA-262

ECMA-420

Dart programming language

ECMA-222

Adaptive Lossless Data Compression Algorithm

ECMA-74

Measurement of Airborne Noise Emitted by Information Technology and Telecommunications Equipment

ECMA-262

ECMAScript® Language Specification

ActionScript

Adobe ExtendScript

JScript

V8
SpiderMonkey
Chakra

JavaScript

ECMA-262 6th edition

The ECMAScript® 2015 Language Specification

ES6

ES6

- **Arrow functions**
- Binary and Octal Literals
- **Block-scoped variables (let / const)**
- **Classes**
- **Default + rest + spread parameters**
- Destructuring
- **Enhanced object literals**
- Generators
- Iterators + for..of
- Map + Set + WeakMap + WeakSet

ES6

- new Math, Number, String, Array, Object APIs
- Modules
- Promises
- Proxies
- Reflect API
- Subclassable Built-ins
- Symbols
- Tail Calls
- **Template strings**
- Unicode (full support)

ES2016

ES2016

- Array.prototype.includes
- Exponentiation Operator

Block-scoped variables

```
function logPaul() {

    if (true) {
        var name = 'Paul Verbeek';
    }

    console.log(name); // Paul Verbeek
}

logPaul();
```

```
function logPaul() {
    var name = undefined;

    if (true) {
        name = 'Paul Verbeek';
    }

    console.log(name); // Paul Verbeek
}

logPaul();
```

```
function logPaul() {

    if (true) {
        let name = 'Paul Verbeek';
    }

    console.log(name); // name is not defined
}

logPaul();
```

```
function logAllThePauls() {
    var pauls = ['Paul Verbeek',
                 'Paul McCartney',
                 'Paul Simon’];


    for (var i = 0; i < pauls.length; i++) {
        console.log(pauls[i]);
    }

    console.log(i); // 3
}

logAllThePauls();
```

```
function logAllThePauls() {
    let pauls = ['Paul Verbeek',
                 'Paul McCartney',
                 'Paul Simon’];


    for (let i = 0; i < pauls.length; i++) {
        console.log(pauls[i]);
    }

    console.log(i); // i is not defined
}

logAllThePauls();
```

```
var globalVar = 'global variable';
let globalLet = 'also global';

function f() {
    var functionVar = 'function-scoped';
    let functionLet = 'this one too';
}

f();
```

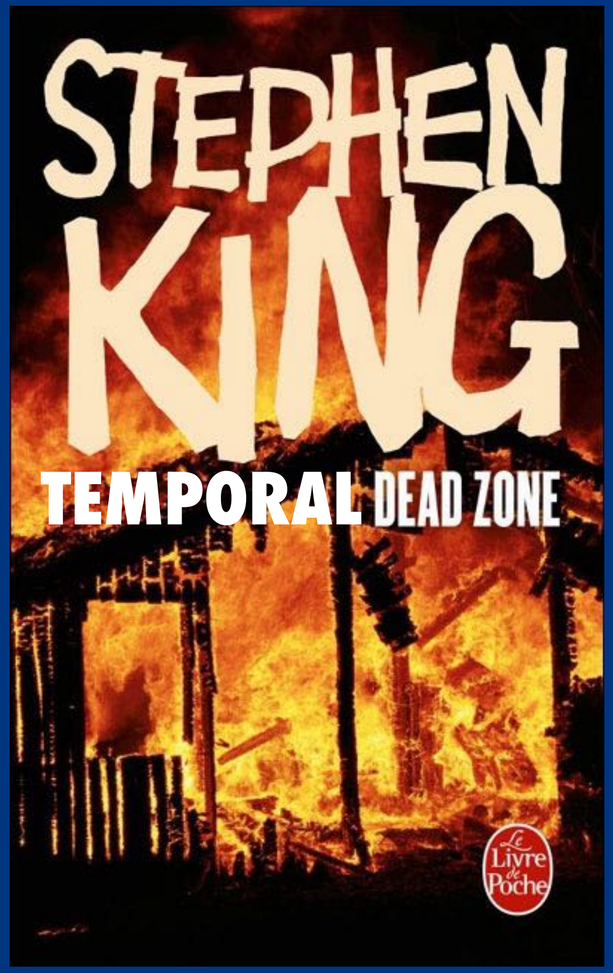
```
function logPaul() {
    let name = 'Paul Verbeek';
    let name = 'Paul McCartney';
    // Identifier 'name' has already been declared

    console.log(name);
}

logPaul();
```

```
function logConstantPaul() {
    const paul = 'Paul Verbeek’;

    paul = 'Paul Simon';
    // 'paul' is read-only

    console.log(paul);
}

logConstantPaul();
```

```
function logConstantPaul() {
    const paul = {
        firstName: 'Paul',
        lastName: 'McCartney'
    };

    paul = 'Verbeek';

    console.log(paul.lastName); // Verbeek
}

logConstantPaul();
```

```
function logConstantPaul() {
    const paul = {
        firstName: 'Paul',
        lastName: 'McCartney'
    };

    paul = {};
    // 'paul' is read-only
}

logConstantPaul();
```

Temporal Dead Zone (TDZ)



```
console.log(typeof foo); // ReferenceError
console.log(typeof bar); // 'undefined'

let foo = 'inner scope’;
var bar = 'inner scope’;
```

```
let foo = 'outer scope';

(function() {
    console.log(typeof foo); // ReferenceError

    let foo = 'inner scope';
}());
```

const? let? ~~var~~?

Gebruik altijd `const`, behalve als je variabele moet kunnen veranderen.

Gebruik anders `let`.

Block-scoped variables




Template strings

```
console.log('In ECMAScript 5.1 is
dit een error.’);

// Uncaught SyntaxError: Unexpected token ILLEGAL


console.log(`In ES6 zal dit
een nieuwe regel geven.`);

// In ES6 zal dit
// een nieuwe regel geven.
```

Een wat?

Een backtick

of accent grave

Dus: `
in plaats van: '

```
var location = 'PFCongres',
    topic = 'ES6';

console.log('Bij ' + location + ' over ' + topic + ' aan het
praten');
// Bij PFCongres over ES6 aan het praten


const location = 'PFCongres',
    topic = 'ES6';

console.log(`Bij ${location} over ${topic} aan het praten`);
// Bij PFCongres over ES6 aan het praten
```

Template strings




Enhanced object literals

```
function getPerson(firstName, lastName, age) {
   return {
        name: `${firstName} ${lastName}`
    };
}


const me = getPerson('Paul', 'Verbeek', 29);

console.log(me);
// {"name":"Paul Verbeek"}
```

```
function getPerson(firstName, lastName, age) {
   return {
        name: `${firstName} ${lastName}`,
        age: age
    };
}


const me = getPerson('Paul', 'Verbeek', 29);

console.log(me);
// {"name":"Paul Verbeek","age":29}
```

```
function getPerson(firstName, lastName, age) {
   return {
        name: `${firstName} ${lastName}`,
        age
    };
}


const me = getPerson('Paul', 'Verbeek', 29);

console.log(me);
// {"name":"Paul Verbeek","age":29}
```

```
function getPerson(firstName, lastName, age) {
    const name = `${firstName} ${lastName}`;

    return { name, age };
}


const me = getPerson('Paul', 'Verbeek', 29);

console.log(me);
// {"name":"Paul Verbeek","age":29}
```

```
function getPerson(firstName, lastName, age) {
   return {
        name: `${firstName} ${lastName}`,
        age,

        makeOlder: function() {
            this.age++;
        }
    };

}


const me = getPerson('Paul', 'Verbeek', 29);

me.makeOlder();

console.log(me);
// {"name":"Paul Verbeek","age":30}
```

```
function getPerson(firstName, lastName, age) {
    return {
        name: `${firstName} ${lastName}`,
        age,

        makeOlder() {
            this.age++;
        }
    };
}


const me = getPerson('Paul', 'Verbeek', 29);

me.makeOlder();

console.log(me);
// {"name":"Paul Verbeek","age":30}
```

```
function getPerson(firstName, lastName, age) {
    return {
        name: `${firstName} ${lastName}`,
        age,

        makeOlder() {
            this.age++;
        },


    };
}


const me = getPerson('Paul', 'Verbeek', 29);

me.makeOlder();

console.log(me);
// {"name":"Paul Verbeek","age":30}
```

```
function getPerson(firstName, lastName, age) {
    return {
        name: firstName + ' ' + lastName,
        age,

        makeOlder() {
            this.age++;
        },

        ['is' + firstName]: true
    };
}


const me = getPerson('Paul', 'Verbeek', 30);

console.log(me.isPaul); // true
```

Enhanced object literals




Arrow functions
(λ expressions)

```
// ES5.1
setTimeout(function() {
    console.log('foo');
}, 100);

var square = function (x) { return x * x; };


// ES6
setTimeout(() => {
    console.log('foo');
}, 100);

const square = x => x * x;
```

1. Minder typen

```
const arr = [1, 2, 3];
let square;

// ES5.1
square = arr.map(function(a) { return a * a; });


// ES6
square = arr.map(a => a * a);
```

2. Niet meer 'var that=this'

```
// ES5.1
function Interface() {
    var that = this;
    var button = document.getElementById('myButton');

    button.addEventListener('click', function() {
        console.log('CLICK');
        that.handleClick();
    });
}

Interface.prototype.handleClick = function() { ... };
```

2. Niet meer 'var that=this'

```
// ES6
function Interface() {

    const button = document.getElementById('myButton');

    button.addEventListener('click', () => {
        console.log('CLICK');
        this.handleClick();
    });
}

Interface.prototype.handleClick = function() { ... };
```

Arrow functions
(λ expressions)

Classes

```
function Person(firstName, lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
}

Person.prototype.fullName = function() {
    return this.firstName + ' ' + this.lastName;
};

Person.prototype.toString = function() {
    return 'Mijn naam is ' + this.fullName();
};

var me = new Person('Paul', 'Verbeek');

console.log(me.toString());
// Mijn naam is Paul Verbeek
```

```
class Person {
    constructor(firstName, lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    get fullName () {
        return `${this.firstName} ${this.lastName}`;
    }

    toString () {
        console.log(`Mijn naan is ${this.fullName}`);
    }
}

const me = new Person('Paul', 'Verbeek');

console.log(me.toString());
// Mijn naam is Paul Verbeek
```

```
class Person {
  ...
}

class Employee extends Person {
    constructor(firstName, lastName, company) {
        super(firstName, lastName);

        this.company = company;
    }

    toString () {
        return `${super.toString()}
            en ik werk bij ${this.company}`;
    }
}

const me = new Employee('Paul', 'Verbeek', 'Booking.com');

console.log(me.toString());
// Mijn naan is Paul Verbeek
// en ik werk bij Booking.com
```

Classes




Default, rest and spread

```
multiply(3); // 6
multiply(3, 3); // 9

function multiply(x, y) {
    if (typeof y === "undefined") { y = 2; }

    return x * y;
}
```

```
multiply(3); // 6
multiply(3, 3); // 9

function multiply(x, y = 2) {


    return x * y;
}
```

```
addOne(3, 7, 14); // [4, 8, 15]

function addOne() {
    var numbers = Array.prototype.slice.call(arguments);

    return numbers.map(function (number) {
        return number + 1;
    });
}
```

```
addOne(3, 7, 14); // [4, 8, 15]

function addOne(...numbers) {
    return numbers.map(number => number + 1);
}
```

```
add(6, 3, 7, 14); // [9, 13, 20]

function add(addition, ...numbers) {
    return numbers.map(number => number + addition);
}
```

```
function myFunction(x, y, z) {
    return x + y + z;
}

var args = [16, 23, 42];
myFunction.apply(null, args); // 81


myFunction(...args); // 81
```

```
var middleNumbers = [15, 16];
var allNumbers = [4, 8, middleNumbers, 23, 42];

// [4, 8, [15, 16], 23, 42]

var allNumbers = [4, 8].concat(middleNumbers, [23, 42]);



const allNumbers = [4, 8, ...middleNumbers, 23, 42];

// [4, 8, 15, 16, 23, 42]
```

Default, rest and spread

https://ponyfoo.com/

http://www.2ality.com/

ECMA-262 6th edition

The ECMAScript® 2015 Language Specification

Waar ik het niet over heb gehad

- Binary and Octal Literals
- Destructuring
- Generators
- Iterators + for..of
- Map + Set + WeakMap + WeakSet
- Modules
- new Math, Number, String, Array, Object APIs
- Promises
- Proxies
- Reflect API
- Subclassable Built-ins
- Symbols
- Tail Calls
- Unicode (full support)

ECMA-262 7th edition

The ECMAScript® 2016 Language Specification

ES2016

exponentiation operator (**)

```
x ** y

Math.pow(x, y);

let a = 3;
a **= 3;
// a = 27
```




Array.prototype.includes

```
['a', 'b', 'c'].indexOf('a') >= 0; // true
['a', 'b', 'c'].indexOf('d') >= 0; // false

['a', 'b', 'c'].includes('a'); // true
['a', 'b', 'c'].includes('d'); // false

['a', 'b', NaN].includes(NaN); // true
['a', 'b', NaN].indexOf(NaN) >= 0; // false

[ , , ].includes(undefined); // true
[ , , ].indexOf(undefined) >= 0; // false
```



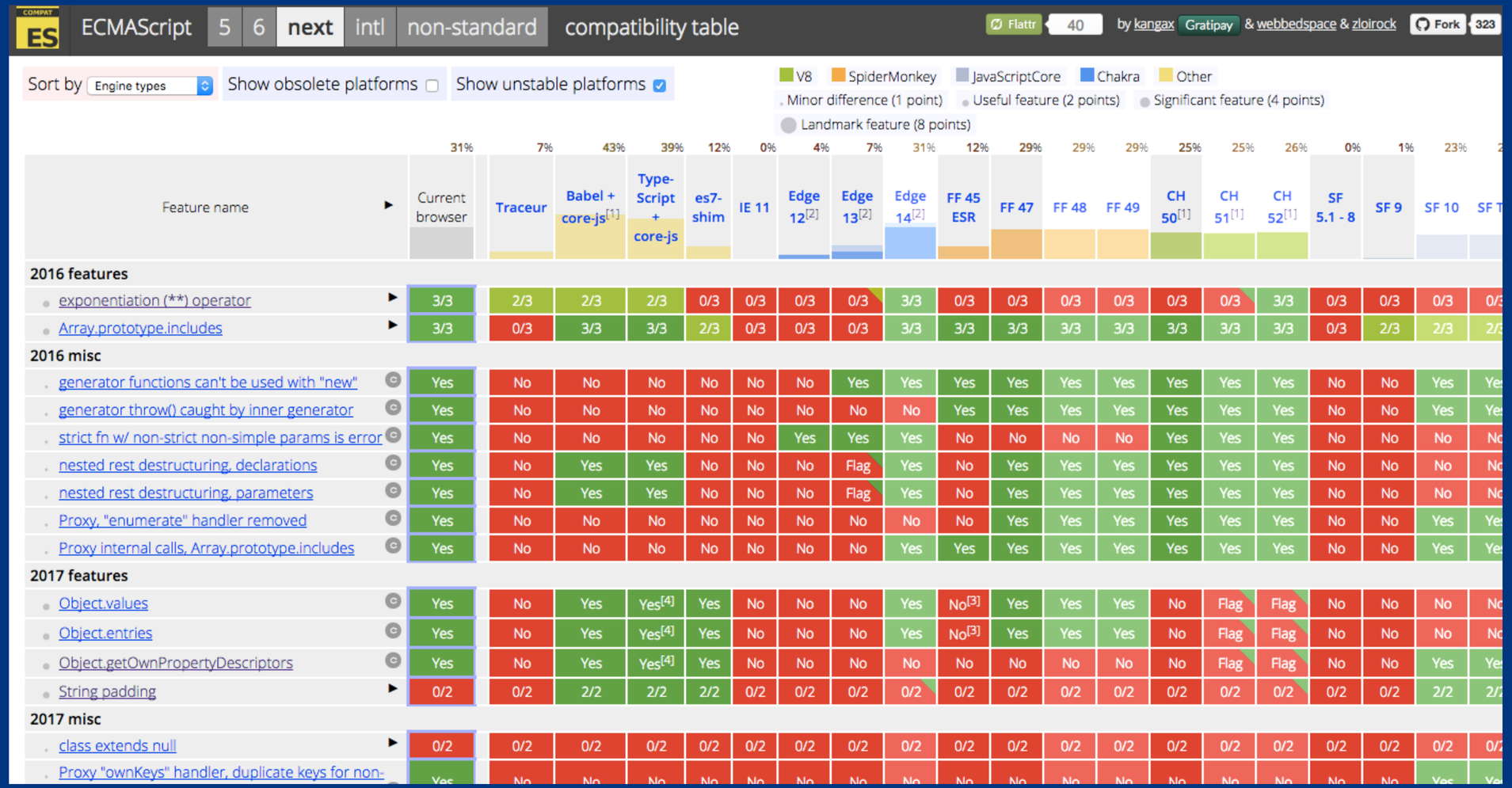

ES2017

ES2017

- Object.values/Object.entries
- String padding
- Object.getOwnPropertyDescriptors
- Trailing commas in function parameter lists and calls
- Async Functions

https://github.com/tc39/ecma262/

https://tc39.github.io/ecma262/



http://kangax.github.io/compat-table/esnext/

Booking.com

workingatbooking.com


Expedia®

 @_paulverbeek

 @nlhtml5

 verbeek.p@gmail.com

 4815162342

Vragen?