const transformJavaScript = (🐛) => 🦋;

Cosmin Stamate
Web Developer @ SmartBill

ECMAScript & JavaScript

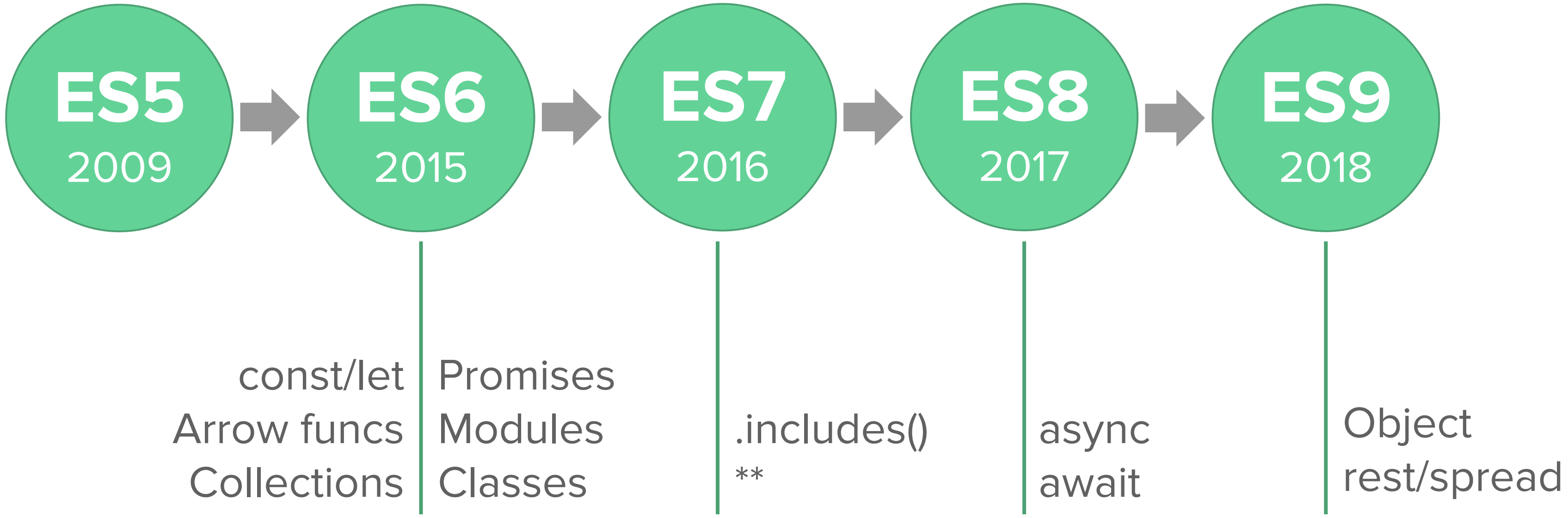
ECMAScript

ECMAScript (or ES) is a scripting-language specification standardized by Ecma International.

JavaScript

A general purpose scripting language that conforms to the ECMAScript specification.

Recent History


ES5
2009
ES6
2015
const/let
Arrow funcs
Collections
Promises
Modules
Classes
ES7
2016
.includes()
**
ES8
2017
async
await
ES9
2018
Object
rest/spread

const / let

```
var a = 3;

while (a > 0) {
    var b = a + 5;
    a--;
    console.log(a, b);
}

console.log(a, b);
// 2 8
// 1 7
// 0 6
// 0 6
```

```
var a = 3;
var b;

while (a > 0) {
    b = a + 5;
    a--;
    console.log(a, b);
}

console.log(a, b);
// 2 8
// 1 7
// 0 6
// 0 6
```

ES6

const / let

```
let a = 3;

while (a > 0) {
    let b = a + 5;
    a--;
    console.log(a, b);
}

console.log(a, b);

// 2 8
// 1 7
// 0 6
// Uncaught ReferenceError:
b is not defined
```

```
let a = 3;
let b;

while (a > 0) {
    b = a + 5;
    a--;
    console.log(a, b);
}

console.log(a, b);

// 2 8
// 1 7
// 0 6
// 0 6
```

```
const a = 3;
let b;

while (a > 0) {
    b = a + 5;
    a--;
    console.log(a, b);
}

console.log(a, b);

// Uncaught TypeError:
Assignment to constant
variable.
```

ES6

Arrow functions

```
const A = {
    prefix: 'abc',

    addPrefix(list) {
        return list.map(function (item) {
            return this.prefix + item;
        });
    },
};

console.log(A.addPrefix(['d', 'e']));

// ["undefinedd", "undefinede"]
```

```
const A = {
    prefix: 'abc',

    addPrefix(list) {
        return list.map((item) =>
            this.prefix + item);
    },
};

console.log(A.addPrefix(['d', 'e']));

// ["abcd", "abce"]
```

ES6

Collections / Map

```
const grades = new Map();
grades.set('John', 10);
grades.set('Anna', 6);
grades.set('Michael', 8);
console.log(grades.get('John')); // 10

grades.delete('John');
console.log(grades.has('John')); // false

console.log(grades);
// Map(2) {"Anna" => 6, "Michael" => 8}
```

ES6

Collections / Set

```
const animals = new Set();
animals.add('lion');
animals.add('zebra');
animals.add('monkey');
console.log(animals); // Set(3) {"lion", "zebra", "monkey"}

animals.delete('lion');
console.log(animals.has('lion')); // false

animals.add('monkey');
console.log(animals); // Set(2) {"zebra", "monkey"}
```

ES6

Promises

```
const promise = new Promise((resolve, reject) => {
    setTimeout(() => resolve('foo'), 1000);
});

promise.then((result) => console.log(result));
// 1 second later: foo

console.log(promise); // Promise {<pending>}
```

ES6

Modules

```
export default function counter () {};

export const CREATE_MESSAGE_URL = '...';
export const FIND_MESSAGE_URL = '...';



export {
    findElement: () => {},
    deleteElement: () => {},
    addElement: () => {}
};
```

```
import counter from 'counter.js';

import {
    CREATE_MESSAGE_URL,
    FIND_MESSAGE_URL,
} from 'constants.js';

import utils from 'element-utils.js';
utils.findElement();
utils.deleteElement();
utils.addElement();
```

ES6

Classes

```
class Animal {
    constructor(species) {
        this.species = species;
    }

    speak() {
        switch (this.species) {
            case 'dog': return 'woof';
            case 'cat': return 'meow';
        }
    }
}

const dog = new Animal('dog');
console.log(dog.speak()); // woof
```

```
class Dog extends Animal {
    constructor() {
        super('dog');
    }
}

class Cat extends Animal {
    constructor() {
        super('cat');
    }
}

const cat = new Cat();
console.log(cat.speak()); // meow
```

ES6

Array.prototype.includes() / Exponentiation operator

```
const list = [1, 2, 3, NaN];

list.includes(1); // true
list.includes(4); // false
list.indexOf(NaN); // -1
list.includes(NaN); // true
```

```
const a = 2;
const b = 10;

console.log(a ** b); // 1024
console.log(2 ** 2); // 4
console.log(b ** 1); // 10
```

ES7

async / await

```
const promise1 = new Promise((resolve) =>
    setTimeout(() => resolve('foo'), 1000));

async function handlePromise(promise) {
    const result = await promise; // result === 'foo'
    return result + 'bar';
}

const promise2 = handlePromise(promise1); // Promise {<pending>}
promise2.then((result) => console.log(result));
// 1 second later:
// foo
// foobar
```

ES8

async / await

```
async function getPostWithComments(postId) {
    const post = await $.ajax(`/api/posts/${postId}`);
    const messages = await $.ajax('/api/messages', {
        data: { postId },
    });

    messages.forEach((message) =>
        message.imagePath = `/images/${postId}` + message.image);

    post.messages = messages;
    return post;
}
```

ES8

Object spread destructuring

```
const object1 = { a: 1, b: 2, c: 3 };
const { a, ...object2 } = object1;
console.log(object2); // {b: 2, c: 3}

const object3 = { ...object1, d: 4 };
console.log(object3); // {a: 1, b: 2, c: 3, d: 4}

const object4 = { a: 5, ...object1, c: 6 };
console.log(object4); // {a: 1, b: 2, c: 6}

const object5 = { ...object2, ...object3 };
console.log(object5); // {b: 2, c: 3, a: 1, d: 4}
```

ES9

Q&A

Thank you!