



# The Future of JavaScript

---

A peek into what's coming and when

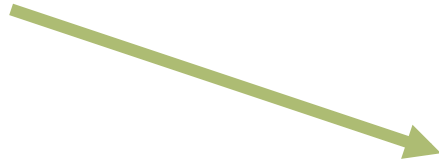
Technology continues to improve but ...



**"The web is an odd place where  
breaking changes can't just be  
hidden behind a version."**

**- Jared Faris**

ECMAScript



**ES**




The year  
that the  
spec was  
solidified



aka.

- ES6
- JavaScript 6
- ES Harmony
- JavaScript 2015
- ES.Next



We're adding  
new features  
every year

Best to stick  
with "ES20XX"  
naming



# The best features of ES2015


|                            | Edge | Firefox | Chrome | Safari | Opera |
|----------------------------|------|---------|--------|--------|-------|
| Arrow functions            | 13   | 43      | 45     | 10     | 35    |
| Block scoping (let, const) | 13   | 44      | 45     | 10     | 35    |
| Default parameters         | 14   | 45      | 49     | 10     | 36    |
| Classes                    | 13   | 45      | 49     | 9      | 36    |
| Promises                   | 12   | 38      | 49     | 9      | 36    |
| New collections (Map, Set) | 12   | 38      | 49     | 9      | 36    |
| New iterators (for-of)     | 12   | 38      | 49     | 9      | 36    |
| String templates (`\${}`)  | 12   | 38      | 49     | 9      | 36    |
| Destructuring              | 14   | 38      | 49     | 9      | 36    |
| Modules (export, import)   | X    | X       | X      | 10.1   | X     |
| Spread & rest              | 13   | 38      | 49     | 10     | 39    |

# The best features of ES2016

|                          | Edge | Firefox | Chrome | Safari | Opera |
|--------------------------|------|---------|--------|--------|-------|
| Exponentiation Operator  | 14+  | 52+     | 54+    | 10.1+  | 43+   |
| Array.prototype.includes | 14+  | 40+     | 54+    | 10+    | 43+   |

# The best features of ES2017

|                             | Edge | Firefox | Chrome | Safari | Opera |
|-----------------------------|------|---------|--------|--------|-------|
| Object static methods       | 15   | 51      | 56     | 10.1   | 43    |
| String padding              | 15   | 51      | 57     | 10     | 44    |
| Trailing commas in function | 14   | 52      | 58     | 10     | 45    |
| Async functions             | X    | 52      | 56     | 10.1   | 43    |
| Atomsics                    | X    | 52      | X      | 10.1   | X     |



# Here's what you can expect to see in each section

1. What the new feature is
2. Why you'd want to use it
3. Syntax
4. Examples comparing traditional vs new



# Default parameters

---

# What and Why

- What
  - Devs can supply default values for function parameters
- Why
  - Because JavaScript is variadic. it is possible to have undefined input parameters which can lead to unexpected results

# Syntax

- Just add default values in the function definition with an equal sign

```
function (a="val1", b="val2" ...) { ... }
```

# Traditional way

```
function foo(first, last, age) {  
  if (! first)  
    first = "John";  
  last = last || "Doe";  
  age = age || getVotingAge();  
  // Do stuff with first, last, and age here  
}
```

- Note: if first is falsey in any way, it'll use "John".

## New way

```
function foo(first="John",  
    last="Doe", age=getVotingAge()) {  
    // Do stuff with first, last, and age here  
}
```

- If you supply a value it'll be used. If not, the default value is.
- Allows you to pass in null, "", 0, or false as valid values and have them used.

# Destructuring

---

# What and Why

- What
  - Use arrays and objects as lvalues for assignments
- Why
  - Fewer lines of code

# Syntax - Object form

- Just put the object on the left side of the =

```
let {prop1:v1, prop2:v2} = someObject;
```

- Where prop1 & prop2 are properties of someObject
- And v1 & v2 will become the new variables.
- Note: You're not creating an object. You're matching based on the shape of the object.



# Syntax - Array form

- Just put the **array** on the left side of the =

```
let [x, y] = someArray;
```

- x and y will be populated with the first and second elements in someArray
- Note: You're not creating an array. You're matching positionally

## New way - array form

```
let [ln1, ln2, ln3, , ln5] = allLines.split('\n');  
console.log(ln1); // I'm line one  
console.log(ln5); // I'm the 5th line
```

- Note that we're ignoring lines four, six, and up.

## New way - object form

```
let {username: user, password: pass} = req.body;  
// same as  
// let user = req.body.username;  
// let pass = req.body.password;  
• or more directly ...  
let {username, password} = req.body;  
// same as  
// let username = req.body.username;  
// let password = req.body.password;
```

# Spread operator

---

... and rest parameters

# What and Why

- What
  - The ... operator allows intelligent interpolation
- Why
  - Less typing and guesswork with more flexibility

# Syntax

Spreads an  
array into  
values

`...someArray`

- In any JavaScript line
- Called "spread operator"
- Because it *spreads* out the array into its parts

Collects  
values into  
an array

`function (...rest) { ... }`

- Only in a function declaration
- Called "rest parameters"
- Because that is the *rest* of the parameters

## Traditional way - Spread

```
var fourTo9 = [4, 5, 6, 7, 8, 9];  
var foo = [1, 2, 3, 10];  
fourTo9.reverse().forEach(x => {  
    foo.splice(3, 0, x);  
});  
// foo now has 1 - 10
```

## New way - Spread

```
const fourTo9 = [4, 5, 6, 7, 8, 9];  
const foo = [1, 2, 3, ...fourTo9, 10];  
// foo now has 1 - 10
```



# Traditional way - Rest parameters

```
function sum() {  
    let total = 0;  
    for (var x=0 ; x<arguments.length ; x++) {  
        total += arguments[x];  
    }  
    return total;  
}
```

# New way - Rest parameters

```
function sum(...nums) {  
  let total = 0;  
  nums.forEach(x => total += x);  
  return total;  
}
```

- arguments is array-like. nums is a true array.

# Property shorthands

---

# What and Why

- What
  - Object properties can be shortened if they're named similarly
- Why
  - Less typing and more concise code

# Syntax

```
const o = { foo, bar, ... }
```

- In a JavaScript object
- Note: no colons (:)

# Traditional way

```
var a = 1; var b = 2;
```

```
var foo = {
```

```
  a: a,
```

```
  b: b,
```

```
  c: function (d) {
```

```
    // do stuff
```

```
  }
```

```
};
```

// foo has two properties that happen to be  
named the same as variables and one function

## New way

```
const a = 1; const b = 2;
```

```
const foo = {
```

```
  a,
```

```
  b,
```

```
  c (d) { // do stuff }
```

```
};
```

```
// foo has two properties that happen to be  
named the same as variables and one function
```

# String templates

---

Borrowing from mustaches/handlebars



# What and Why

- What
  - Create a template for string output using variable substitution.
- Why
  - Less typing. More declarative. More abstract.

# Syntax

- Just use backtics
- Not single quotes
- Substituted values are put in `${var}`

# Traditional way

```
var name = 'Your name is ' + first + ' ' +  
last + '.';  
var url = 'http://us.com/api/messages/' +  
id;
```

## New way

```
var name = `Your name is ${first} ${last}.`  
var url = `http://us.com/api/messages/${id}`
```

# Multiline strings

```
var roadPoem =  
`Then took the other, as just as fair,  
And having perhaps the better claim  
Because it was grassy and wanted wear,  
Though as for that the passing there  
Had worn them really about the same,`;
```

# Modules

---

# What and Why

- What
  - Formalizing creation of JavaScript modules
- Why
  - Organization of hairy code, but mostly ... encapsulation!
- Building on the AMD/CommonJS formats and requireJS library

# Syntax

- To expose an object, function, string, class, whatever  
`export <anyObject>`

- To read that in another file

```
import {anyObject} from 'theFileName.js';  
// Now you can do something with anyObject.
```



# Traditional Way

- Create an IIFE to encapsulate the code

# New Way

Car.js

```
class Car{  
  ...  
};  
export Car;
```

Main.js

```
import {Car} from 'Car.js';  
const c = new Car();
```

# tl;dr

- ES2015 introduces some cool new features including
  - Default parameters
  - String templates
  - Destructuring
  - Modules
  - Spread operator and rest parameters

# Further Study

- Concise, cruftless overview of all ES2015 features
  - <http://github.com/bevacqua/es6>
- Brendan Eich's presentation on ES2015
  - <http://brendaneich.github.io/ModernWeb.tw-2015>