

# JAVASCRIPT ES6

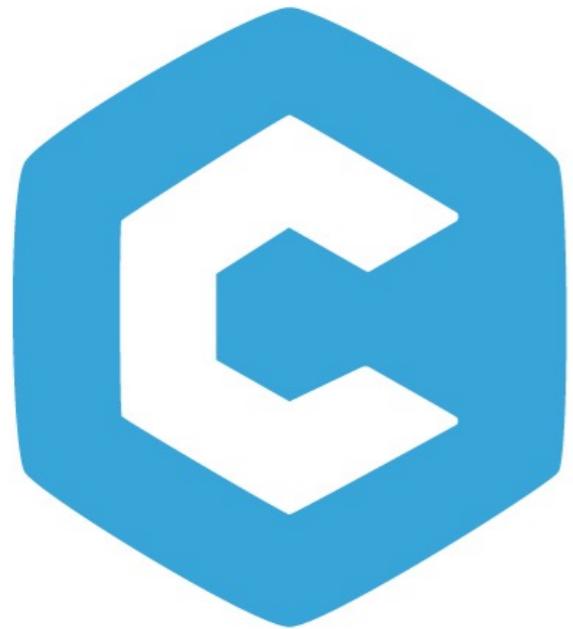
(the best vanilla you've ever tasted)

[github.com/ryanhagerty/dc-es6](https://github.com/ryanhagerty/dc-es6)



# RYAN HAGERTY

Front-End Developer



**CHROMATIC**

[chromatichq.com](http://chromatichq.com)

A close-up photograph of a pepperoni pizza. The pizza is covered in melted cheese that has browned in spots, and numerous red pepperoni slices are scattered across the surface. The crust is visible at the edges.

@HOTPIZZAS

# INTRO

JavaScript Releases

ES7

Why use ES6?

jQuery

Babel

# WHAT YOU'RE HERE FOR

const & let

promises

scope

classes

default & rest

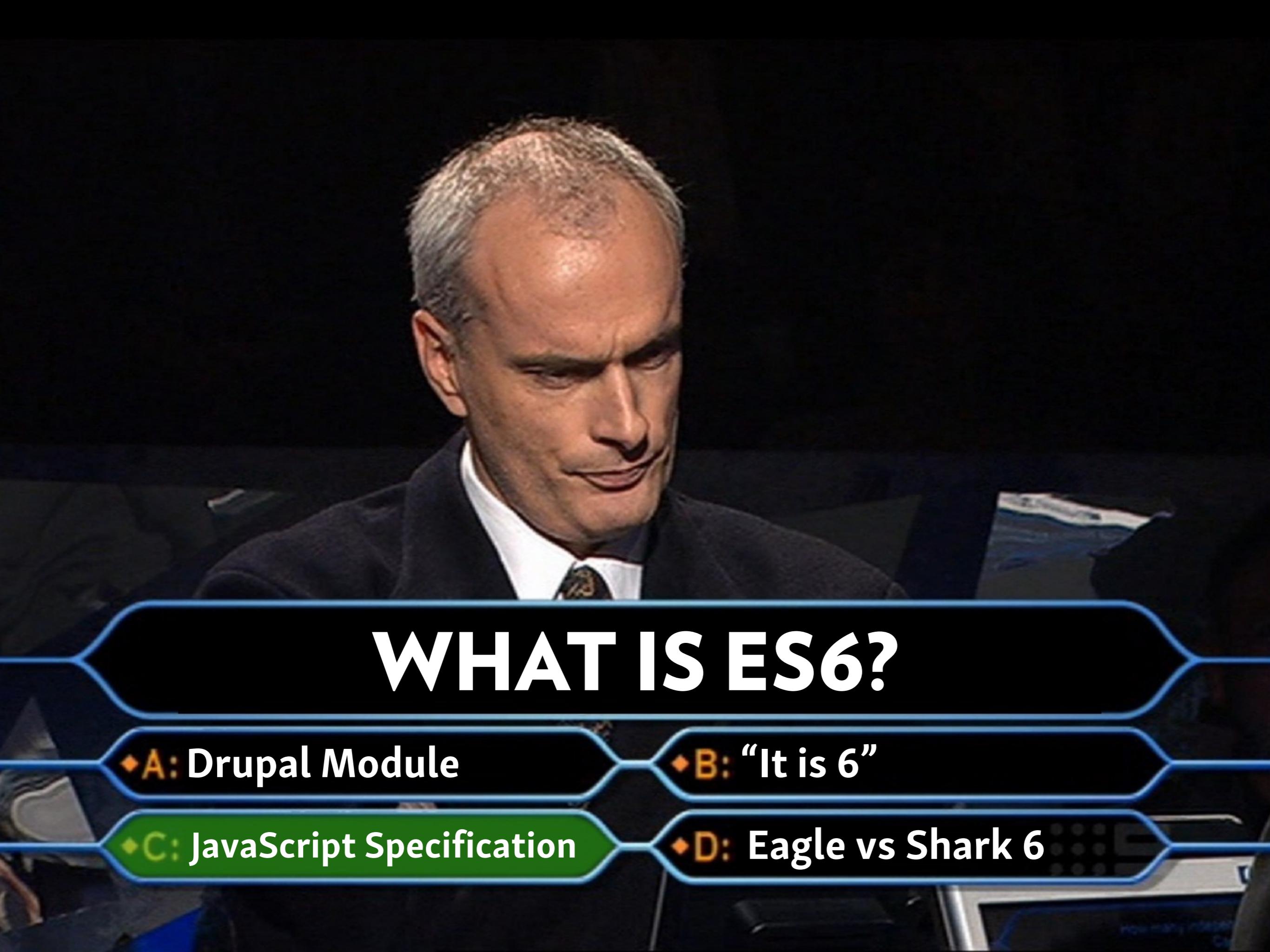
template literals

arrow functions

destructuring

implicit returns

modules

A man with grey hair, wearing a dark suit, white shirt, and patterned tie, is looking down at a game board with a serious expression. The game board has blue and green sections and some numbers. The background is dark.

# WHAT IS ES6?

♦ A: Drupal Module

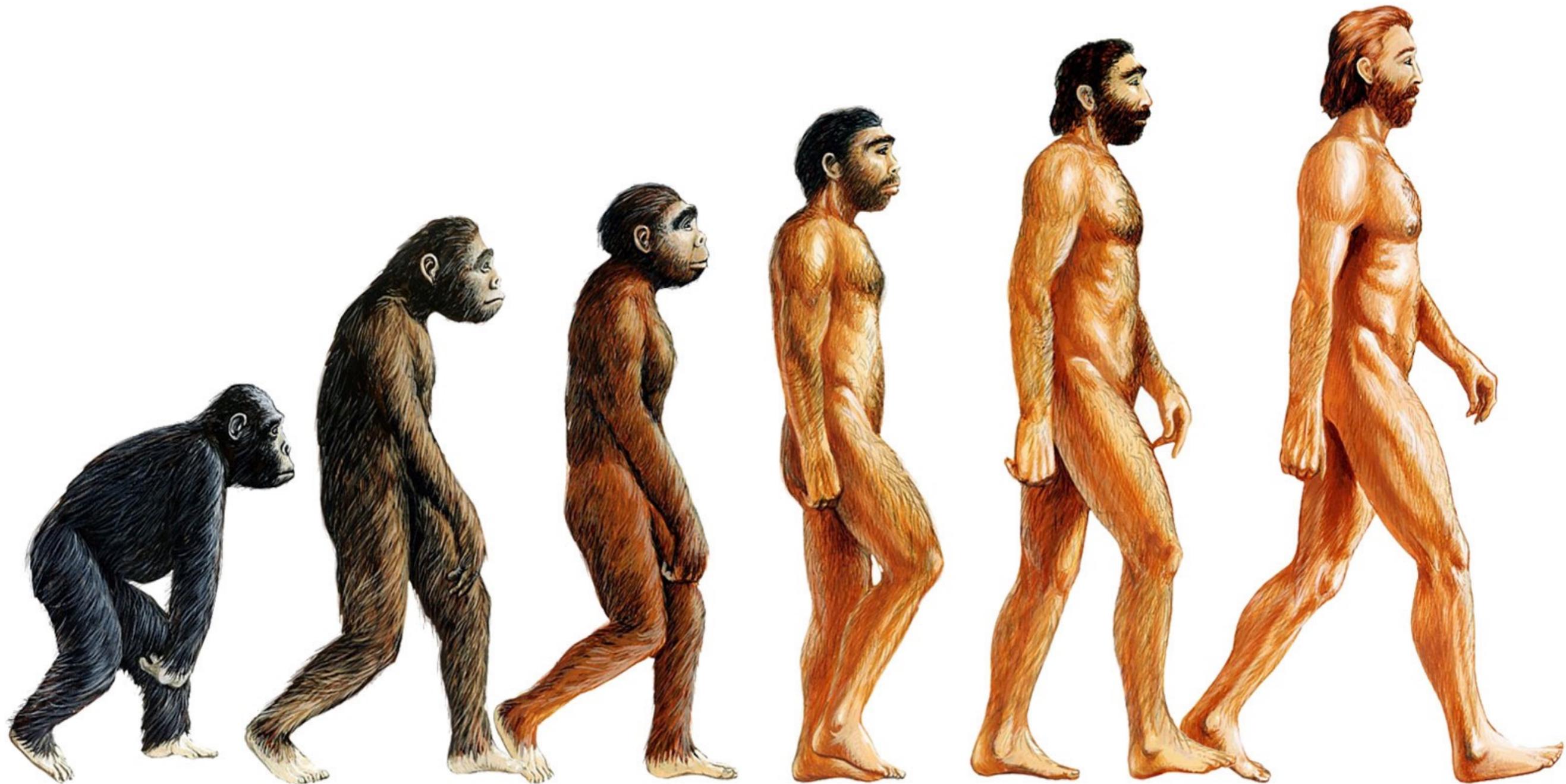
♦ B: “It is 6”

♦ C: JavaScript Specification

♦ D: Eagle vs Shark 6

# ECMASCRIPT 2015





97

ES1

98

ES2

99

ES3

07

ES4

09

ES5

15

ES6

“In a very real sense, the completion  
of the sixth edition is the culmination  
of a fifteen year effort.”

— Allen Wirfs-Brock  
6th Edition Project Editor

A scene from the movie Back to the Future Part II. Marty McFly (Michael J. Fox) is on the left, wearing his iconic brown and red jacket, holding a newspaper. Doc Brown (Christopher Lloyd) is on the right, wearing his signature colorful pajamas and a headband, looking shocked. They are standing in front of a car with a license plate that reads "GO BACK TO THE FUTURE".

THE FUTURE

“The sixth edition provides the foundation for regular, incremental language and library enhancements.”

— Allen Wirfs-Brock  
6th Edition Project Editor

# YEARLY UPDATES!



# ECMASCRIPT2016

(ES7)

# ES7

Array.prototype.includes

```
[1, 2, 3].includes(1); // true
```

```
[1, 2, 3].includes(4); // false
```

# ES7

## Exponentiation Operator

```
3 ** 2; // 9
```

```
Math.pow(3, 2); // 9
```

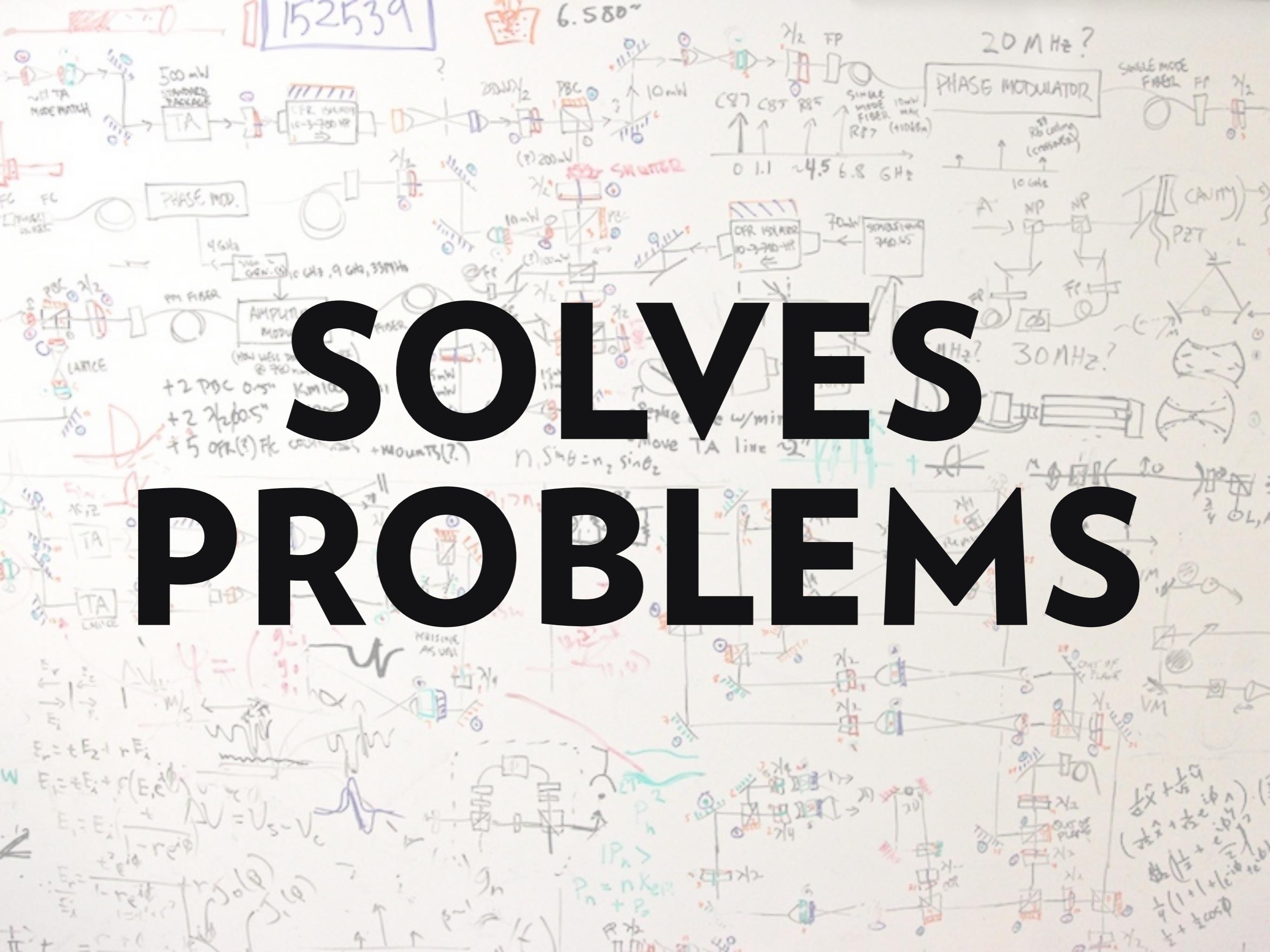
# YOU DID IT!



COOL.

**BUT WHY?**

# SOLVES PROBLEMS



# scope

scope

promises

scope

promises

“this”

scope

promises

“this”

modules

scope

promises

“this”

modules

parameters

scope

promises

“this”

modules

parameters

classes

scope

promises

“this”

modules

parameters

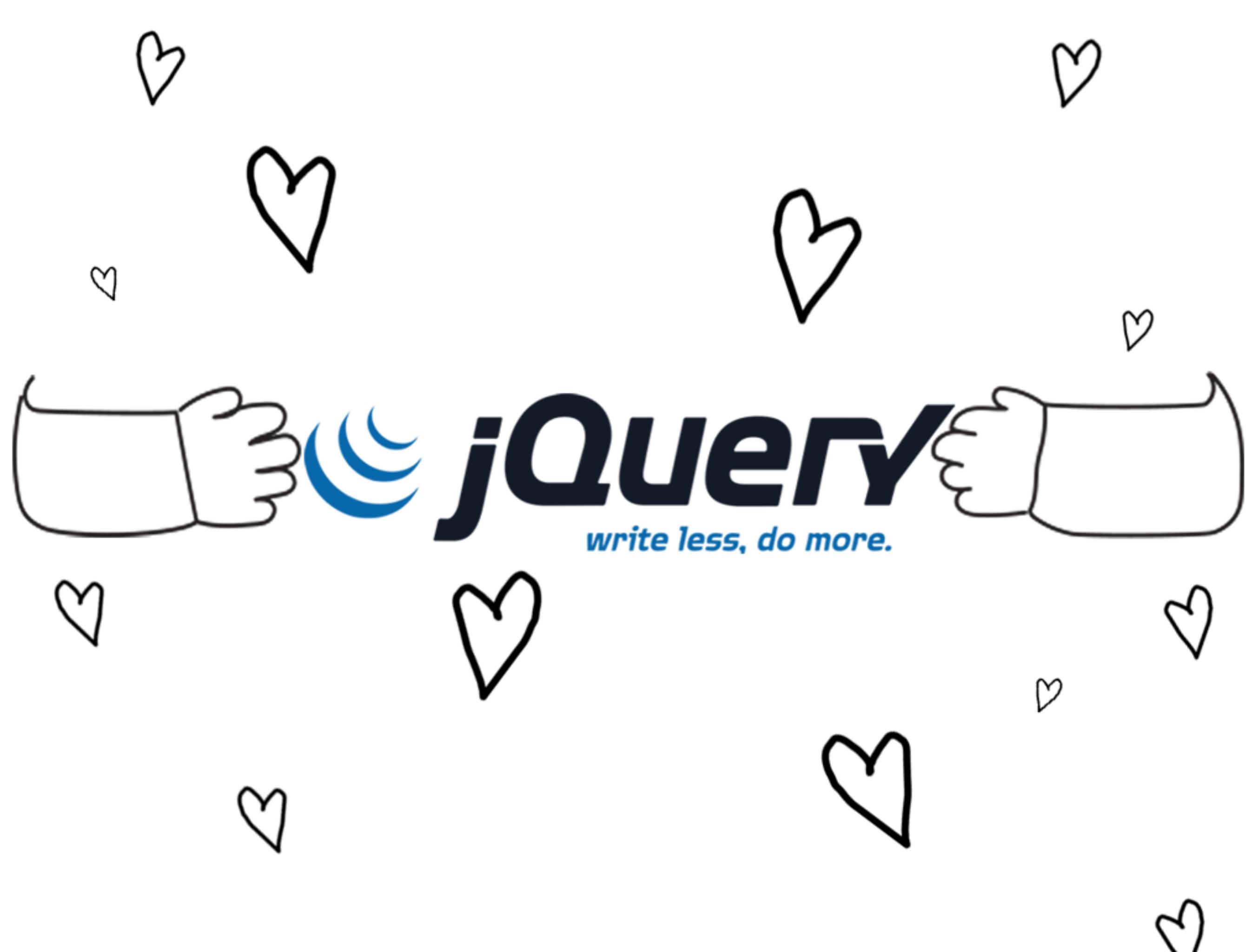
classes

**AND MORE!**



# WEB STANDARDS

**BUT**  
**MAH**  
**JQUERY**



# **ES6 VS JQUERY**

**SOLVES  
DIFFERENT  
PROBLEMS**



# PERFORMANCE



**21 FLAVORS  
IN MY CACHE**

A man in a dark space suit with white stripes on the sleeves is floating in a dark, star-filled space. He has his arms raised above his head, palms facing forward. His hair is blonde and appears to be glowing or reflecting light.

BETTER  
UNDERSTANDING  
OF JS

[adult swim]

# NATIVE SOLUTIONS

addClass

hasClass

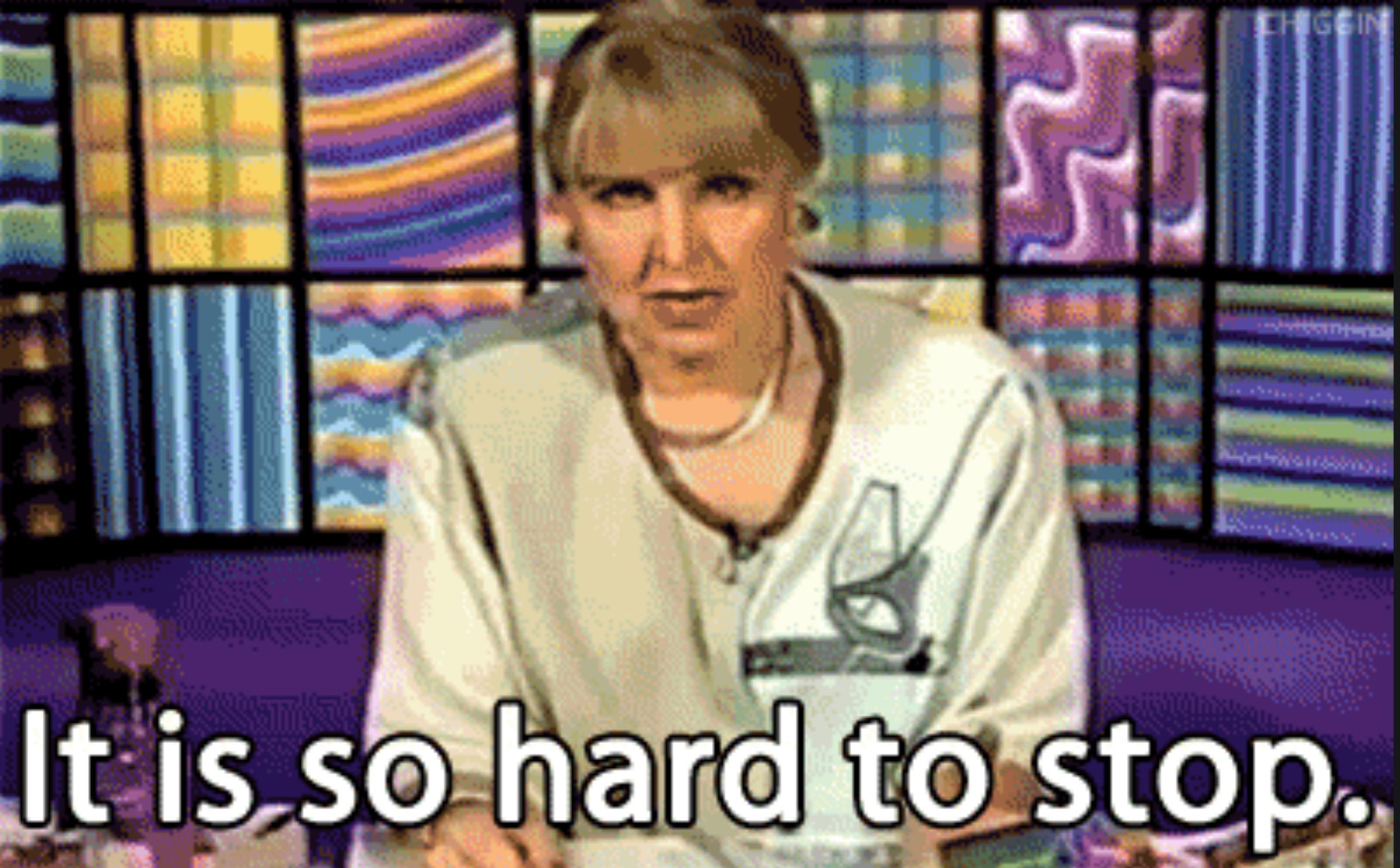
find

attr

toggleClass

\$.map

**PROBABLY  
USE ES6**



**It is so hard to stop.**

A black and white photograph of a city skyline featuring a prominent bridge in the foreground and several tall buildings in the background, all set against a backdrop of scattered clouds.

**LET'S  
GET  
STARTED**

IE 7	IE 8	IE 9	IE 10	IE 11	Edge 12 <sup>[4]</sup>	Edge 13 <sup>[4]</sup>	Edge 14 <sup>[4]</sup>	FF 38 ESR	FF 44	FF 45 ESR	FF 46	FF 47	FF 48	FF 49	FF 50	FF 51	FF 52	Other
0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2
0/7	0/7	0/7	0/7	0/7	0/7	0/7	7/7	3/7	4/7	4/7	4/7	4/7	4/7	4/7	6/7	6/7	0/7	0/7
0/5	0/5	0/5	0/5	0/5	5/5	5/5	5/5	4/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5
0/15	0/15	0/15	0/15	0/15	12/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	0/15
0/6	0/6	0/6	0/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	0/6
0/9	0/9	0/9	0/9	0/9	6/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	0/9
0/4	0/4	0/4	0/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	0/4
0/5	0/5	0/5	0/5	0/5	4/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5
0/5	0/5	0/5	0/5	0/5	2/5	5/5	5/5	2/5	2/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5
0/22	0/22	0/22	0/22	0/22	0/22	0/22	21/22	19/22	19/22	19/22	19/22	21/22	21/22	21/22	21/22	21/22	21/22	0/22
0/24	0/24	0/24	0/24	0/24	0/24	0/24	23/24	20/24	21/24	21/24	21/24	23/24	23/24	23/24	23/24	23/24	23/24	0/24
0/23	0/23	0/23	0/23	0/23	0/23	0/23	22/23	18/23	18/23	18/23	18/23	19/23	19/23	19/23	19/23	19/23	19/23	0/23
0/2	0/2	0/2	0/2	0/2	2/2	2/2	2/2	0/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	0/2
0/2	0/2	0/2	0/2	0/2	0/2	1/2	2/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	0/2

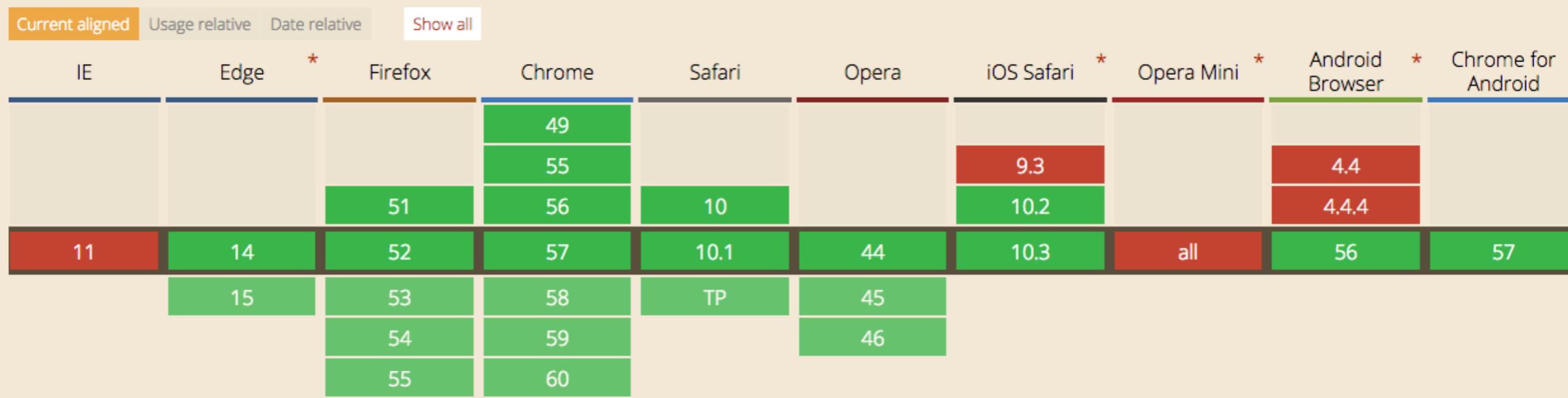
compatibility table

## # Arrow functions - OTHER

Global

70.76%

Function shorthand using => syntax and lexical this binding.



Notes

Known issues (0)

Resources (6)

Feedback

No notes

[caniuse.com](http://caniuse.com)

**BABEL**

**WHAT'S A  
BABE?**

“Babel is a compiler that transforms  
ES6 into legacy browser ES5 code.  
Yes, even IE9.”

— Ryan Hagerty  
The guy speaking right now.

```
// ES6  
[1, 2, 3].map(n => ** 2);
```

```
// Compiled ES5  
[1, 2, 3].map(function(n) {  
    return Math.pow(n, 2);  
});
```

test yourself!

**NPM**

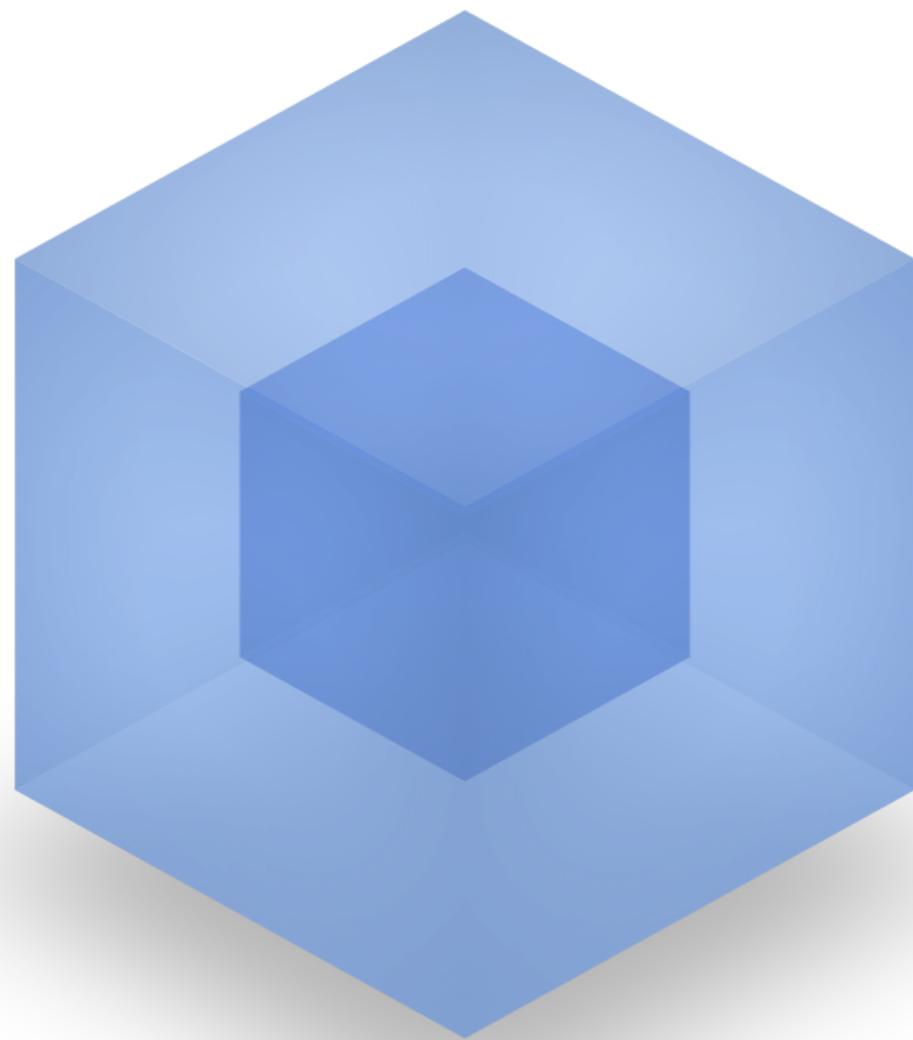
or

**YARN**



Gulp





**webpack**  
MODULE BUNDLER

**BABEL**

**CLI**

# PACKAGE.JSON

```
"scripts": {  
  "babel-cli": "./node_modules/.bin/babel src --out-dir dist"  
},
```

# SRC/

```
const implicitReturn = ((food) => 'I LOVE ' + food)('pizza!')
```

# CLI

```
ryanhagerty@Ryans-MacBook-Pro ➤ /Applications/MAMP/htdocs/dc-es6 ➤ npm run babel-cli  
> dc-es6@1.0.0 babel-cli /Applications/MAMP/htdocs/dc-es6  
> babel src --out-dir dist  
  
src/javascript/babel_cli.js -> dist/javascript/babel_cli.js  
ryanhagerty@Ryans-MacBook-Pro ➤ /Applications/MAMP/htdocs/dc-es6 ➤ █
```

# DIST/

```
var implicitReturn = function (food) {  
  return 'I LOVE ' + food;  
}('pizza!');
```

Gulp

# CLI

```
AMP/htdocs/dc-es6 ➤ npm install --save-dev gulp gulp-babel babel-preset-es2015
```

# GULPFILE.JS

```
gulp.task('js', () => {
  return gulp.src("./src/javascript/*.js")
    .pipe(babel({
      presets: ['es2015']
    }))
    .pipe(gulp.dest("./dist/javascript"));
});
```

```
// Default Gulp task.
```

```
gulp.task('default', ['js']);
```

```
ryanhagerty@Ryans-MacBook-Pro ➤ /Applications/MAMP/htdocs/dc-es6 ➤ gulp
[16:48:35] Using gulpfile /Applications/MAMP/htdocs/dc-es6/gulpfile.js
[16:48:35] Starting 'js'...
[16:48:35] Finished 'js' after 505 ms
[16:48:35] Starting 'default'...
[16:48:35] Finished 'default' after 35 µs
ryanhagerty@Ryans-MacBook-Pro ➤ /Applications/MAMP/htdocs/dc-es6 ➤
```

# DEBUG?





**SOURCE  
MAPS**



babel\_cli.js x babel\_cli.js [sm]

```
3 /**
4  * @file
5  * An example file for babel-cli
6  */
7
8 var implicitReturn = function (food) {
9   return 'I LOVE ' + food;
10}('pizzaaaa!');
11//# sourceMappingURL=data:application/json;charset=utf8;base64,eyJ2ZXJzaW9uIjozLCJz
```

{ } Line 10, Column 16





babel\_cli.js

babel\_cli.js [sm] ×

```
1  /**
2   * @file
3   * An example file for babel-cli
4   */
5
6  const implicitReturn = ((food) => 'I LOVE ' + food)('pizzaaaa!');
```

{ } Line 1, Column 1 (source mapped from babel\_cli.js)

A close-up photograph of a person's hand and forearm. The hand is pointing its index finger directly upwards towards the top of the frame. The skin tone is light. The background is a solid blue color with two white, fluffy clouds positioned behind the hand, one on each side.

**REAL  
ES6  
TIME!**

**(FINALLY)**

# **CONST/LET**

(the new var)

# VAR

```
var dc = "Drupalcon";
dc = "Durpalcon";

console.log(dc); // Durpalcon
```

(can update)

# VAR

```
var dc = "Drupalcon";
var dc = "Durrrrrrrrpalcon";

console.log(dc); // Durrrrrrrrpalcon
```

(can redefine!)



**GROSS**

# LET

```
let number = 41;  
number = 42;  
  
console.log(number); // 42
```

(can update)

# LET

```
let number = 41;  
let number = 42;  
  
console.log(number);  
// Uncaught SyntaxError: Identifier 'number'  
// has already been declared
```

(can't redefine)

# VAR

```
for(var i = 0; i < 5; i++) {  
  setTimeout(function () {  
    console.log(i); // Logs 5 five times  
  }, 100);  
}
```

(loops)

# LET

```
for(let i = 0; i < 5; i++) {  
    setTimeout(function () {  
        console.log(i); // 0 1 2 3 4  
    }, 100);  
}
```

(loops)

# CONST

```
const name = "Ryan";
name = "Rye";
|
console.log(name);
// Uncaught TypeError:
//Assignment to constant variable.
```

(can't update)

# CONST

```
const name = "Ryan";
const name = "Ryerye";

console.log(name);
// Uncaught SyntaxError: Identifier
// 'name' has already been declared
```

(can't redefine)

# CONST

```
const person = {  
    name: "Ryan",  
};  
  
person.name = "Ryerye";  
  
console.log(person.name); // Ryerye
```

(not immutable!)

# **CONST vs. LET**



A WINNER IS CONST

**SCOPE**

# SCOPE

```
const name = "Ryan";  
  
{  
  const name = "Ryerye";  
  console.log(name); // Ryerye  
}  
  
console.log(name); // Ryan
```

**DEFAULT &  
REST  
PARAMETERS**

# DEFAULT

# DEFAULT

```
function logFavoriteFood(food = "pizza") {  
  console.log(food);  
}  
  
logFavoriteFood(); // pizza  
logFavoriteFood("is beer a food?"); // is beer a food?
```

# DEFAULT

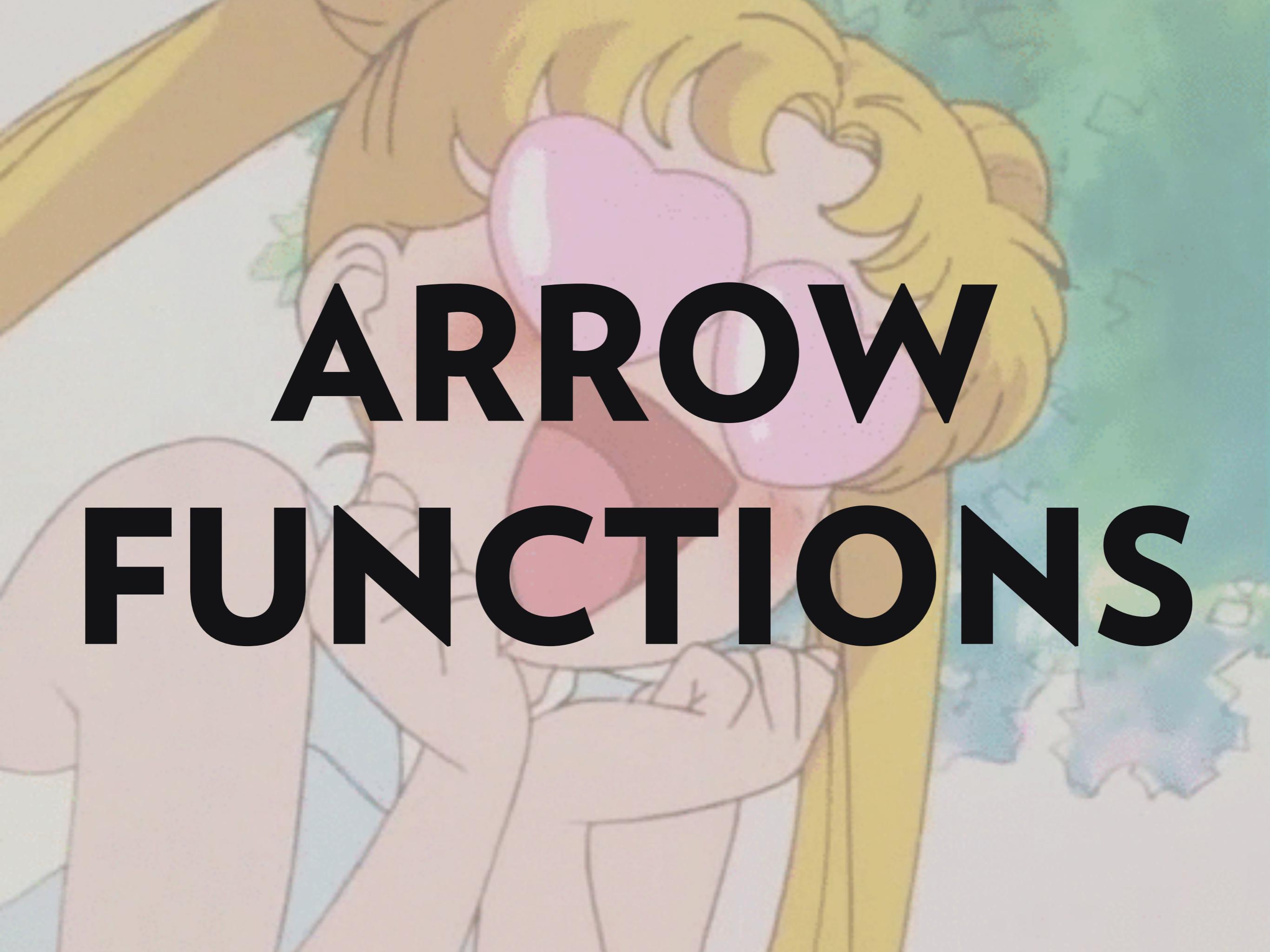
```
function changeBodyColor(color = "dodgerblue") {  
  document.body.style.backgroundColor = color;  
}  
  
changeBodyColor();  
changeBodyColor("#e7e7e7");
```

# **REST**

# REST

```
function logNumbers(...numbers) {  
  console.log(numbers.length);  
}  
  
logNumbers(1); // 1  
logNumbers(1,2,3,4); // 4
```

# ARROW FUNCTIONS



# ARROW FUNCTIONS

```
var logName = function(name) {  
    console.log(name);  
};
```

```
logName("Ryan"); // Ryan
```

(old)

# ARROW FUNCTIONS

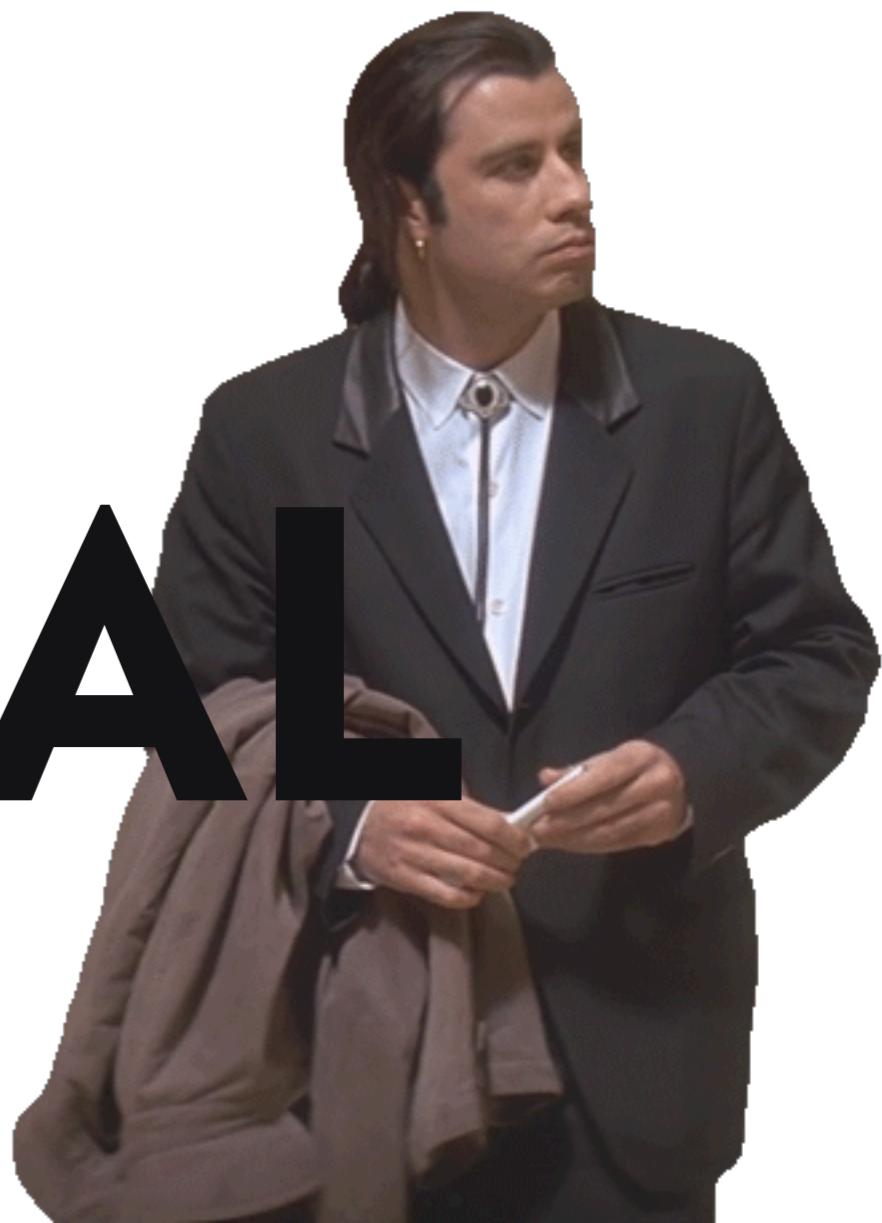
```
const logName = (name) => {  
  console.log(name);  
};
```

```
logName("Ryerye"); // Ryerye
```

(new)

**so...**

**WHAT'S  
THE  
BIG DEAL.**





Lexically binds “this”

THIS!#%\$

# ARROW FUNCTIONS

```
const buttonStuff = {  
  init() {  
    this.listenClick();  
  },  
  
  listenClick() {  
    const button = document.getElementById('button');  
    button.addEventListener('click', function() {  
      this.handleClick();  
    });  
  },  
  
  handleClick() {  
    console.log(`click`);  
  },  
}  
  
buttonStuff.init();
```

# WE COULD...

```
listenClick() {  
  const button = document.getElementById('button');  
  button.addEventListener(`click`, this.logClick);  
},
```

**SUSPEND  
DISBELIEF**

# ARROW FUNCTIONS

```
listenClick() {  
  const button = document.getElementById('button');  
  button.addEventListener(`click`, function() {  
    this.logClick(); // this.logClick is not a function  
  });  
},
```

# ARROW FUNCTIONS

```
listenClick() {  
  const button = document.getElementById('button');  
  const that = this;  
  
  button.addEventListener(`click`, function() {  
    that.logClick();  
  });  
},
```

(shame! )

# ARROW FUNCTIONS

```
listenClick() {  
  const button = document.getElementById('button');  
  button.addEventListener(`click`, function() {  
    this.handleClick();  
  }.bind(this));  
},
```

(better)

# ARROW FUNCTIONS

```
listenClick() {  
  const button = document.getElementById('button');  
  button.addEventListener(`click`, () => {  
    this.handleClick();  
  });  
},
```

(yea!)

# **IMPLICIT RETURNS**

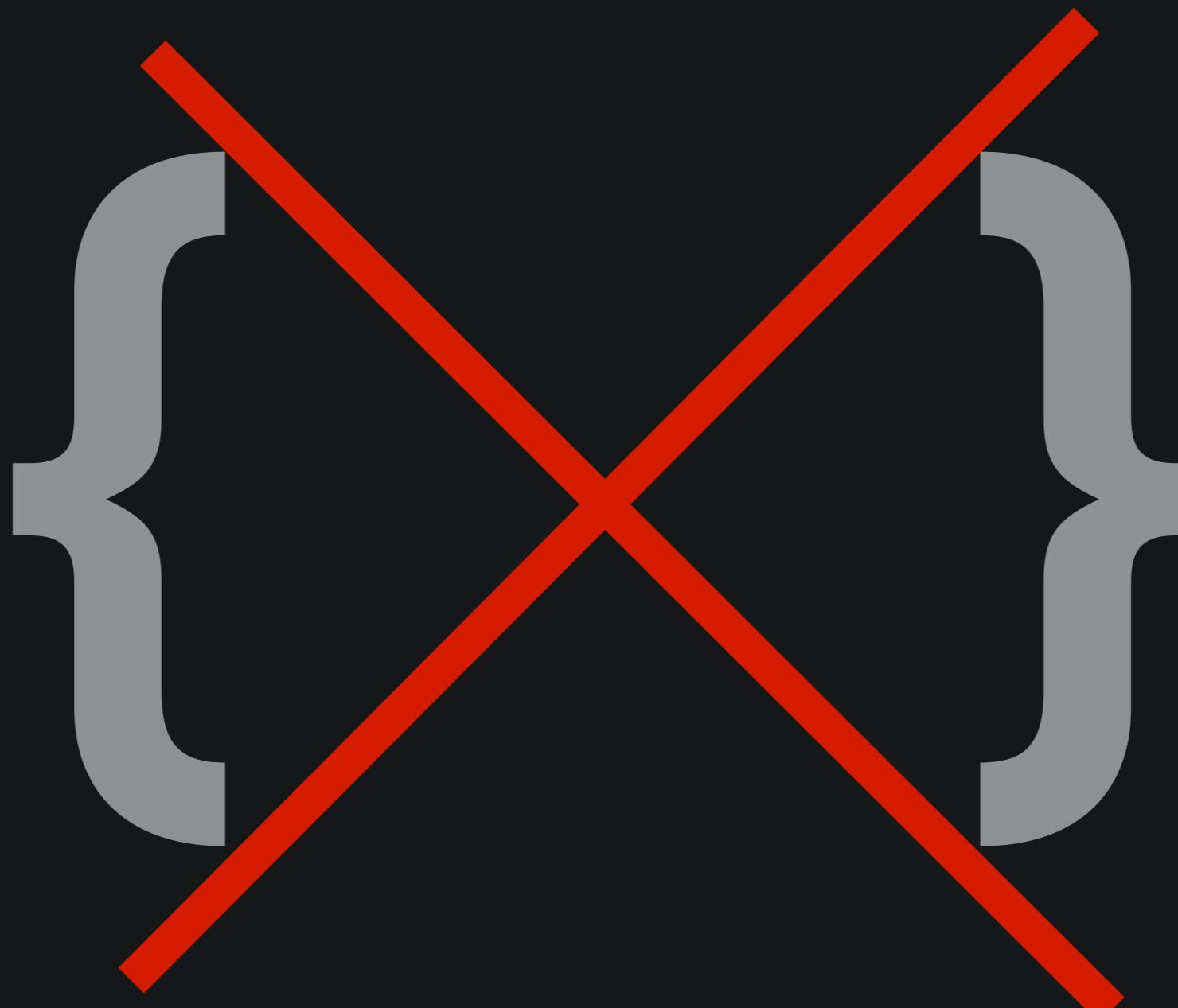
# IMPLICIT RETURNS

```
const number = (x) => {  
  x * x;  
};  
  
console.log(number(2)); // undefined  
  
(nope)
```

# IMPLICIT RETURNS

{ } { }

# IMPLICIT RETURNS



(bye!)

# IMPLICIT RETURNS

```
const number = (x) => x * x;
```

```
console.log(number(2)); // 4
```

(awww yes)

# PROMISES

Put her there, part.

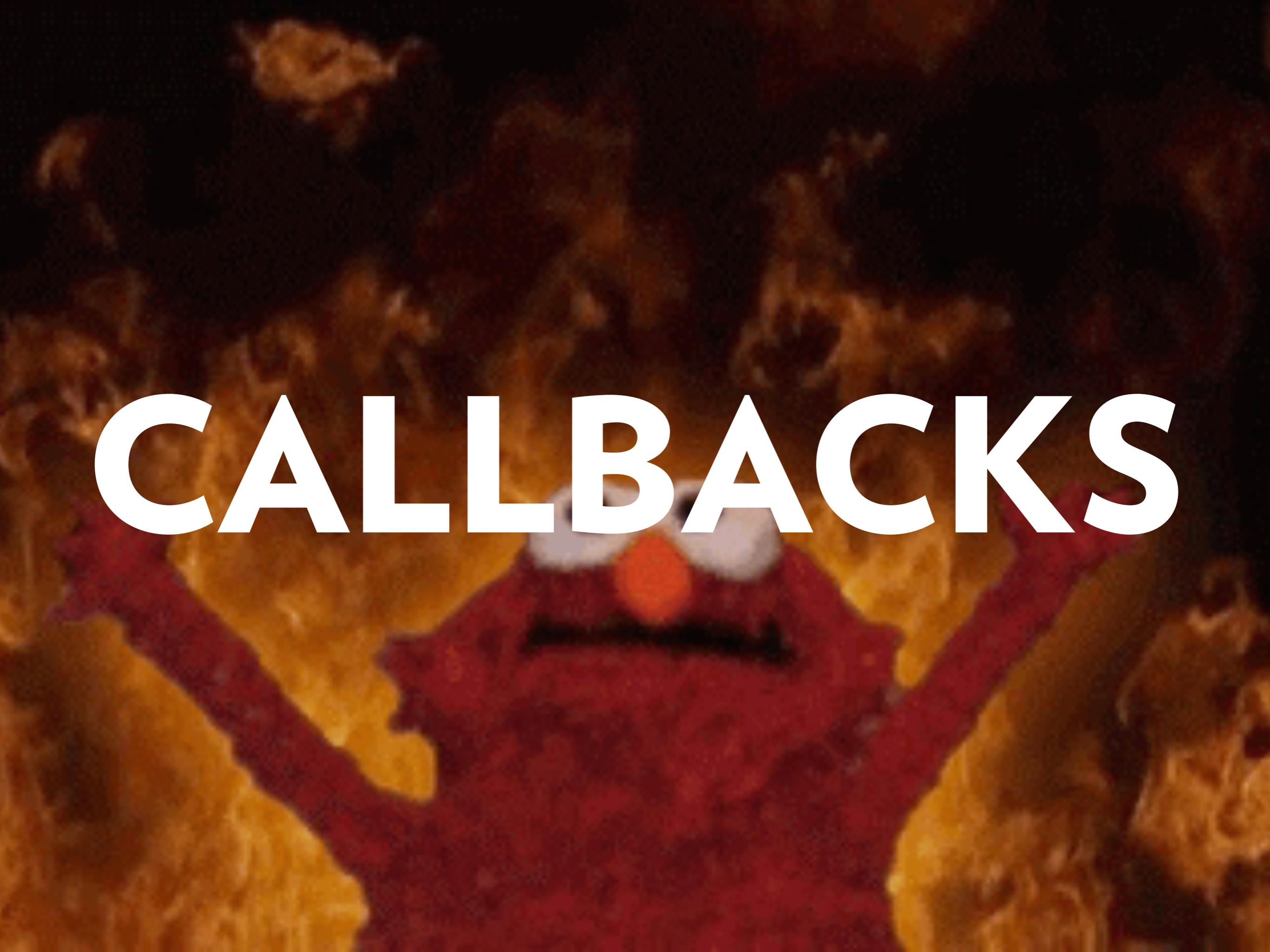
# CALLBACKS

```
getData(function(dataA) {  
    getOtherData(dataA, function(dataB) {  
        // do something with dataA and dataB  
    });  
});
```

# CALLBACKS

```
getData(function(dataA) {  
    getOtherData(dataA, function(dataB) {  
        getOtherData(dataB, function(dataC) {  
            getOtherData(dataC, function(dataD) {  
  
                } );  
            } );  
        } );  
    } );  
});
```

# CALLBACKS



**CALLBACKS**



**KYLE'S  
EXAMPLE**

(source)

# PROMISES

```
const yourPromise = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve("Your data!");
  }, 1000);
});
```

# PROMISES

```
const yourPromise = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve("Your data!");
  }, 1000);
});
```

```
yourPromise.then(message) => {
  console.log("What did I get? " + message);
  // What did I get? Your data!
});
```

# PROMISES

```
const amazonAWS = true;
```

```
const yourPromise = new Promise((resolve, reject) => {
  if (amazonAWS) {
    setTimeout(() => {
      resolve("Your data!");
    }, 1000);
  }
  else {
    reject("It's down.")
  }
});
```

# PROMISES

```
yourPromise.then(function (message) {  
    console.log("What did I get? " + message);  
    // What did I get? Your data!  
}, function (err) {  
    console.log("Oh no! " + err);  
    // Oh no! It's down.  
});
```

# PROMISES

```
const amazonAWS = true;
```

```
const yourPromise = new Promise((resolve, reject) => {
  if (amazonAWS) {
    setTimeout(() => {
      resolve("Your data!");
    }, 1000);
  } else {
    reject("It's down.");
  }
});
```

```
yourPromise.then((message) => {
  console.log("What did I get? " + message);
  // What did I get? Your data!
}, function (err) {
  console.log("Oh no! " + err);
});
```

# PROMISES

```
const amazonAWS = false;
```

```
const yourPromise = new Promise((resolve, reject) => {
  if (amazonAWS) {
    setTimeout(() => {
      resolve("Your data!");
    }, 1000);
  } else {
    reject("It's down.");
  }
});
```

```
yourPromise.then((message) => {
  console.log("What did I get? " + message);
}, function (err) {
  console.log("Oh no! " + err);
  // Oh no! It's down.
});
```

# PROMISES

```
const newPromise = new Promise((resolve, reject) => {
  const result = 2;
  resolve(result);
});  
  
newPromise.then(result) => {
  const newResult = result * 2; // 4
  return newResult;
}).then(result) => {
  const newResult = result * 2; // 8
  console.log(newResult);
};
```

(chained)

# PROMISES

```
const errorFunction = () => {
  console.log('here be an errrrrrror!');
};

newPromise.then((result) => {
  const newResult = result * 2; // 4
  //return newResult;
  return Promise.reject();
}).catch(errorFunction).then((result) => {
  const newResult = result * 2; // 8
  console.log(newResult);

  // here be an errrrrrror!
});
```

(catch)

# PROMISES

```
const apis = ['site1', 'site2', 'site3'];
```

```
Promise.all(apis).then((results) => {
  results.forEach((api) => {
    console.log(api);
    // site1
    // site2
    // site3
  });
});
```

(all)

A photograph of a person's hands raised in front of a solid blue background. The hands are positioned with fingers slightly spread, suggesting a gesture of inquiry or participation. The lighting is even, highlighting the skin tone against the blue.

# CLASSES

# CLASSES

```
class Food {  
  constructor(name) {  
    this.name = name;  
  }  
  
  describe() {  
    console.log(this.name + " is the best food.");  
  }  
}  
  
const pizza = new Food("Pizza");  
pizza.describe(); // Pizza is the best food.
```

(ES6)

# CLASSES

```
var Food = function(name) {  
    this.name = name;  
};  
  
Food.prototype.describe = function() {  
    console.log(this.name + " is the best food.");  
};  
  
var pizza = new Food("Pizza");  
pizza.describe(); // Pizza is the best food.
```

(ES5)

A close-up photograph of a metal spoon resting on a pile of white granulated sugar. The spoon is positioned diagonally, with its handle pointing towards the top left and its bowl pointing towards the bottom right. The sugar is contained within a shallow, transparent dish, which is partially visible at the bottom. The lighting is soft, creating a warm, monochromatic feel.

**SYNTACTIC  
SUGAR**

# CLASSES

```
var Food = function(name) {  
  this.name = name;  
};  
  
Food.prototype.describe = function() {  
  console.log(this.name + " is the best food.");  
};
```

(prototype!)



**CLASSES**

The word "CLASSES" is written in large, bold, black capital letters. It is partially obscured by a large, red circular "prohibited" or "no entry" sign. The sign features a thick red border and a diagonal red line from the top-left corner to the bottom-right corner, crossing over the letters. The letters "C", "L", "A", and "S" are visible on the left side of the slash, while "S", "I", and "E" are visible on the right side.

# CLASSES

```
class Food {  
    constructor(name) {  
        this.name = name;  
    }  
  
    describe() {  
        console.log(this.name + " is the best food.");  
    }  
}  
  
const pizza = new Food("Pizza");  
  
Food.prototype.describe = () => {  
    console.log("Food is altered!")  
};  
  
pizza.describe(); // Food is altered!
```

# CLASSES

```
class Pizza extends Food {  
    toppings(ingredients) {  
        console.log("I like " + ingredients + " on mine.")  
    }  
}  
  
const myStyle = new Pizza();  
myStyle.toppings("mushrooms and onions");  
// I like mushrooms and onions on mine.
```

(extend)

# CLASSES

```
class Pizza extends Food {  
    constructor(name) {  
        super();  
        this.name = "Pizza";  
    }  
  
    toppings(ingredients) {  
        super.describe();  
        console.log("I like " + ingredients + " on mine.");  
    }  
}  
  
const myStyle = new Pizza();  
myStyle.toppings("mushrooms and onions");  
// Pizza is the best food.  
// I like mushrooms and onions on mine.
```

(super)

# TEMPLATE LITERALS



# TEMPLATE LITERALS

```
var beverage = "BEER";
console.log("Time to go drink " + beverage + "!");
// Time to go drink BEER!
```

(ES5)

# TEMPLATE LITERALS

```
const beverage = `BEER`;
console.log(`Time to go drink ${beverage}!` );
// Time to go drink BEER!
```

(ES6)

# TEMPLATE LITERALS

```
const beverage = "BEER";
console.log("Time to go drink ${beverage}!!");
```

```
Time to go drink ${beverage}!!
```

(use backticks!)

# TEMPLATE LITERALS

```
const beverage = `BEER`;
console.log(`Time to go drink ${beverage}!` );
// Time to go drink BEER!
```

(with backticks)

# TEMPLATE LITERALS

```
const isPizza = true;
const cheer = () => `

YEAAAAAA!
`;

const template = () => `

<div>
  ${isPizza ? `${cheer()}` : `I'm so sad. :(`}
</div>
`;
// YEAAAAAA!

const nextLink = document.querySelector(`.next`);
nextLink.insertAdjacentHTML(`afterend`, template());
```

(ternary)

# **DESTRUCTURING**



# DESTRUCTURING

```
var name1 = 'Ryan';
var name2 = 'Rye';
var name3 = 'Ryerye';
```

(ES5)

# DESTRUCTURING

```
const [ name1, name2, name3 ] = [ `Ryan`, `Rye`, `Ryerye` ];  
console.log(name2);  
// Rye
```

(ES6)

# DESTRUCTURING

```
const names = [ `Ryan`, `Rye`, `Ryerye` ];  
const [name1, name2, name3] = names;  
console.log(name2);  
// Rye
```

(ES6)

# DESTRUCTURING

```
const superHero = {  
  alias: `Spider-man`,  
  name: `Peter Parker`  
};
```

```
const { alias, name } = superHero;  
console.log(alias);  
// Spider-man
```

(objects)

# DESTRUCTURING

```
const superVillain = {  
  alias: `Sandman`,  
  name: `William Baker`,  
  weaknesses: {  
    weakness: `Water`,  
    weakness2: `Planning heists`  
  }  
}
```

```
const { alias, weaknesses: { weakness } } = superVillain;  
console.log(alias, weakness);  
// Sandman Water
```

(nested)

A photograph of a person's hands holding a colorful, modular electronic device against a yellow background. The device has a red base layer with blue and white text that appears to read "LUMI". Above this is a purple layer with a yellow star-shaped button and some smaller controls. The top layer is pink. The hands are wearing light-colored, possibly white, gloves.

**MODULES**

A close-up, slightly blurred portrait of a man with short, light brown hair, looking downwards with a serious expression. He is wearing a light-colored shirt.

**PROBLEMS**

# MODULES

```
<script src="https://ajax.googleapis.com/ajax/libs/  
jquery/1.6.2/jquery.min.js"></script>  
<script src="jquery.flexslider.js"></script>
```

(old)

# MODULES

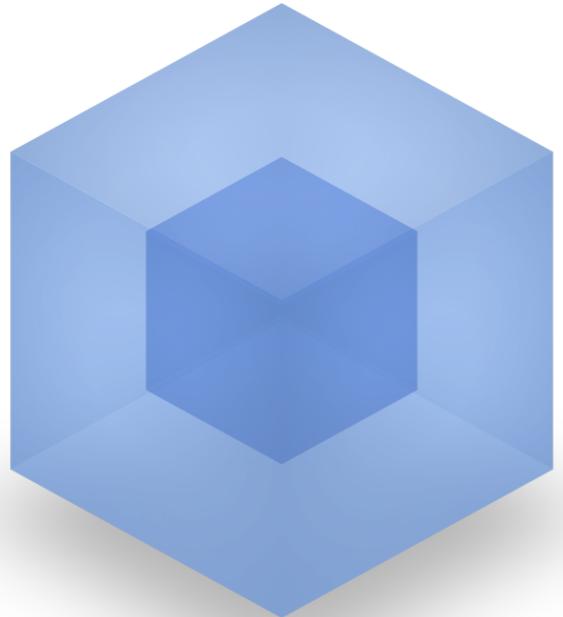
```
$window.load(function() {  
    $('.flexslider').flexslider({  
        animation: "slide"  
    });  
});
```

(old)

A cartoon illustration of Homer Simpson from the TV show "The Simpsons". He is shown from the waist up, wearing his signature yellow shirt and blue pants. He has a large, round face with white eyes and a wide, toothy grin. He is holding a large, brown paper cup with a straw in his right hand. The background consists of a green grassy field with a brown wooden fence and some small, wispy clouds. The word "NOPE" is overlaid in large, bold, black capital letters.

**NOPE**





**webpack**  
MODULE BUNDLER

 jspm.io



A photograph showing a large, dense pile of fallen apples on a grassy field. In the foreground, a red mechanical harvester or conveyor belt is visible, partially buried in the apples. The background is filled with the branches and leaves of apple trees. The overall scene depicts a typical apple harvesting operation.

# TREE SHAKING

# **IMPORT/EXPORT**

# MODULES

```
export const ColorChange = {  
  changeColor(color, el) {  
    el.style.backgroundColor = color;  
  }  
};
```

(modules/colorChange.js)

# MODULES

```
import { ColorChange } from './modules/colorChange.js';  
  
ColorChange.changeColor(`dodgerblue`, document.body);
```

(10\_modules.js)

# MODULES

```
import { ColorChange, AnotherModule } from './modules/moduleList.js';
```

(multiple modules)

# MODULES

hi.js

```
export default function () { console.log(`hi!`) }
```

```
import hi from `./modules/hi.js`  
hi();
```

(functions)

# MODULES

Can also import:

classes

objects

primitives

A man in a dark tuxedo jacket and white shirt is holding a bouquet of red and yellow flowers. He is looking slightly to his left with a serious expression. The background is a plain, light-colored wall.

ES6!

**WHAT'S  
NEXT**

**ECMASCRIPT2016**  
**(ES7)**

**ECMASCRIPT2017**  
**(ES8)**

# WHAT DID WE LEARN?

const & let

promises

scope

classes

default & rest

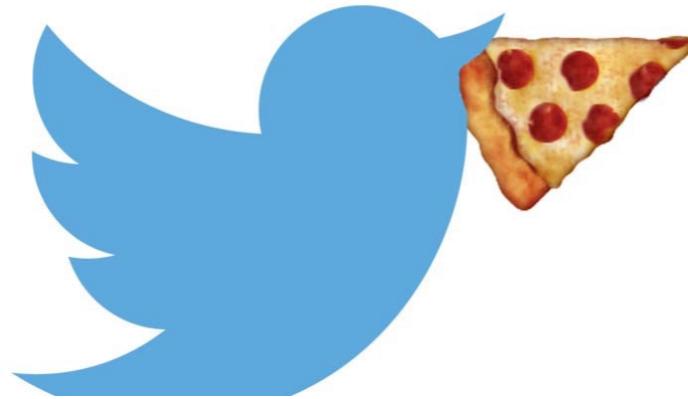
template literals

arrow functions

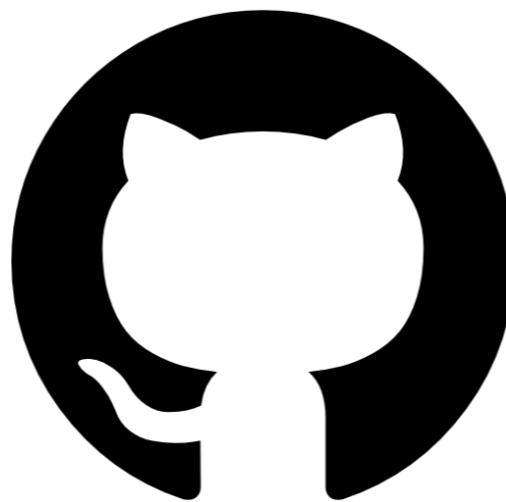
destructuring

implicit returns

modules



[twitter.com/hotpizzas](https://twitter.com/hotpizzas)



[github.com/ryanhagerty/dc-es6](https://github.com/ryanhagerty/dc-es6)



**THANK  
YOU!**