

# *JAVASCRIPT*

---

# 2099

---

The Future and Present of JS

# ***CHRISTOPHER PLUMMER***



Developer at Upstatement

***JAVASCRIPT***

***ECMAScript***

ECMAScript == JavaScript

ECMAScript === JavaScript

***“ECMAScript WAS ALWAYS  
AN UNWANTED TRADE  
NAME THAT SOUNDS LIKE  
A SKIN DISEASE.”***

Brendan Eich

*Internet.*

**Serious Business**



**ES**

**ES3**

**ESU**



***HARMONY***

**TC-39**

A man in a dark suit, striped shirt, and patterned tie is smiling and holding a computer monitor. The background is a warm, reddish-brown color.

*Internet.2*

*still*

➤ Serious Business.

**ES** **S**



# ***HIGHER ORDER ARRAY FUNCTIONS***

```
['flux', 'flux', 'flux', 'capacitor'].filter( function(item) {  
    return item === 'capacitor';  
}); // ['capacitor']
```

```
['1.0', '1.21'].forEach( function(item) {  
    console.log(item + ' gigawatts!!!1 ');  
});  
// 1.0 gigawatts!!!1 1.21 gigawatts!!!1
```

# OBJECT.CREATE()

```
var alien = {  
  name: 'Bob',  
  announce: function() {  
    return 'I am ' + this.name;  
  }  
}
```

```
var marklar = Object.create(alien, {  
  name: {  
    value: 'marklar',  
    configurable: false  
  }  
});
```

```
marklar.name = 'bob';  
marklar.announce(); // 'marklar'
```

# OTHER ITEMS

- ★ `getters & setters`
- ★ `Object.keys()`
- ★ `'use strict';`
- ★ `JSON.parse,`  
`JSON.stringify`
- ★ `Date.now()`
- ★ `legal trailing commas`

*The passionate, functional, micro-serviced approach*



*Expert*

# Resumé Driven Development

O RLY?

@ThePracticalDev

**ES6**

~~ES6~~

ES2015

- ★ arrows
- ★ classes
- ★ enhanced object literals
- ★ template strings
- ★ destructuring
- ★ default + rest + spread
- ★ let + const
- ★ iterators + for..of
- ★ generators
- ★ unicode
- ★ modules

- ★ module loaders
- ★ map + set + weakmap + weakset
- ★ proxies
- ★ symbols
- ★ subclassable built-ins
- ★ promises
- ★ math + number + string + array + object APIs
- ★ binary and octal literals
- ★ reflect api
- ★ tail calls

**ES2015**



**ES2016**

**ES2017**

**ES2099**

*This time you have definitely chosen the right libraries and build tools*



*Real World*

## Rewriting Your Front End Every Six Weeks

O RLY?

@ThePracticalDev

***WHY BOTHER  
LEARNING THIS  
STUFF?***

O'Reilly Press

# JavaScript for Millennials

I heard react was good



O'REILLY®

MACKLEMORE 著  
訳

***BECAUSE I  
SAID SO,  
THAT'S WHY.***

*WHAT'S IN*

**ES 2015**



*THINGS THAT  
CHANGE HOW YOU*

**WRITE JS**

# ***TEMPLATE STRINGS***

```
var captain = {  
  name: 'Kirk'  
};  
var location = 'mountain';  
  
'Captain ' + captain.name + ' is climbing the ' + location + '.';  
  
// "Captain Kirk is climbing the mountain."
```

```
`Captain ${captain.name} is climbing the ${location}.`;`  
  
// "Captain Kirk is climbing the mountain."
```

# ARROW SYNTAX

```
[1, 2, 3, 4, 5].filter(function(num) {  
  return num > 2;  
}); // [3, 4, 5]
```

```
[1, 2, 3, 4, 5].map(function(num, index) {  
  return num + index;  
}); // [1, 3, 5, 7, 9]
```

```
[1, 2, 3, 4, 5].filter(num => num > 2);  
// [3, 4, 5]
```

```
[1, 2, 3, 4, 5].map((num, index) => num + index);  
// [1, 3, 5, 7, 9]
```

# ARROW SYNTAX

```
var Runner = {  
  name: 'Logan',  
  run: function() {  
    var self = this;  
  
    [1, 2, 3].forEach(function(i) {  
      console.log(`${self.name} is running ${i}!`);  
    });  
  }  
};  
  
Runner.run();
```

```
var Runner = {  
  name: 'Logan',  
  run: function() {  
    [1, 2, 3].forEach( i => {  
      console.log(`${this.name} is running ${i}!`);  
    });  
  }  
};
```

# DEFAULT ARGUMENTS

```
function farnsworth(audience) {  
  audience = audience || 'everyone';  
  console.log(`Good news ${audience}!`);  
};  
  
// Good news everyone!
```

```
function farnsworth(audience = 'everyone') {  
  console.log(`Good news ${audience}!`);  
};  
  
// Good news everyone!
```

# ***SPREAD OPERATORS***

```
function getFirstStarship() {  
  var starships = arguments;  
  return starships[0];  
};
```

```
getFirstStarship('galactica', 'enterprise', 'serenity');  
// 'galactica'
```

```
function getFirstStarship(...starships) {  
  return starships[0];  
};
```

```
getFirstStarship('galactica', 'enterprise', 'serenity');  
// 'galactica'
```

# ***SPREAD OPERATORS II***

```
function exampleFunction(arg1, arg2) {  
  return console.log(arg1, arg2);  
};  
  
function callExample() {  
  return exampleFunction.apply(null, arguments);  
};
```

```
function callExample() {  
  return exampleFunction(...arguments);  
};  
  
callExample('benjamin', 'sisko');  
// 'benjamin sisko'
```

# ***DESTRUCTURING***

```
var Runner = {  
  name: 'logan',  
  job: 'run'  
};
```

```
var name = Runner.name;  
var job = Runner.job;  
// 'logan'
```

```
var { name, job } = Runner;
```



# ***OBJECT LITERAL SYNTAX***

```
var makeRobot = function(name, service) {  
  return {  
    name: name,  
    service: service  
  };  
}
```

```
var makeRobot = function(name, service) {  
  return { name, service };  
}
```

# OBJECT LITERAL SYNTAX II

```
var deathStar = {  
  location: 'alderaan',  
  demonstrateFirepower: function(planet) {  
    this.destroy(planet);  
  }  
}
```

```
var deathStar = {  
  location: 'alderan',  
  demonstrateFirepower(planet) {  
    this.destroy(planet);  
  }  
}
```

# CLASS SYNTAX

```
function Jedi(name) {  
  this.name = name;  
};
```

```
Jedi.prototype.attack = function(target) {  
  console.log(`${this.name} punches ${target.name}`)  
}
```

```
var obiWan = new Jedi('obi-wan');  
var sith = new Jedi('vader');
```

```
sith.attack = function(target) {  
  console.log(`${this.name} shoots lightning at ${target.name}`)  
};
```

```
sith.attack(obiWan); // vader shoots lightning at obi-wan  
obiWan.attack(sith); // obi-wan punches vader
```

# CLASS SYNTAX

```
class Jedi {  
    constructor(name) {  
        this.name = name;  
    }  
  
    attack(target) {  
        console.log(`${this.name} punches ${target.name}`);  
    }  
};  
  
class Sith extends Jedi {  
    attack(target) {  
        console.log(`${this.name} shoots lightning ${target.name}`);  
    }  
}  
  
var obiWan = new Jedi('obi-wan');  
var sith = new Sith('vader');
```

***THINGS THAT  
CHANGE HOW YOU***

**USE JS**

# ***PROMISES***

```
getAJAX('spacejam.com', function(data) {  
  if (data.err) {  
    console.log(err);  
  } else {  
    return res;  
  }  
});
```

```
getAJAX('spacejam.com')  
  .then( data => data )  
  .catch( err => {  
    console.log(err);  
  } );
```

# OBJECT.ASSIGN()

```
var brundle = {  
  species: 'human',  
  name: 'brundle'  
};  
  
var fly = {  
  species: 'fly'  
};  
  
var brundleFly = Object.assign(brundle, fly);  
  
// { species: 'fly', 'name': 'brundle' }
```

# MODULES

```
var Enterprise = (function(WarpCore, $, undefined) {  
  var engineering = WarpCore.init();  
  return {  
    engineering: engineering  
  };  
  
})(window.WarpCore, window.jQuery)
```

```
import WarpCore from 'lib/warp-core';  
import * as $ from 'node_modules/jquery';  
  
let Enterprise = {  
  engineering: WarpCore.init()  
}  
  
exports default Enterprise;
```



# LET, CONST

```
var PLANET = { type: 'apes' };

if (PLANET.type !== '') {
  var DrZaius = new Ape();
  DrZaius.judge();
}

DrZaius.judge();
```

```
const PLANET = { type: 'apes' };

if (PLANET.type !== '') {
  let DrZaius = new Ape();
  DrZaius.judge();
}

PLANET = null;
// Uncaught TypeError: Assignment to constant variable.

DrZaius.judge();
// Uncaught ReferenceError: DrZaius is not defined
```

# ***SET***

```
var replicants = ['roy', 'pris', 'leon', 'roy'];  
replicants.push('pris');  
  
console.log(replicants);  
// ['roy', 'pris', 'leon', 'roy', 'pris']
```

```
var replicants = new Set(['roy', 'pris', 'leon', 'roy']);  
replicants.add('pris');  
  
console.log(replicants); // ['roy', 'pris', 'leon']  
replicants.toString()   // object Set
```

# ARRAY.FROM()

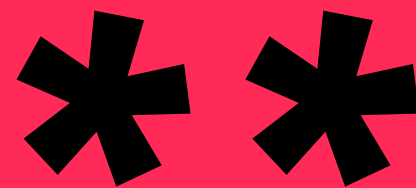
```
var arr = [];  
var len = 3;  
  
for (var i = 0; i < len; i++) {  
    arr[i] = i;  
}  
  
console.log(arr) // [0,1,2]
```

```
Array.from({length: 3}, (value, key) => key); // [0,1,2]
```

# *OTHER ITEMS*

- ★ Symbols
- ★ `for..of`
- ★ Unicode
- ★ subclassable built-ins
- ★ hyperbolic geometry
- ★ binary, octal literals
- ★ proxies
- ★ tail calls
- ★ Reflect API
- ★ Maps, WeakMaps, WeakSets

**ES2016**



# ***ARRAY.INCLUDES()***

```
var replicants = ['roy', 'pris', 'leon'];  
  
var isLeonReplicant = replicants.indexOf('leon') > -1;  
  
console.log(isLeonReplicant); // true
```

```
var replicants = ['roy', 'pris', 'leon'];  
  
var isDeckerReplicant = replicants.includes('decker');  
  
console.log(isDeckerReplicant); // false
```

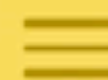
***WHEN CAN  
I USE THIS  
MAGIC SAUCE?***



***BUT I WANT  
IT NOW!***

BABEL  
6765

# BABEL



☒ Evaluate   Presets:

1 c

1 "use strict";

# BABEL



☒ Evaluate

Presets:

1

1

```
"use strict";
```

***TOMORROW'S  
JAVASCRIPT  
PROBLEMS,  
TODAY!***

***WP + BABEL =***



K

```
.. (up a dir)
```

```
▶ dist/
```

src/

functions.php

style.css

~~~~~





**TOOLS**

```
→ javascript-2099-demo git:(start-over) X
```

```
→ javascript-2099-demo git:(start-over) X
```

```
→ javascript-2099-demo git:(start-over) ✗ browserify \
src/js/app.js -o dist/js/app.js \
--transform [ babelify ]

/Users/plumcakes/Sites/wordcamp/javascript-2099-demo/src/js/app.js:1
import HelloWorld from './module';
^
ParseError: 'import' and 'export' may appear only with 'sourceType: module'
→ javascript-2099-demo git:(start-over) ✗
```

→ javascript-2099-demo git:(start-over) X





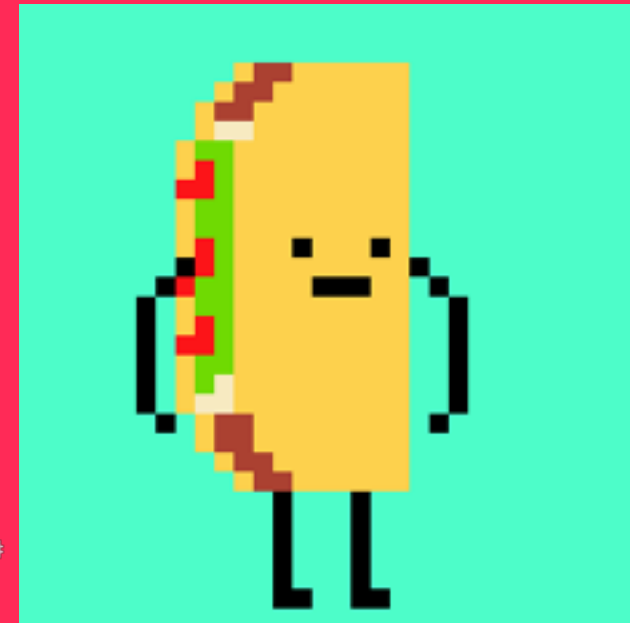
→ javascript-2099-demo git:(start-over) X

Session: camp 1 1 1:..ipt-2099-demo\* 2:..ipt-2099-demo- 19 Jul 20:13



→ javascript-2099-demo git:(start-over) X

Session: camp 1 1 1:..ipt-2099-demo\* 2:..ipt-2099-demo- 19 Jul 20:33



# ***EXAMPLE REPO***

[https://github.com/signal-intrusion/  
javascript-2099-demo](https://github.com/signal-intrusion/javascript-2099-demo)

*Does it run? Just leave it alone.*



# Writing Code that Nobody Else Can Read

*The Definitive Guide*

O RLY?





@ThePracticalDev

**THE  
FUTURE**

# ***PROPOSAL PROCESS***

<https://github.com/tc39/proposals>



|    | Proposal                                                              | Champion                                                   | Stage |
|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------|------------------------------------------------------------|-------|
|                                                                                     | <a href="#">SIMD.JS - SIMD APIs + polyfill</a>                        | John McCutchan, Peter Jensen, Dan Gohman, Daniel Ehrenberg | 3     |
|                                                                                     | <a href="#">Async Functions</a>                                       | Brian Terlson                                              | 3     |
|                                                                                     | <a href="#">Trailing commas in function parameter lists and calls</a> | Jeff Morrison                                              | 3     |
|                                                                                     | <code>Function.prototype.toString</code> revision                     | Michael Ficarra                                            | 3     |
|                                                                                     | <a href="#">Asynchronous Iterators</a>                                | Kevin Smith                                                | 2     |
|                                                                                     | <a href="#">function.sent metaproperty</a>                            | Allen Wirfs-Brock                                          | 2     |
|                                                                                     | <a href="#">Rest/Spread Properties</a>                                | Sebastian Markbage                                         | 2     |
|                                                                                     | <a href="#">Shared memory and atomics</a>                             | Lars T Hansen                                              | 2     |
|                                                                                     | <a href="#">System.global</a>                                         | Jordan Harband                                             | 2     |
|                                                                                     | <a href="#">Lifting Template Literal Restriction</a>                  | Tim Disney                                                 | 2     |
|                                                                                     | <a href="#">ArrayBuffer.transfer</a>                                  | Luke Wagneer & Allen Wirfs-Brock                           | 1     |
|  | <code>export * as ns from "mod";</code> <a href="#">statements</a>    | Lee Byron                                                  | 1     |
|  | <code>export v from "mod";</code> <a href="#">statements</a>          | Lee Byron                                                  | 1     |
|  | <a href="#">Class and Property Decorators</a>                         | Yehuda Katz and Jonathan Turner                            | 1     |

# ***DECORATORS***

[https://github.com/wycats/  
javascript-decorators](https://github.com/wycats/javascript-decorators)



# DECORATORS

```
function readonly(target, key, descriptor) {  
    descriptor.writable = false;  
    return descriptor;  
}  
  
class Robot {  
    @readonly  
    name() { return `${this.first} ${this.last}`; }  
}  
  
let klatu = new Robot();  
klatu.name = 'Barada Nicto';  
  
// Cannot assign read only property 'name'
```

# ***SYSTEM.GLOBAL***

`https://github.com/tc39/  
proposal-global`

# ***SYSTEM.GLOBAL***

```
window.$.toString();
```

```
// 'function (selector,context){return new  
jQuery.fn.init(selector,context)}'
```

```
System.global.$.toString();
```

```
// 'function (selector,context){return new  
jQuery.fn.init(selector,context)}'
```

# ***SIMD.JS***

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/SIMD](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/SIMD)

# ***ASYNC FUNCTIONS***

[https://github.com/tc39/  
ecmascript-asyncawait](https://github.com/tc39/ecmascript-asyncawait)

# ASYNC FUNCTIONS

```
getAJAX('spacejam.com')  
  .then( data => {  
    return sendAJAX('derpy-api.biz', data);  
  }).then( res => {  
    console.log(res);  
  });
```

```
async function() {  
  let data = await getAjax('spacejam.com');  
  let res = await sendAjax('derpy-api.biz');  
  console.log(res)  
}
```

*BABEL*

~~Object.observe~~



**THE  
FUTURE**

***AGAIN***

***THE HUMAN  
ADVENTURE IS  
JUST BEGINNING...***

# ***CHRISTOPHER PLUMMER***



`chris.plummer@upstatement.com`

`@IntrusionSignal`

`github.com/signal-intrusion`

`@signalintrusion [Ember Community Slack]`

demo: [https://github.com/signal-intrusion/  
javascript-2099-demo](https://github.com/signal-intrusion/javascript-2099-demo)