



CS196

JavaScript

What is JavaScript?



Only runs in the browser..

Terrible syntax

Just a toy...

Script

scripting or **script language** is a programming language that supports scripts, i.e. programs written for a special run-time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator. **Scripting languages** are often interpreted (rather than compiled).

JavaScript Is...

- high-level
- dynamic
- weakly typed
- object based
- multi-paradigm (imperative, event-driven,
functional, object-oriented)
- interpreted

Why Learn JavaScript?

You Have No Choice



JavaScript, Java, Python,
PHP, Ruby, C++, C, Objective
C, R, Go, Perl, Swift, Scala,
Groovy, Rust, D



JavaScript



You Have No Choice



You Have No Choice

Then...



...now



Office 365

JavaScript Runs Everywhere

JavaScript
Java

Script

JavaScript, Java, Python,
PHP, Ruby, C++, C,
Objective C, R, Go, Perl,
Swift, Scala, Groovy, Rust, D

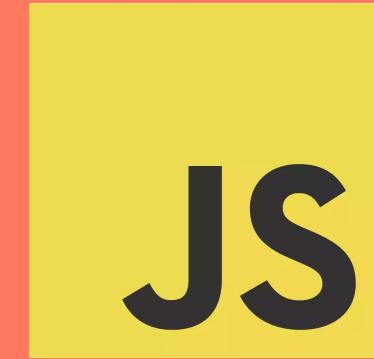


Full Stack



\$ node yourcode.js

- Server side
- Since 2009



<script>yourcode</script>

- Browser side
- Since forever

JavaScript Runs Everywhere

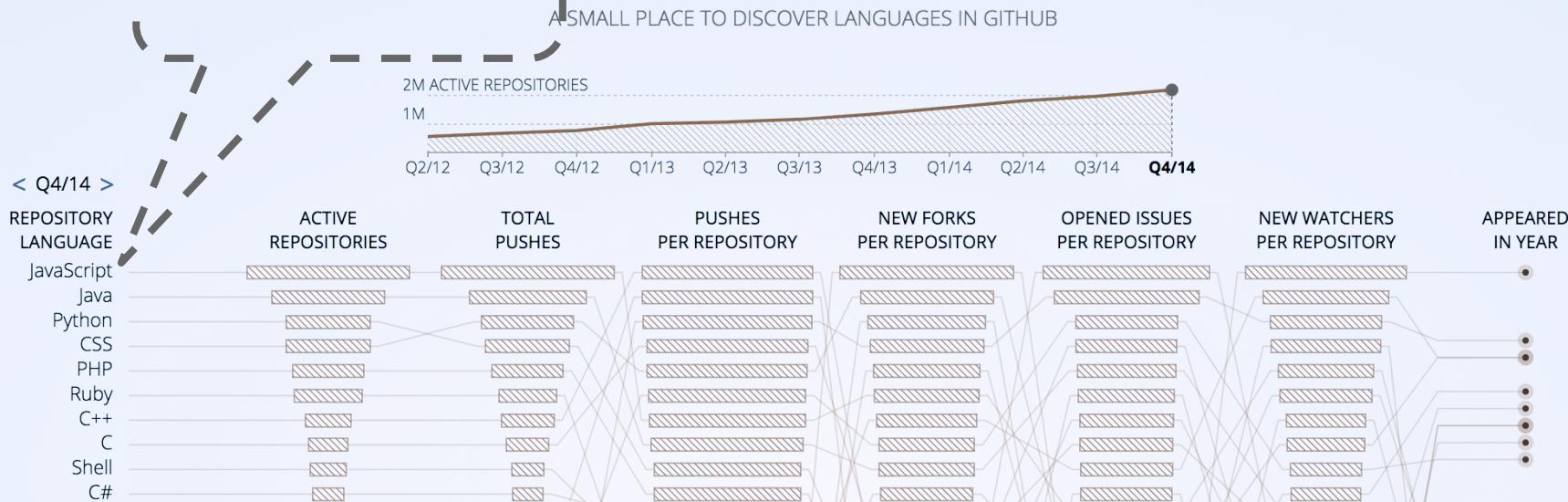
And JavaScript can be used to run *almost anything*:

- Web applications
- Web APIs
- Actual scripts
- Command-line utilities
- GUI applications (meaning web applications)

JavaScript Has a Great Community

We're #1!

GitHut



<http://githut.info/>

JavaScript Has a Great Community

NETFLIX

LinkedIn



P PayPal

Medium

React

eBay®

AngularJS
by Google

JavaScript Has a Fantastic Package Manager

CS196 

```
$ cd myproject  
$ npm install --save multimatch
```

```
$ git clone <whatever>  
$ cd <whatever>  
$ npm i
```

JavaScript Has a Fantastic Package Manager

CS196 

Stats

4 downloads in the last day

12 downloads in the last week

31 downloads in the last month

No open issues on GitHub

No open pull requests on GitHub

Stats

541,491 downloads in the last day

3,266,244 downloads in the last week

11,827,825 downloads in the last month

222 open issues on GitHub

74 open pull requests on GitHub

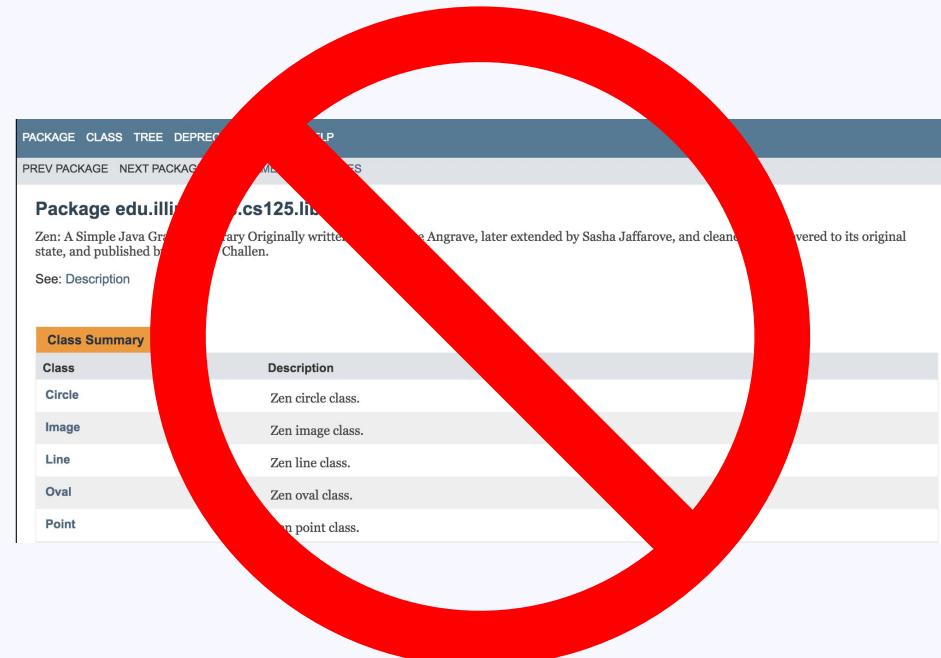
These are **not** the
droids I'm looking for...

<https://npmjs.org>

JavaScript Has a Fantastic Package Manager

[NPM](#) makes it *extremely* easy to publish libraries. As a result, JavaScript libraries tend to be:

- Small
- Well-documented
- Well-maintained
- Easy to evaluate



JavaScript is Flexible and Evolving

```
1 var array = [1, 2, 3, 4, 5, 6];
2 var sum = 0;
3 for (var i = 0; i < array.length; i++) {
4     if (array[i] % 2 == 0) {
5         sum += array[i];
6     }
7 }
8 console.log(sum);
```

<https://runkit.com/gchallen/javascript-idiom-comparison/0.0.4>

JavaScript is Flexible and Evolving

```
1 const _ = require('underscore' 1.8.3 );
2 let array = [1, 2, 3, 4, 5, 6];
3 let sum = _.chain(array)
4   .filter((value) => {
5     return value % 2 == 0;
6   })
7   .reduce((sum, value) => {
8     return sum += value;
9   }, 0)
10  .value();
11 console.log(sum);
```

<https://runkit.com/gchallen/javascript-idiom-comparison/0.0.4>

JavaScript is Flexible and Evolving (ES6)

ECMAScript 6 — New Features: Overview & Comparison

[Tweet](#)[Star](#) 4,080[Fork me on GitHub](#)

Constants

[Constants](#)

Scoping

[Block-Scoped Variables](#)[Block-Scoped Functions](#)

Arrow Functions

[Expression Bodies](#)[Statement Bodies](#)[Lexical this](#)

Extended Parameter Handling

[Default Parameter Values](#)[Rest Parameter](#)[Spread Operator](#)

Template Literals

[String Interpolation](#)[Custom Interpolation](#)[Raw String Access](#)

Extended Literals

[Binary & Octal Literal](#)[Unicode String & RegExp Literal](#)

Enhanced Regular Expression

[Regular Expression Sticky Matching](#)

Arrow Functions

Expression Bodies

More expressive closure syntax.

ECMAScript 6 — syntactic sugar: [reduced](#) | [traditional](#)

```
odds  = evens.map(v => v + 1)
pairs = evens.map(v => { even: v, odd: v + 1 })
nums  = evens.map((v, i) => v + i)
```



ECMAScript 5 — syntactic sugar: [reduced](#) | [traditional](#)

```
odds  = evens.map(function (v) { return v + 1; });
pairs = evens.map(function (v) { return { even: v, odd: v + 1 }; });
nums  = evens.map(function (v, i) { return v + i; });
```



<http://es6-features.org/>

JavaScript is Flexible and Evolving (ES7)

Async Functions

Of all the features in ES2016+, async functions are by far the biggest game changer. If you only learn one thing from this article, it should be that you need to start using async functions as soon as possible.

Why, you ask?

If you've ever written code like this, you know the pain that is asynchronous workflow in JavaScript:

```
1 function createEmployeeWorkflow(cb){  
2  
3     createEmployee(function(err, employee){  
4         If (err) { return cb(err); }  
5  
6         if (employee.needsManager()) {  
7  
8             selectManager(employee, function(err, manager){  
9                 If (err) { return cb(err); }  
10            }  
11        }  
12    }  
13}  
14
```

JavaScript will
continue challenging
you as you grow as a
hacker

Programming Is Fun?

Programming Is Fun!

(If it's not, you're using the wrong tools.)





JavaScript Is Most Fun

- Easy things are *really easy*
- Hard things are **possible**
- You can write *beautiful* code
- The libraries and community support are **fantastic**

JS

var
arr



3];

ELPFUL

Roger Hargreaves



int[]
array



3};



A Brief (And Interesting) History of JavaScript (And Java)



Released

**Development
Time**

**Development
Team**

Result

JS

Java's "Write Once, Run Anywhere"

1. Install an otherwise useless piece of software called the Java interpreter
2. Download a compiled class file and hope that it runs on your computer, and that you can even launch it

JavaScript's "Write Once, Run Anywhere"

1. Install a web browser
2. Visit a website



Moral of the
Story:

The Internet
Wins

Concurrency in JavaScript

What is Concurrency?

In computer science, concurrency is the decomposability property of a program, algorithm, or problem into order-independent or partially-ordered components or units. This means that even if the concurrent units of the program, algorithm, or problem are executed out-of-order or in partial order, the final outcome will remain the same. This allows for parallel execution of the concurrent units, which can significantly improve overall speed of the execution in multi-processor and multi-core systems.

OK, But What Is Concurrency *Really*?

If some part of your code has to stop,
another part can keep going

Why Would Your Code Stop?

There are a variety of reasons that your code might have to stop:

- 1.
- 2.
- 3.
- 4.

Concurrency Is Not Parallelism

Concurrent code *can* harness, but
doesn't *require* parallelism

Concurrency In JavaScript

JavaScript emerged from the browser...

1. One thread per tab that does *everything*: rendering HTML, applying CSS rules, listening for user input, painting... and **running JavaScript**.
2. As a result all JavaScript blocking operations are done asynchronously. **There is no way (until recently) for a function to wait for a blocking request to complete!**

Concurrency Example

```
1 const request = require('request' 2.83.0 );
2 let response = request('https://www.google.com');
3 console.log(response);
```

Concurrency Example

```
1 const request = require('request' 2.83.0 );
2 let response = request('https://www.google.com', function (error, response, body) {
3   console.log(body.length);
4 });
```

<https://runkit.com/gchallen/asynchronous-example/0.0.1>

Let's Write Some JavaScript!

Clone and edit this [RunKit](#) example:

<https://goo.gl/VjJ78L>