JavaScriptures:

The JavaScript Toolchain

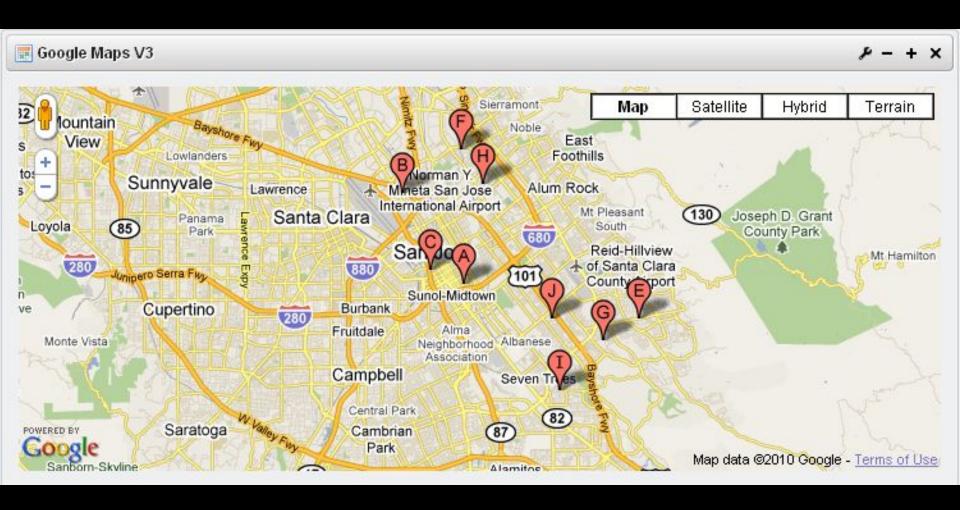# But first, what is JavaScript?

Ja·va·Script

/ˈjävəskript,ˈjavə-/ 🔊

noun *trademark*

an object-oriented computer programming language commonly used to create interactive effects within web browsers.
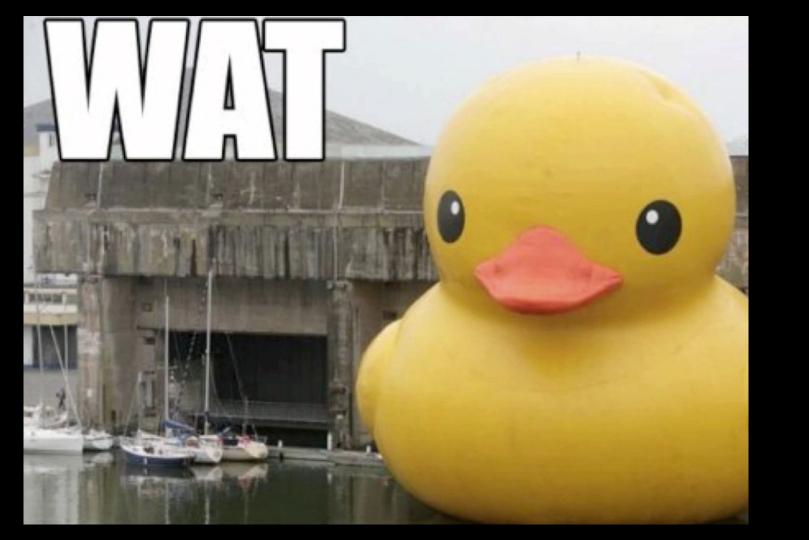
Then Google Maps happened

JavaScript quickly went from being a "toy language" to something that was understood to be capable of real power

But there were problems...

Wat

@garybernhardt

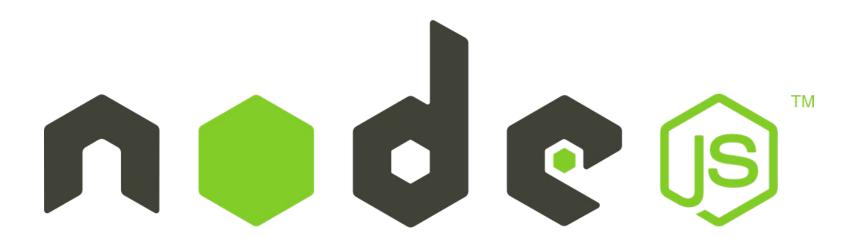Joey Clover [Follow]

Technical Co–founder @ TabbDrink

Jan 23 · 5 min read

I used to hate JavaScript. Now I like it.

NodeJS opened up the world of JavaScript to a new community

(But the language had problems)

About 378,000 results (0.34 seconds)

Why do people hate Node.js? - Quora
https://www.quora.com/Why-do-people-hate-Node-js ▾
I think it all stems from a relatively large number of developers being highly critical of JavaScript itself as a language, and as it happens many of those people ...

The Problem with Node.js - Developer Angst
devangst.com/the-problem-with-nodejs/ ▾
May 23, 2015 - An honest appraisal of Node.js and it's frustrations from a Developer who's ... Firstly, please don't take this post as an "**I hate Node.js**" post (as ...

Why do people hate Node.js? : webdev - Reddit
https://www.reddit.com/r/webdev/comments/1drh80/why_do_people_hate_nodejs/ ▾
May 6, 2013 - Do some people **hate Node.js** so much that they would downvote anyone who even asks a neutral question about it or am I missing something ...

"I Hate JavaScript" - François Zaninotto's blog
www.redotheweb.com/2015/12/04/i-hate-havascript.html ▾
Dec 4, 2015 - "**I hate Node.js**: it's based on a language that was written in 10 days by a guy called Brendan". Wake up dude, JavaScript has evolved a lot ...

Why I think Node.JS is a terrible platform | Hacker News
https://news.ycombinator.com/item?id=4495101 ▾
I agree that the **Node.js** model of concurrency is one of the weakest models out there ...... Except that if you don't really know javascript, and **hate** cps, coffeescript ...

However, server-side JavaScript started to be used in all sorts of creative ways

It started to rewrite itself

SEMICOLON TERMINATOR 3

RISE OF THE TRANSPILERS

CoffeeScript was the spark. It showed the world that if you didn't like the language you could change it, and you didn't need the browser vendors to do so

Babel is a JavaScript compiler.

Use next generation JavaScript, today.

Put in next-gen JavaScript	Get browser-compatible JavaScript out

```
const x = [1, 2, 3];
foo([...x]);
```

```
var x = [1, 2, 3];
foo([].concat(x));
```

Babel has support for the latest version of JavaScript through syntax transformers. These plugins allow you to use new syntax without waiting for browser support

Before browser vendors implemented support for modern JavaScript this was essential

Babel unlocked JavaScript's potential as a language that can be enjoyable to use

It also created a *platform* for real-time, open-source language development and exploration

📁 babel-plugin-proposal-async-gen...	v7.0.0-beta.44	10 days ago
📁 babel-plugin-proposal-class-prop...	Fix super nested class bugs (#76	an hour ago
📁 babel-plugin-proposal-decorators	Migrate `babel-cli` and `babel-c	a day ago
📁 babel-plugin-proposal-do-express...	v7.0.0-beta.44	10 days ago
📁 babel-plugin-proposal-export-def...	v7.0.0-beta.44	10 days ago
📁 babel-plugin-proposal-export-na...	v7.0.0-beta.44	10 days ago
📁 babel-plugin-proposal-function-bi...	Add "use strict" directive (#7411	2 days ago
📁 babel-plugin-proposal-function-se...	v7.0.0-beta.44	10 days ago
📁 babel-plugin-proposal-logical-assi...	Migrate `babel-cli` and `babel-c	a day ago
📁 babel-plugin-proposal-nullish-coal...	v7.0.0-beta.44	10 days ago
📁 babel-plugin-proposal-numeric-se...	v7.0.0-beta.44	10 days ago
📁 babel-plugin-proposal-object-rest...	v7.0.0-beta.44	10 days ago

Language proposals are submitted by TC39, and prototyped with Babel

These proposals are then moved in stages through the approval process

🔗 Active proposals

Proposals follow this process document. This list contains only stage 1 proposals and higher that have not yet been withdrawn/rejected, or become finished.

Stage 3

🚀	Proposal	Author	Champion	Tests
🚀	`Function.prototype.toString` revision	Michael Ficarra	Michael Ficarra	✅
	`global`	Jordan Harband	Jordan Harband	✅
	`import()`	Domenic Denicola	Domenic Denicola	❓
	Legacy RegExp features in JavaScript	Claude Pache	Mark Miller Claude Pache	✅
	`BigInt`	Daniel Ehrenberg	Daniel Ehrenberg	🚧
	Optional `catch` binding	Michael Ficarra	Michael Ficarra	✅
	`import.meta`	Domenic Denicola	Domenic Denicola	❓

Once a proposal has reached Stage IV it is considered stable and will be incorporated into the language

How does this apply to Artsy?

```json
{
  "presets": [
    ["@babel/env", {
      "targets": {
        "browsers": ["last 2 versions"]
      }
    }],
    "@babel/react",
    "@babel/preset-typescript",
    "@babel/stage-3"
  ],
```

Our JavaScript projects incorporate many language features, some of which are not a part of the official JavaScript spec

```
export const LoginModal = props => {
  return (
    <Modal>
      <Container>
        <Content>
          <Label>Log in</Label>
          <Input type="text" placeholder="Username" />
          <Input type="password" placeholder="Password" />
        </Content>
        <ActionButtons>
          <ActionButton color="white" hover="black">
            Log in
          </ActionButton>
        </ActionButtons>
      </Container>
    </Modal>
  )
}
```

TypeScript

"JavaScript that Scales" (it's true)

```tsx
import Button from "../Buttons/Default"
import { ProgressIndicator } from "../ProgressIndicator"

interface State {
  count: number
  finished: boolean
}

class Steps extends React.Component<any, State> {
  public numberOfSteps: number = 4
  state = {
    count: 0,
    finished: false,
  }
}
```

These language features don't run natively in the browser

They need to be *transpiled* into something that the browser (or the server) can understand

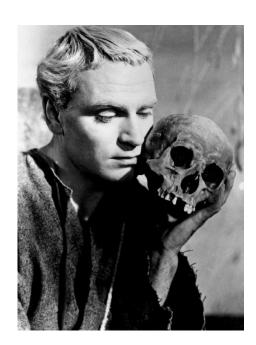# JavaScript is Assembly Language for the Web: Part 2 - Madness or just Insanity?

With JavaScript's low and high-level language features, anything is possible

There are more things in heaven and earth, Horatio,
Than are dreamt of in your philosophy.
- *Hamlet* (1.5.167-8), Hamlet to Horatio

4. You are expected to expect the unexpected every minute, every hour of every day and of every night.

\- John C. Lilly

And things move fast....

Which JavaScript library should I use?

2010   **Jquery**

2016 **React+Redux+Immutable or** angular.js, vue.js, knockout.js and don´t forget to use Gulp or bower and webpack? Webpack + Gulp? Or just use Meteor.js then you don´t need anything else? Wait.... Ok, Maybe Webpack+React+Redux+Immutable.js+Gulp and Jquery 2.0?

Things that were previously impossible to imagine have become possible, as well as common

WEBASSEMBLY

WebAssembly (abbreviated *Wasm*) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable target for compilation of high-level languages like C/C++/Rust, enabling deployment on the web for client and server applications.
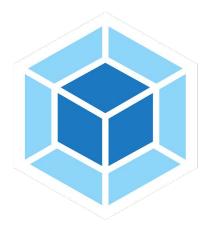
JavaScript frameworks can make use of WebAssembly to confer massive performance advantages and new features while still making functionality easily available to web developers.
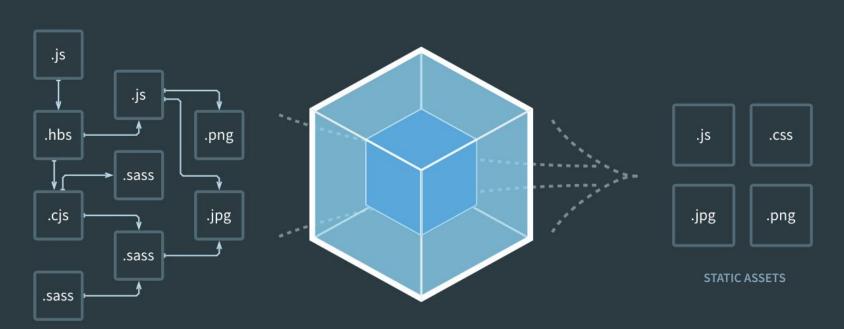
But how does one manage this outward-spiraling complexity?

bundle your scripts

.js

.js

.hbs

.png

.sass

.cjs

.jpg

.sass

.sass

MODULES WITH DEPENDENCIES

.js

.css

.jpg

.png

STATIC ASSETS

Webpack is a *Module Bundler* that runs during development

It builds a dependency graph of your application and combines those assets into one or more bundles to be consumed by a target
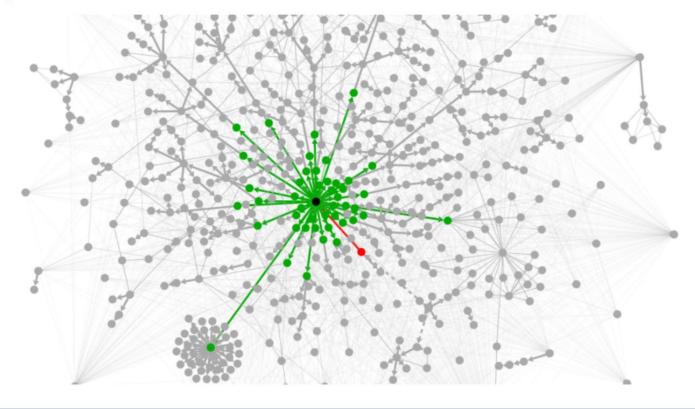
Kindling

Simply put…

```html
<head>
    <script src="foo.js" type="text/javascript"></script>
    <script src="bar.js" type="text/javascript"></script>
    <script src="baz.js" type="text/javascript"></script>
    <script src="bam.js" type="text/javascript"></script>
</head>
```

```html
<head>
    <script src="bundle.js" type="text/javascript"></script>
</head>
```
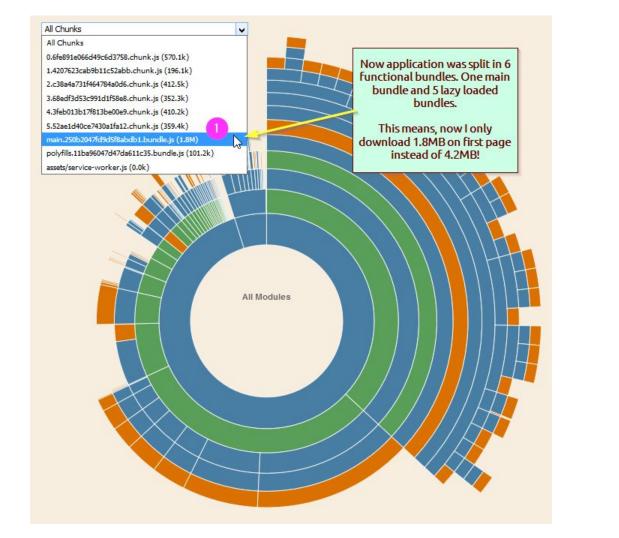
It provides a common set of tools for managing application complexity

All Chunks

All Chunks
0.6fe891e066d49c6d3758.chunk.js (570.1k)
1.4207623cab9b11c52abb.chunk.js (196.1k)
2.c38a4a731f464784a0d6.chunk.js (412.5k)
3.68edf3d53c991d1f58e8.chunk.js (352.3k)
4.3feb013b17f813be00e9.chunk.js (410.2k)
5.52ae1d40ce7430a1fa12.chunk.js (359.4k)
main.250b2047fd9d5f8abdb1.bundle.js (1.8M)
polyfills.11ba96047d47da611c35.bundle.js (101.2k)
assets/service-worker.js (0.0k)

All Modules

Now application was split in 6 functional bundles. One main bundle and 5 lazy loaded bundles.

This means, now I only download 1.8MB on first page instead of 4.2MB!

Because of its robust dependency graph architecture, powerful developer tooling can be built on top of it
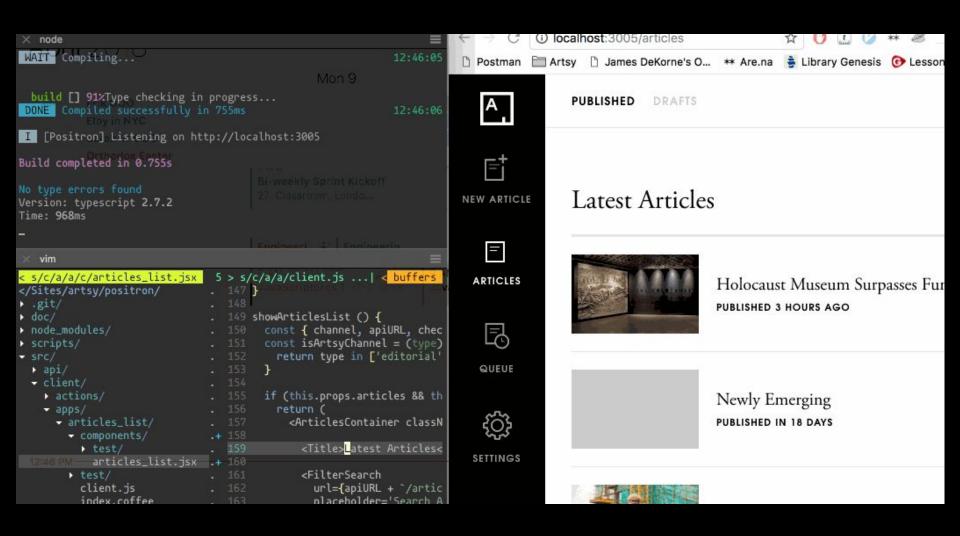
Log

```
BabelLoaderError: SyntaxError: Unexpected token (17:19)

  15 | import FeaturePage from 'containers/FeaturePage/Loadable';
  16 | import NotFoundPage from 'containers/NotFoundPage/Loadable';
> 17 | import Header from components/Header';
     |                    ^
  18 | import Footer from 'components/Footer';
  19 |
```

Status

Failed

Operation

idle (3s)

Progress

0%

Modules

`1:main.js`

Name
Size (min+gz)   Percentage

react-dom@15.6.1
72.66 KB   22.8%

core-js@2.5.1
69.35 KB   21.7%

intl@1.2.5
21.33 KB   6.69%

react-router@4.2.0
16.88 KB   5.29%

immutable
15.25 KB   4.78%

Assets

Name   Size

icon-192x192.png   17.46 KB

Problems

`1:main.js`

No problems detected!

And the overall developer experience highly optimized

```
WAIT  Compiling...                                    12:46:05

build [] 91%Type checking in progress...
DONE  Compiled successfully in 755ms                  12:46:06

I [Positron] Listening on http://localhost:3005

Build completed in 0.755s

No type errors found
Version: typescript 2.7.2
Time: 968ms
```

```
< s/c/a/a/c/articles_list.jsx   5 > s/c/a/a/client.js ...| < buffers
</Sites/artsy/positron/          . 147  }
▶ .git/                          . 148
▶ doc/                           . 149  showArticlesList () {
▶ node_modules/                  . 150    const { channel, apiURL, chec
▶ scripts/                       . 151    const isArtsyChannel = (type)
▼ src/                           . 152      return type in ['editorial'
  ▶ api/                         . 153  }
  ▼ client/                      . 154
    ▶ actions/                   . 155    if (this.props.articles && th
    ▼ apps/                      . 156      return (
      ▼ articles_list/           . 157        <ArticlesContainer classN
        ▼ components/          .+ 158
          ▶ test/                  159          <Title>Latest Articles<
          articles_list.jsx    .+ 160
        ▶ test/                    161          <FilterSearch
      client.js                  . 162            url={apiURL + `/artic
      index.coffee               . 163            placeholder='Search A
```

localhost:3005/articles

Postman Artsy James DeKorne's O... ** Are.na Library Genesis Lesson

PUBLISHED DRAFTS

A

NEW ARTICLE

Latest Articles

ARTICLES

Holocaust Museum Surpasses Fu

PUBLISHED 3 HOURS AGO

QUEUE

Newly Emerging

PUBLISHED IN 18 DAYS

SETTINGS

However, Webpack configuration can be complex

David Fox Powell Follow

@dtothefp => Hyperloop One

May 22, 2015 · 12 min read

Why Can't Anyone Write a Simple Webpack Tutorial?

It is designed for planet scale

But it can also be simple

webpack --mode development --output dist/bundle.js --watch

vim

1 > s/index.js

</damassi/Desktop/webpack/ . 1 []
▸ node_modules/ .
▾ src/ .
    index.js .
  package.json .
  yarn.lock .

..sktop/webpack

webpack % _

Webpack 4 can run without configuration via the "development" --mode flag

What does a basic Webpack config look like?

```
module.exports = {
  mode: 'development' // or production
}
```

```
module.exports = {
  mode: 'production',
  entry: './src/index.js'
}
```

```javascript
module.exports = {
  mode: 'production',
  entry: './src/index.js',
  output: {
    path: path.join(__dirname, 'dist'),
    filename: 'bundle.js'
  }
}
```

```javascript
module.exports = {
  mode: 'production',
  entry: './src/index.js',
  output: {
    path: path.join(__dirname, 'dist'),
    filename: 'bundle.js'
  },
  module: {
    rules: [
      {
        test: /\.js$/,
        include: /src/,
        use: {
          loader: 'babel-loader',
          options: {
            presets: [
              '@babel/env',
              '@babel/stage-3',
              '@babel/react'
            ]
          }
        }
      }
    ]
  }
}
```

What about a server?

webpack-dev-server

Use webpack with a development server that provides live reloading. This should be used for **development only**.

It uses webpack-dev-middleware under the hood, which provides fast in-memory access to the webpack assets.

```json
"scripts": {
  "build": "webpack",
  "build:prod": "webpack --mode production",
  "start": "webpack-dev-server --hot"
},
```

```
6_babel-webpack [6-babel-webpack] % yarn start
yarn run v1.5.1
$ webpack-dev-server --hot
i 「wds」: Project is running at http://localhost:8080/
i 「wds」: webpack output is served from /dist/
i 「wdm」:
i 「wdm」: Compiled successfully.
```

Thanks!