

**webpack**  
**ACADEMY**

**WEBPACK FUNDAMENTALS**

<https://github.com/thelarkinn/webpack-workshop-2018>

<https://frontendmasters.com/courses/webpack-fundamentals/>

# PROGRAM MANAGER

MICROSOFT WEB PLATFORM, EDGE DEV TOOLS

# MAINTAINER + ADVOCATE

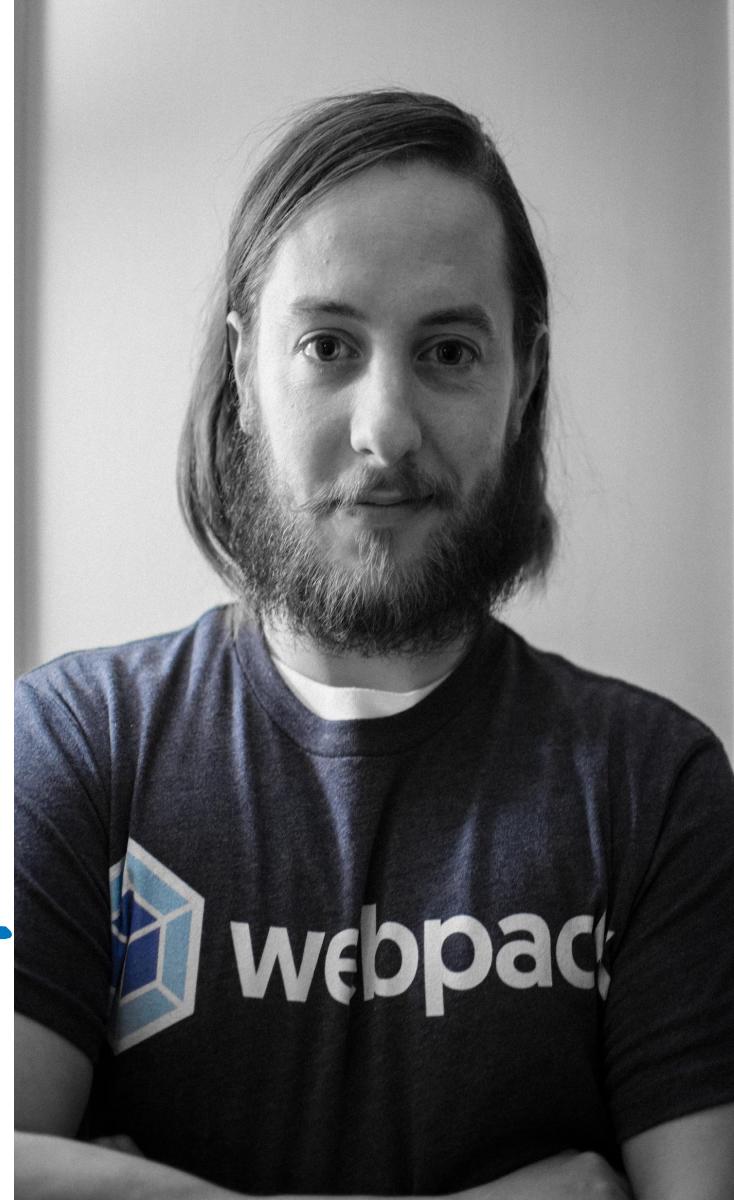
WEBPACK

# CORE TEAM

ANGULAR / ANGULAR-CLI

# EVANGELIST

OPEN SOURCE SUSTAINABILITY





# BACKGROUND

Former Tech Support Rep.  
gone rogue turned Software  
Engineer / Web Developer  
who got tired of never being  
able to really help the  
customer he served.

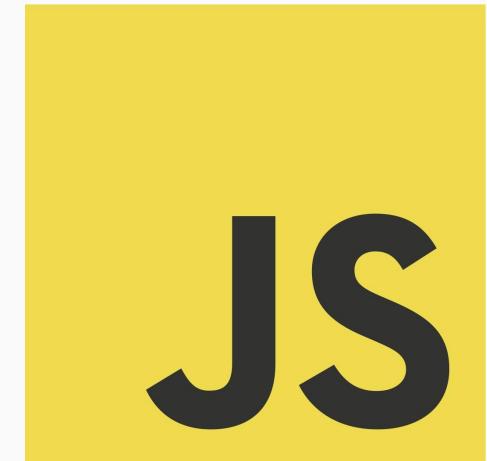
**Languages:** Ruby,  
Objective-C, Swift, Javascript.

**Other:** Woodworker,  
farmer, IoT



## Objective

*C-Programming*



# SUSTAINABLE OPEN SOURCE PRACTICES

JAVASCRIPT

BUILDING  
CONTRIBUTORS,  
COMMUNITY, AND  
ECOSYSTEM



Sean Larkin

TheLarkInn

User Experience Developer  
@mutualofomaha. Javascript, Angular, Ruby, Webpack, Typescript. @webpack core team. @angular cli core team.

@mutualofomaha @webpack...

Lincoln, NE

sean.larkin@cuw.edu

<https://careers.stackoverflow...>

## Organizations



Overview

Repositories 164

Stars 150

Followers 413

Following 53

## Pinned repositories

Customize your pinned repositories

### webpack/webpack

A bundler for javascript and friends. Packs many modules into a few bundled assets. Code Splitting allows to load parts for the application on demand. Through "loaders," modules can be CommonJs, AM...

JavaScript ★ 24k 2.8k

### angular/angular-cli

CLI tool for Angular

TypeScript ★ 7.2k 1.4k

### angular-starter-es6-webpack

This is an Angular Starter App with component and service generators using gulp for easy component development. Uses Karma-Mocha-Chai as testing suite and Babel Loader and Webpack for ES6

JavaScript ★ 71 49

### angular2-template-loader

Chain-to loader for webpack that inlines all html and style's in angular2 components.

JavaScript ★ 100 40

### webpack-developer-kit

webpack dev kit for writing custom plugins and loaders on the fly. Education/Exploration tool as well.

JavaScript ★ 55 7

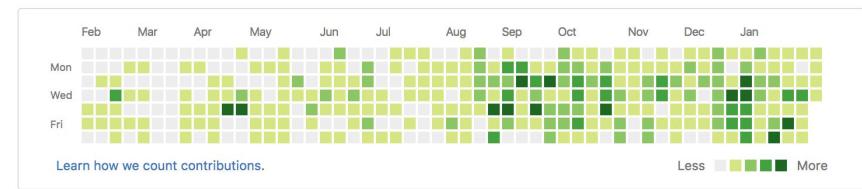
### LazyParseWebpackPlugin

(v8-lazy-parse-webpack-plugin) This is a webpack plugin designed to exploit the V8 engines treatment of functions with parens wrapped around them. This lazy loads the parsing decreasing initial loa...

JavaScript ★ 84 6

1,434 contributions in the last year

Contribution settings ▾



**@THELARKINN**

[Github](#) - [Medium](#) - [Codepen](#) - [Stack Overflow](#) - [LinkedIn](#) - [Twitter](#)

# ASK ME ANYTHING

<http://github.com/thelarkinn/ama>

# EXPECTATIONS

WHY WEBPACK? - HISTORY OF WEB PERFORMANCE + JAVASCRIPT

GETTING STARTED - SETUP, INSTALLATION, SCRIPTS, AND CLI

THE CORE CONCEPTS

STARTING OUT RIGHT

THE ESSENTIALS

PUTTING IT TO PRACTICE

TRIAGE AND DEBUG



# CHAPTER 1: WHY?

ORIGINS

JAVASCRIPT - IT'S  
JUST SCRIPTS!

```
const button = document.createElement('button');
button.innerText = "My button!";
document.body.appendChild(button);
```

2 WAYS TO LOAD

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>
  <script src="index.js"></script>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <script>
        var foo = "hello world!";

        console.log(foo);
    </script>
</body>
</html>
```

# PROBLEMS

DOESN'T SCALE

TOO MANY SCRIPTS

▲ Max Number of default simultaneous persistent connections per server/proxy:

339

Firefox 2: 2  
Firefox 3+: 6  
Opera 9.26: 4  
Opera 12: 6  
Safari 3: 4  
Safari 5: 6  
IE 7: 2  
IE 8: 6  
IE 10: 8  
Chrome: 6

The limit is per-server/proxy, so your wildcard scheme will work.

FYI: this is specifically related to HTTP 1.1; other protocols have separate concerns and limitations (i.e., SPDY, TLS, HTTP 2).

TOO MANY SCRIPTS

# UNMAINTAINABLE SCRIPTS

SCOPE

SIZE

READABILITY

FRAGILITY

MONOLITH FILES

SOLUTION?

TIFE'S

IMMEDIATELY  
INVOKED  
FUNCTION  
EXPRESSION

```
/**  
 * Immediately Invoked Function Expression  
 */  
const whatever = (function(dataNowUsedInside) {  
    return {  
        someAttribute: "youwant"  
    }  
})(1)  
/**  
 * whatever.someAttribute  
 *  
 * > "youwant"  
 */
```

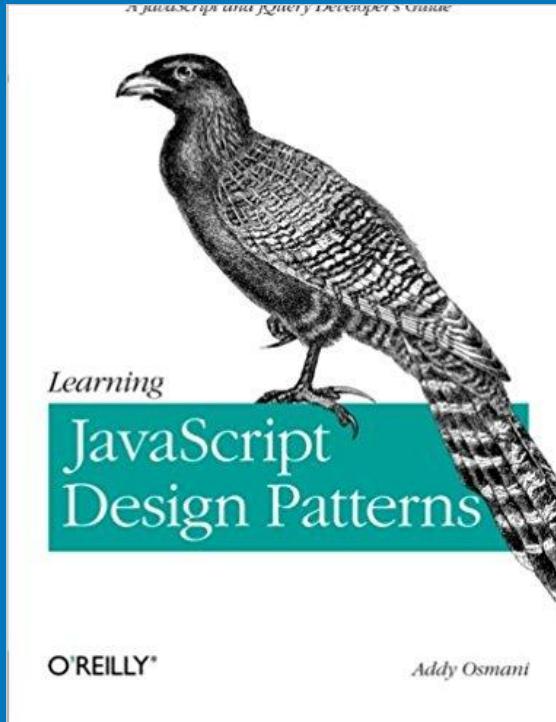
# REVEALING MODULE PATTERN

```
var outerScope = 1;
/**/
 * Immediately Invoked Function Expression
 */
const whatever = (function(dataNowUsedInside) {
    var outerScope = 4;
    return {
        someAttribute: "youwant"
    }
})(1)

console.log(outerScope);
/**/
 * console log returns 1! No inner scope leak!
*/
```

TREAT EACH FILE AS  
IIFE (REVEALING  
MODULE)

# MANY SIMILAR PATTERNS



## Table Of Contents

- Introduction
- What is a Pattern?
- "Pattern"-ity Testing, Proto-Patterns & The Rule Of Three
- The Structure Of A Design Pattern
- Writing Design Patterns
- Anti-Patterns
- Categories Of Design Pattern
- Summary Table Of Design Pattern Categorization
- JavaScript Design Patterns
  - Constructor Pattern
  - Module Pattern
  - Revealing Module Pattern
  - Singleton Pattern
  - Observer Pattern
  - Mediator Pattern
  - Prototype Pattern
  - Command Pattern
  - Facade Pattern
  - Factory Pattern
  - Mixin Pattern
  - Decorator Pattern
  - Flyweight Pattern
- JavaScript MV\* Patterns
  - MVC Pattern
  - MVP Pattern
  - MVVM Pattern
- Modern Modular JavaScript Design Patterns
  - AMD
  - CommonJS
  - ES Harmony
- Design Patterns In jQuery
  - Composite Pattern
  - Adapter Pattern
  - Facade Pattern
  - Observer Pattern
  - Iterator Pattern
  - Lazy Initialization Pattern
  - Proxy Pattern
  - Builder Pattern
- jQuery Plugin Design Patterns
- JavaScript Namespacing Patterns
- Conclusions
- References

# CONCATENATE!

WE CAN "SAFELY" COMBINE  
FILES WITHOUT CONCERN OF  
SCOPE COLLISION!\*

MAKE, GRUNT, GULP,  
BROCCOLI, BRUNCH,  
STEALJS

# PROBLEMS

FULL REBUILDS  
EVERYTIME!

# DEAD CODE

CONCAT DOESN'T HELP TIE USAGES ACROSS FILES

## The cost of small modules

Posted August 15, 2016 by Nolan Lawson in performance, Web. [78 Comments](#)

**Update (30 Oct 2016):** since I wrote this post, a bug was found in the benchmark which caused Rollup to appear slightly better than it would otherwise. However, the overall results are not substantially different (Rollup still beats Browserify and Webpack, although it's not quite as good as Closure anymore), so I've merely updated the charts and tables. Additionally, the benchmark now includes the RequireJS and RequireJS Almond bundlers, so those have been added as well. To see the original blog post without these edits, check out this [archived version](#).

About a year ago I was refactoring a large JavaScript codebase into smaller modules, when I discovered a depressing fact about Browserify and Webpack:

“The more I modularize my code, the bigger it gets. 😞”  
– Nolan Lawson

Later on, Sam Saccone published some excellent research on [Tumblr](#) and [Imgur](#)'s page load performance, in which he noted:

“Over 400ms is being spent simply walking the Browserify tree.”  
– Sam Saccone

In this post, I'd like to demonstrate that small modules can have a surprisingly high performance cost depending on your choice of bundler and module system. Furthermore, I'll explain why this applies not only to the modules in your own codebase, but also to the modules *within dependencies*, which is a rarely-discussed aspect of the cost of third-party code.

LOTS  
OF  
IIFE'S  
ARE  
SLOW

~~DYNAMIC~~ ~~LOADING?~~

# BIRTH OF JAVASCRIPT MODULES



# COMMONJS ( MODULES 1.0 )

```
// index.js
const path = require("path"); // used for builtin Node.js modules
const {add, subtract} = require("./math"); // or also used modules from another file

const sum = add(5, 5);
const difference = subtract(10, 4);

console.log(sum, difference);

/**
 *
 * math.js (has two named exports {add, subtract})
 *
 *
 */
const divideFn = require("./division");

exports.add = (first, second) => first + second;
exports.subtract = (first, second) => first - second;
exports.divide = divideFn;

/**
 *
 * division.js
 *
 *
 * has a default exports "divide"
 */
module.exports = (first, second) => first/second;
```

# STATIC ANALYSIS

# NPM+NODE+MODULES

## MASS DISTRIBUTION

# PROBLEMS

NO BROWSER SUPPORT

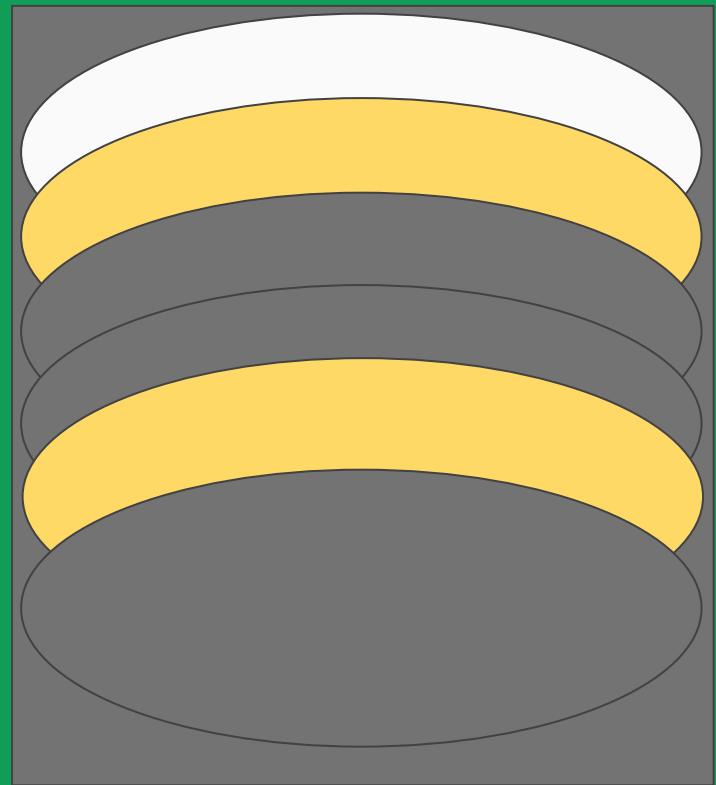
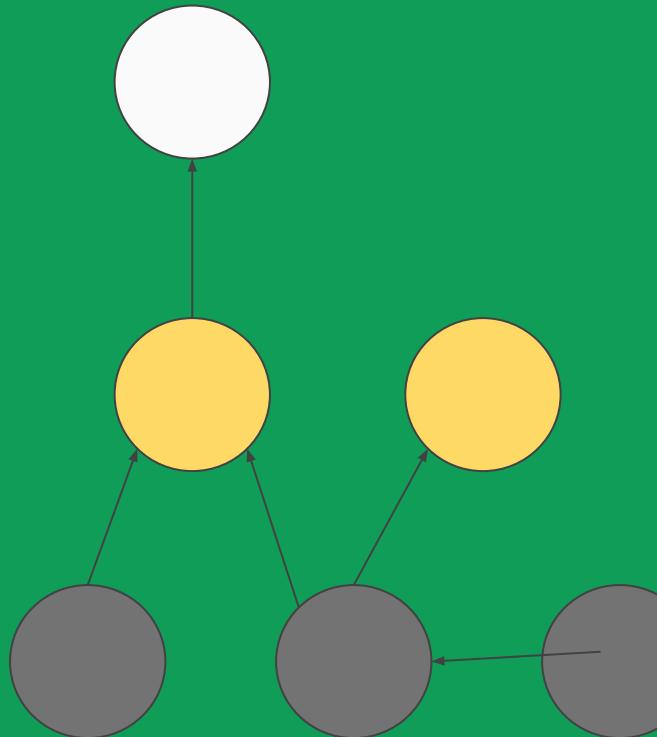
# NO LIVE BINDINGS

PROBLEMS WITH CIRCULAR REFERENCES

SYNC MODULE RESO,  
LOADER (SLOW)

NO  
BROWSER  
SUPPORT

SOLUTION?



# BUNDLERS / LINKERS

BROWSERIFY (STATIC)  
REQUIREJS (LOADER)  
SYSTEMJS (LOADER)

# PROBLEMS

# COMMONJS

```
//loading module  
var _ = require('lodash');
```

```
//declaring module  
module.exports = someValue;
```

NO STATIC / ASYNC /  
LAZY LOADING (ALL  
BUNDLES UP FRONT)

COMMONJS BLOAT  
TOO DYNAMIC

NOT EVERYONE WAS  
SHIPPING COMMONJS.

# AMD

```
define('myAwesomeLib', ['lodash',
'someDep'],
function (_, someDep) {
    return { ... }
});

```

# AMD + COMMONJS

```
define( function(require, exports, module) {  
  var _ = require('lodash');  
  
  //..do things  
  module.exports = someLib;  
});
```

# PROBLEMS

TOO DYNAMIC OF LAZY  
LOADING (MOMENTJS)

**AWKWARD NON  
STANDARD SYNTAX (NO  
REAL MODULE SYSTEM)**

SOLUTION?

# ESM

```
import {uniq, forOf, bar} from 'lodash-es'  
import * as utils from 'utils';  
  
export const uniqConst = uniq([1,2,2,4]);
```

REUSABLE  
ENCAPSULATED  
ORGANIZED  
CONVENIENT

REUSABLE  
ENCAPSULATED  
ORGANIZED  
CONVENIENT

# PROBLEMS

**ESM FOR NODE?**

# HOW DO THEY WORK IN THE BROWSER?



**ESM FOR BROWSER IS  
VERY VERY VERY SLOW**

# EVERY LIBRARY IS DIFFERENT...

Library authors use the module types that they like  
and choose

AND THIS IS JUST FOR  
JAVASCRIPT...

Each and every other filetype until now has had to have specific ways  
to process it.

WOULDN'T IT BE NICE...



# webpack



WEBPACK IS A MODULE BUNDLER

LETS YOU WRITE ANY MODULE  
FORMAT (MIXED!), COMPILES  
THEM FOR THE BROWSER

SUPPORTS STATIC ASYNC  
BUNDLING

RICH, VAST, ECOSYSTEM

THE MOST PERFORMANT WAY TO  
SHIP JAVASCRIPT TODAY

# WEBPACK - HOW TO USE IT?

## CONFIG

(webpack.config.js) Yes, it's a module too!!!

```
module.exports = {  
  entry: {  
    vendor: './src/vendors.ts',  
    main: './src/main.browser.ts'  
  },  
  output: {  
    path: 'dist/',  
    filename: '[name].bundle.js'  
  }  
}
```

# WEBPACK - HOW TO USE IT?

## WEBPACK CLI

```
$> webpack <entry.js>  
<result.js> --colors  
    --progress
```

```
$> webpack-dev-server  
    --port=9000
```

# WEBPACK - HOW TO USE IT?

NODE API

```
var webpack = require("webpack");

// returns a Compiler instance
webpack({
    // configuration object here!
}, function(err, stats) {
    // ...
    // compilerCallback
    console.error(err);
});
```

QUESTIONS?

**BREAK!**

# CHAPTER 2 - FROM SCRATCH

[github.com/thelarkinn/webpack-workshop-2018](https://github.com/thelarkinn/webpack-workshop-2018)

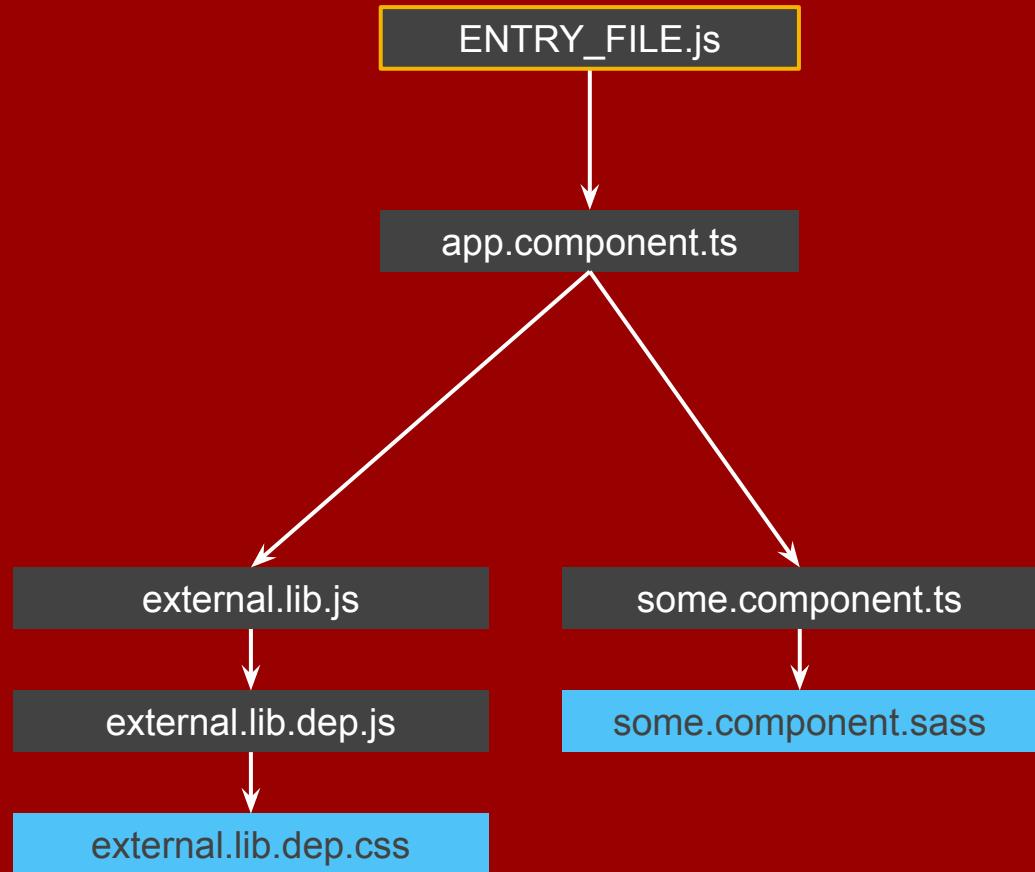
# THE CORE CONCEPTS

ENTRY

# THE CORE CONCEPTS: ENTRY

The first javascript file to load to “kick-off” your app.

webpack uses this as the starting point



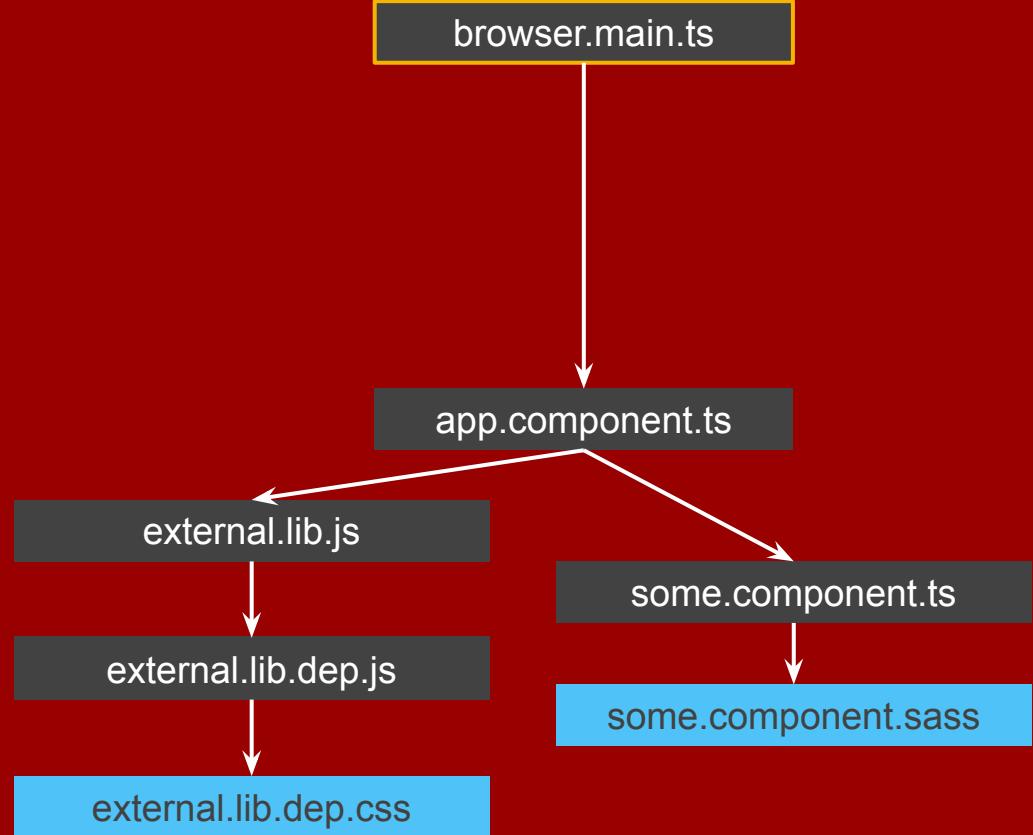
# THE CORE CONCEPTS: ENTRY

```
//webpack.config.js
module.exports = {
  entry: './browser.main.ts',
  //...
}

//browser.main.ts
import {
  Component
} from '@angular/core';

import {
  App
} from './app.component';
bootstrap(App, []);

//app.component.ts
@Component({...})
export class App {};
```



# THE CORE CONCEPTS: ENTRY

```
//webpack.config.js
module.exports = {
  entry: './browser.main.ts',
  //...
}

//browser.main.ts
import {Component} from
'@angular/core';

import {App} from
'./app.component';

bootstrap(App, []);

//app.component.ts
@Component({...})
export class App {};
```

browser.main.ts

app.component.ts

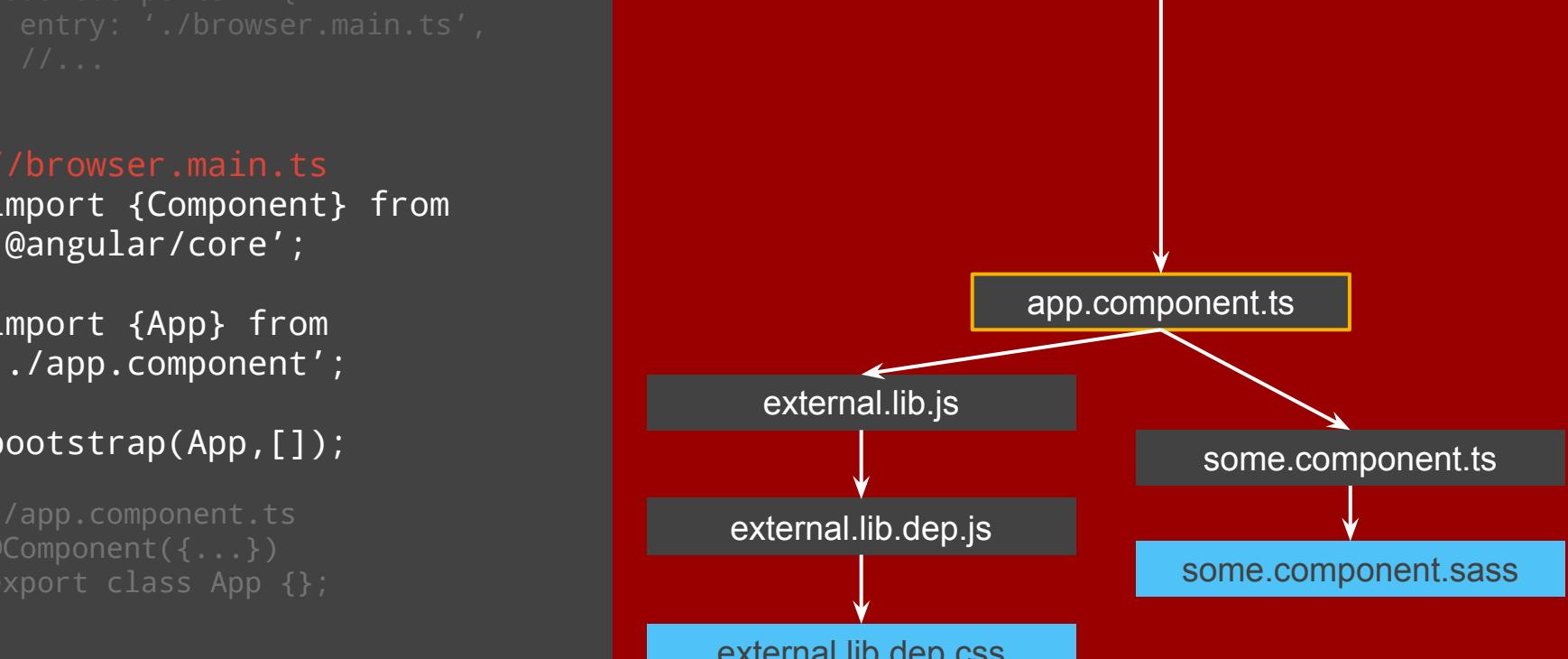
external.lib.js

external.lib.dep.js

external.lib.dep.css

some.component.ts

some.component.sass



# THE CORE CONCEPTS

## ENTRY

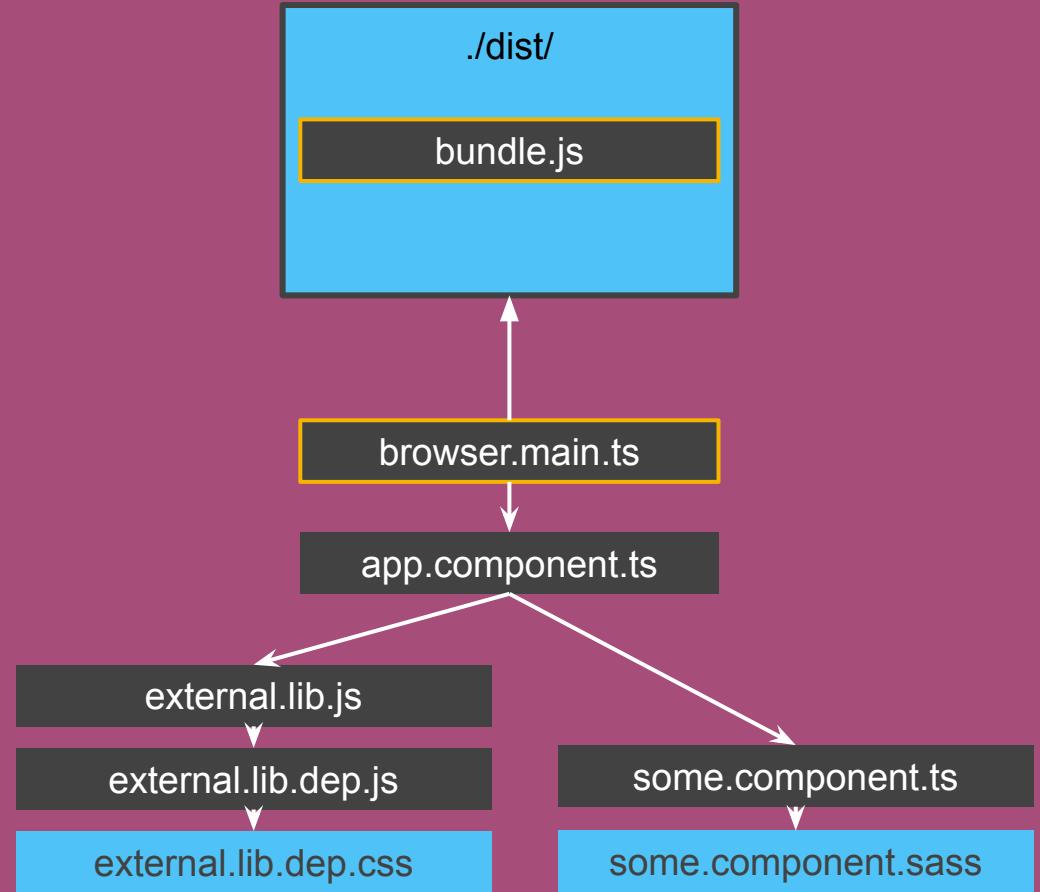
Tells webpack WHAT (files) to load for the browser;  
Compliments the *Output* property.

OUTPUT

# THE CORE CONCEPTS: OUTPUT

```
//webpack.config.js
module.exports = {
  entry: './browser.main.ts',
  output: {
    path: './dist',
    filename: './bundle.js',
  },
  //...
}

//Generates bundle.js
```



# THE CORE CONCEPTS

## ENTRY OUTPUT

Tells Webpack WHERE and HOW to distribute bundles (compilations). Works with Entry.

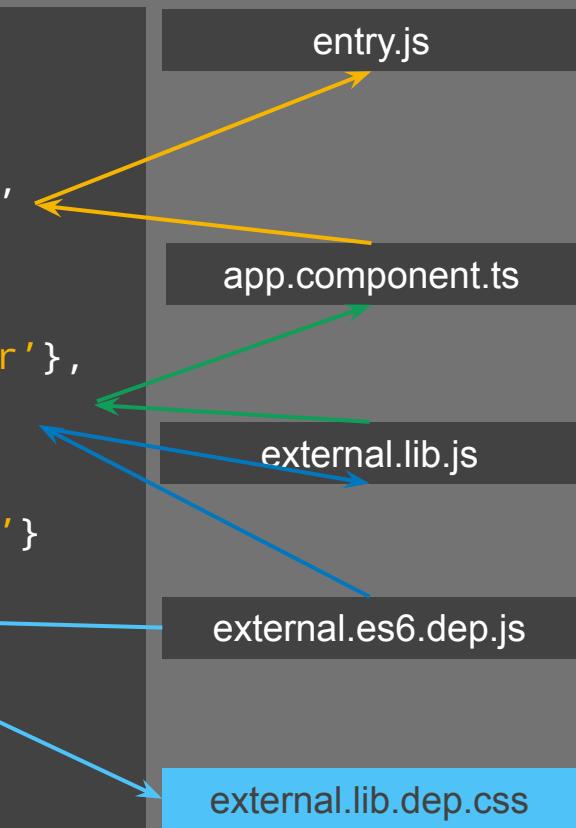
LOADERS +  
RULES

# THE CORE CONCEPTS: LOADERS

Tells webpack how to modify files before its added to dependency graph

Loaders are also javascript modules (*functions*) that takes the source file, and returns it in a [modified] state.

```
module: {  
  rules: [  
    {test: /\.ts$/, use: 'ts-loader'},  
  
    {test: /\.js$/, use: 'babel-loader'},  
  
    {test: /\.css$/, use: 'css-loader'}  
  ],  
}
```



# THE CORE CONCEPTS: LOADERS

```
module: {  
  rules: [  
    {  
      test: regex,  
      use: (Array/String/Function)  
      include: RegExp[],  
      exclude: RegExp[],  
      issuer: (RegExp/String)[],  
      enforce: "pre"/"post"  
    },  
  ],  
}
```

## test

A *regular expression* that instructs the compiler which files to run the loader against.

## use

An array/string/function that returns loader objects.

## enforce

Can be “pre” or “post”, tells webpack to run this rule before or after all other rules

# THE CORE CONCEPTS: LOADERS

```
module: {  
  rules: [  
    {  
      test: /\.ts$/,  
      use: [  
        'awesome-typescript-loader',  
        'ng2-asset-loader'  
      ],  
      include: /some_dir_name/,  
      exclude: [/\.(spec|e2e)\.ts$/],  
    },  
  ],  
}
```

## include

An *array of regular expression* that instruct the compiler which folders/files to include. *Will only search paths provided with the include.*

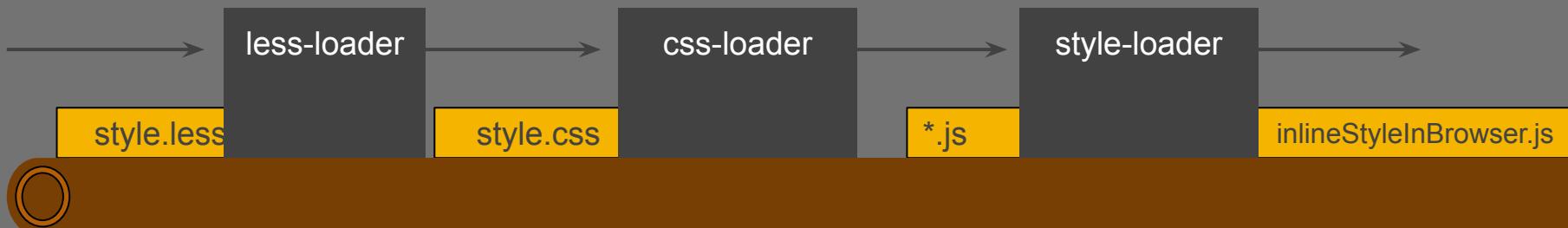
## exclude

An *array of regular expression* that instructs the compiler which folders/files to ignore.

# THE CORE CONCEPTS: LOADERS

## CHAINING LOADERS

```
rules: [  
  {  
    test: /\.less$/,
    use:[ 'style', 'css', 'less' ]  
  }  
]
```



# THE CORE CONCEPTS: LOADERS

json, hson, raw, val, to-string, imports, exports, expose, script, apply, callback, ifdef-loader, source-map, sourceMappingURL, checksum, cowsay, dsv, glsl, glsl-template, render-placement, xml, svg-react, svg-url, svg-as-symbol, symbol, base64, ng-annotate, node, required, icons, markup-inline, block-loader, bundler-configuration, console, solc, .sol, web3, includes, combine, regexp-replace, file, url, extract, worker, shared-worker, serviceworker, bundle, require.ensure, promise, async-module, bundle, require.ensure, react-proxy, react-hot, image, file, url, img, base64-image, responsive, srcset, svgo, svg-sprite, symbol, svg-fill, fill, line-art, baggage, polymer, uglify, html-minify, vue, toJSON, zip-it, file, lzstring, modernizr, s3, path-replace, react-intl, require.ensure, font-subset, w3c-manifest, web-app-manifest, manifest-scope, coffee, coffee-jsx, coffee-redux, json5, es6, esnext, babel, regenerator, livescript, sweetjs, traceur, ts, typescript, awesome-typescript, webpack-typescript, purs, oj, elm-webpack, miel, wisp, sibilant, ion, html, dom, riot, pug, jade-html, jade-react, virtual-jade, virtual-dom, template-html, handlebars, handlebars-template-loader, dust, ractive, jsx, react-templates, em, ejs, ejs-html, mustache, yaml, yml, react-markdown, front-matter, markdown, remarkable, markdown-it, markdownattrs, ng-cache, ngttemplate, hamlc, haml, jinja, nunjucks, soy, smarty, swagger, template-string, ect, tmodjs, layout, swig, twig, mjml-, bootstrap-webpack, font-awesome-webpack, bootstrap-sass, bootstrap, bootstrap, font-awesome, style, isomorphic-style, style-loader, css, cess, less, sass, stylus, csso, rework, postcss, autoprefixer, namespace-css, fontgen, classnames, theo, bulma, css-to-string, css-loader, po, po2mo, format-message, jsxlate, angular-gettext, json, angular-gettext, webpack-angular-translate, angular-gettext-extract, .pot, gettext, preprocessor, amdi18n-loader, .json, .js, .coffee, sprockets-preloader, properties, transifex, mocha, coverjs, istanbul-instrumenter, ibrik-instrumenter, eslint, jshint, jscs, standard, inject, transform, falafel, image-size, csslint, coffeelint, tslint, parker, sjsp, amdcheck, manifest, gulp-rev, html-test, stylelint, stylefmt, scsslint, htmlhint, documentation, sassdoc, performance-loader



**Tom Dale**

@tomdale

Following

# Where is your god now?

Kornel @kornelski

Run PHP in the browser (with source maps!) via Babel transform. This is the stupidest project I've ever done: [gitlab.com/kornelski/babe...](https://gitlab.com/kornelski/babe...)

7:20 AM - 11 Jul 2017 from [Manhattan, NY](#)

82 Retweets 202 Likes



5 82 202



Tweet your reply



**Stephen Fluin** @stephenfluin · Jul 11

Replying to [@tomdale](#) @TheLarkInn

Now if there was only some way to use PHP to se

1

1

2

3

4

5

6

7

8

9

10

11



**Matt Davis** @johnmattdavis · Jul 11

This idea continues to fill me with so many ambiv

1

1

2

3

4

5

6

7

8

9

10

11



**Boz-Textual** @boztek · Jul 11

Replying to [@tomdale](#) @TheLarkInn

Who do you think provided the source maps?

1

1

2

3

4

5

6

7

8

9

10

11

```
test.php x
1 Hello <?php
2 define('FOO', max(floatval($c),
3 $bar['x'])[][$b] = json_encode(
4 class Foo extends Bar\Baz {
5     var $z = "hello" . "world";
6     function __construct($some = a
7         parent::__construct(func_get
8             self::$k} = "{$this->z[10]
9
10
11 }
```

▶ (anonymous) test.php:3

FOO: 3

# THE CORE CONCEPTS

## ENTRY OUTPUT LOADERS

Tells Webpack HOW to interpret and translate files.  
Transformed on a per-file basis before adding to the dependency graph

PLUGINS

# THE CORE CONCEPTS: PLUGINS

Objects (with an `apply` property)

Allow you to hook into the entire compilation lifecycle

webpack has a variety of built in plugins

# THE CORE CONCEPTS: PLUGINS

```
function BellOnBundlerErrorPlugin () { }

BellOnBundlerErrorPlugin.prototype.apply = function(compiler) {
  if (typeof(process) !== 'undefined') {

    // Compiler events that are emitted and handled
    compiler.plugin('done', function(stats) {
      if (stats.hasErrors()) {
        process.stderr.write('\x07');
      }
    });

    compiler.plugin('failed', function(err) {
      process.stderr.write('\x07');
    });
  }
}

module.exports = BellOnBundlerErrorPlugin;
```

## Basic Plugin Example

A plugin is an ES5  
'class' which  
implements an *apply*  
function.

The compiler uses it to  
emit events.

# THE CORE CONCEPTS: PLUGINS

```
// require() from node_modules or webpack or local file
var BellOnBundlerErrorPlugin = require('bell-on-error');
var webpack = require('webpack');

module.exports = {
  //...
  plugins: [
    new BellOnBundlerErrorPlugin(),
    // Just a few of the built in plugins
    new webpack.optimize.CommonsChunkPlugin('vendors'),
    new webpack.optimize.UglifyJsPlugin()
  ]
  //...
}
```

## How to use Plugins

*require()* plugin from *node\_modules* into config.

add *new instance of plugin* to *plugins* key in config object.

provide additional info for arguments

[CLICK HERE TO SEE THE LIST OF PLUGINS](#)

# THE CORE CONCEPTS: PLUGINS

5b25024 on Jun 8

 TheLarkInn Merge

80% of webpack is made up of its own plugin system

17 contributors



300 lines (280 sloc) | 10.6 KB

Raw Blame History

```
1  /*
2   * MIT License http://www.opensource.org/licenses/mit-license.php
3   * Author Tobias Koppers @sokra
4  */
5  var assign = require("object-assign");
6  var OptionsApply = require("./OptionsApply");
7
8  var LoaderTargetPlugin = require("./LoaderTargetPlugin");
9  var FunctionModulePlugin = require("./FunctionModulePlugin");
10 var EvalDevToolModulePlugin = require("./EvalDevToolModulePlugin");
11 var SourceMapDevToolPlugin = require("./SourceMapDevToolPlugin");
12 var EvalSourceMapDevToolPlugin = require("./EvalSourceMapDevToolPlugin");
13
14 var EntryOptionPlugin = require("./EntryOptionPlugin");
15 var RecordIdsPlugin = require("./RecordIdsPlugin");
```



MARIEPHANTOMHIVE.TUMBLR.COM



# THE CORE CONCEPTS

ENTRY  
OUTPUT  
LOADERS  
PLUGINS

Adds additional functionality to *Compilations*(optimized bundled modules).  
More powerful w/ more access to CompilerAPI. Does everything else you'd  
ever want to in webpack.

EXERCISE TIME

# CHAPTER 3 - STARTING OUT RIGHT