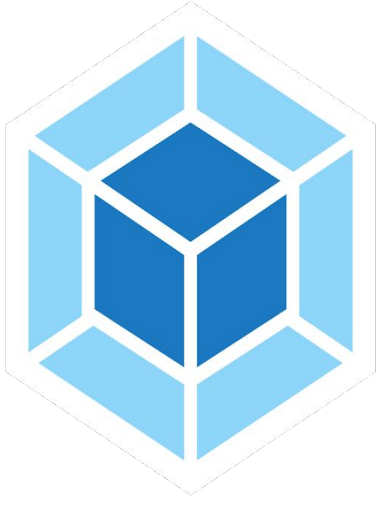


webpack ACADEMY

# 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


# Where most people think Nebraska is...

# 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*

JS

# 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

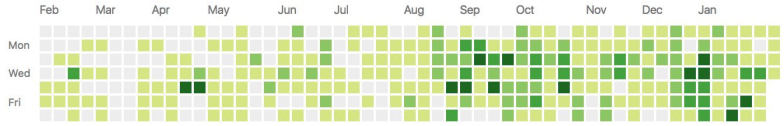
**angular-starter-es6-webpack**
This is an Angular Starter App with component and service generators using gulp for easy component development. Uses Karma-Mocha-Chai as testing suite and Babel Loader and Webpack for ES6
JavaScript ★ 71 49

**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




Learn how we count contributions.
Less More

# @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

```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 src="index.js"></script>
</body>
</html>
```

```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

339

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

```
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?

IIFE'S

# Immediately Invoked Function Expression

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

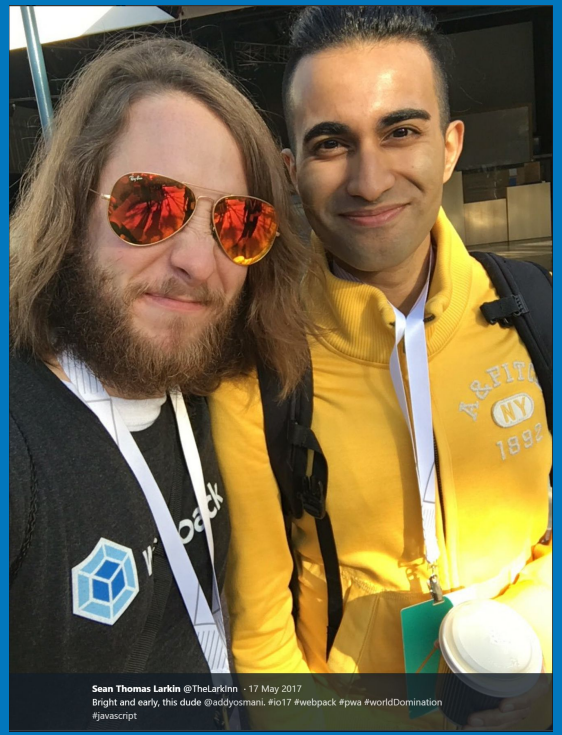
# Revealing Module pattern

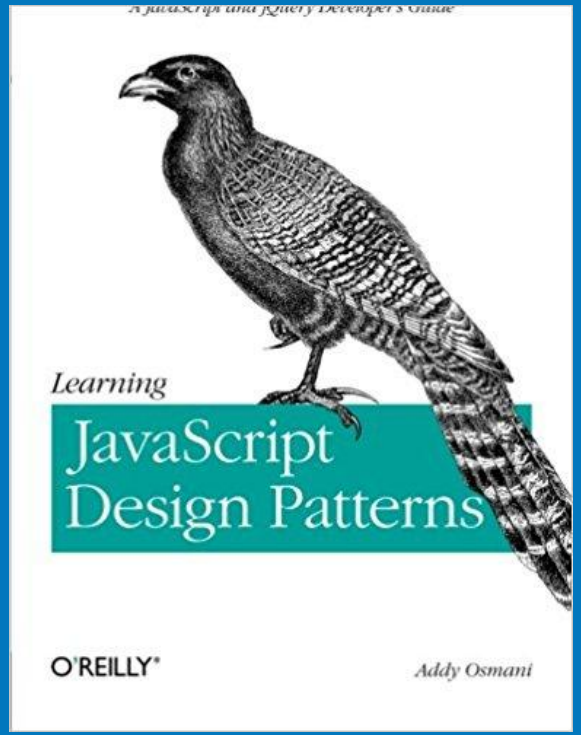
```
var outerScope = 1;
/**
 * Immediately Invoked Function Expression
 */
const whatever = (function(dataNowUsedInside) {
    var outerScope = 4;
    return {
        someAttribute: "youwant"
    }
})(1)

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

# Treat each file as IIFE (Revealing Module)

# Many Similar Patterns




## Table Of Contents

# 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

« On joining Microsoft Edge and moving to Seattle

Progressive enhancement isn't dead, but it smells funny »

# 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


node
JS
®

# 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

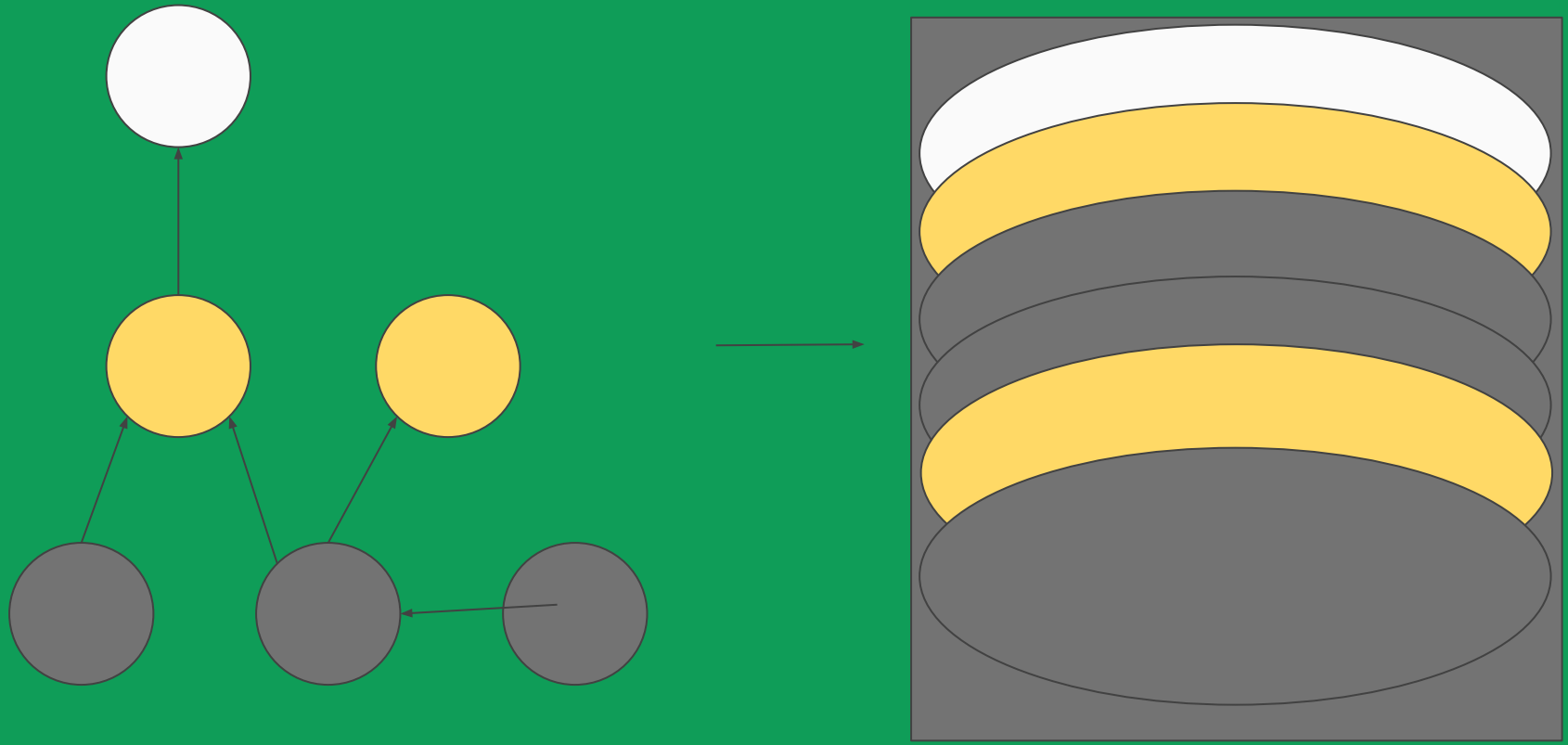
# No Live Bindings

## 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

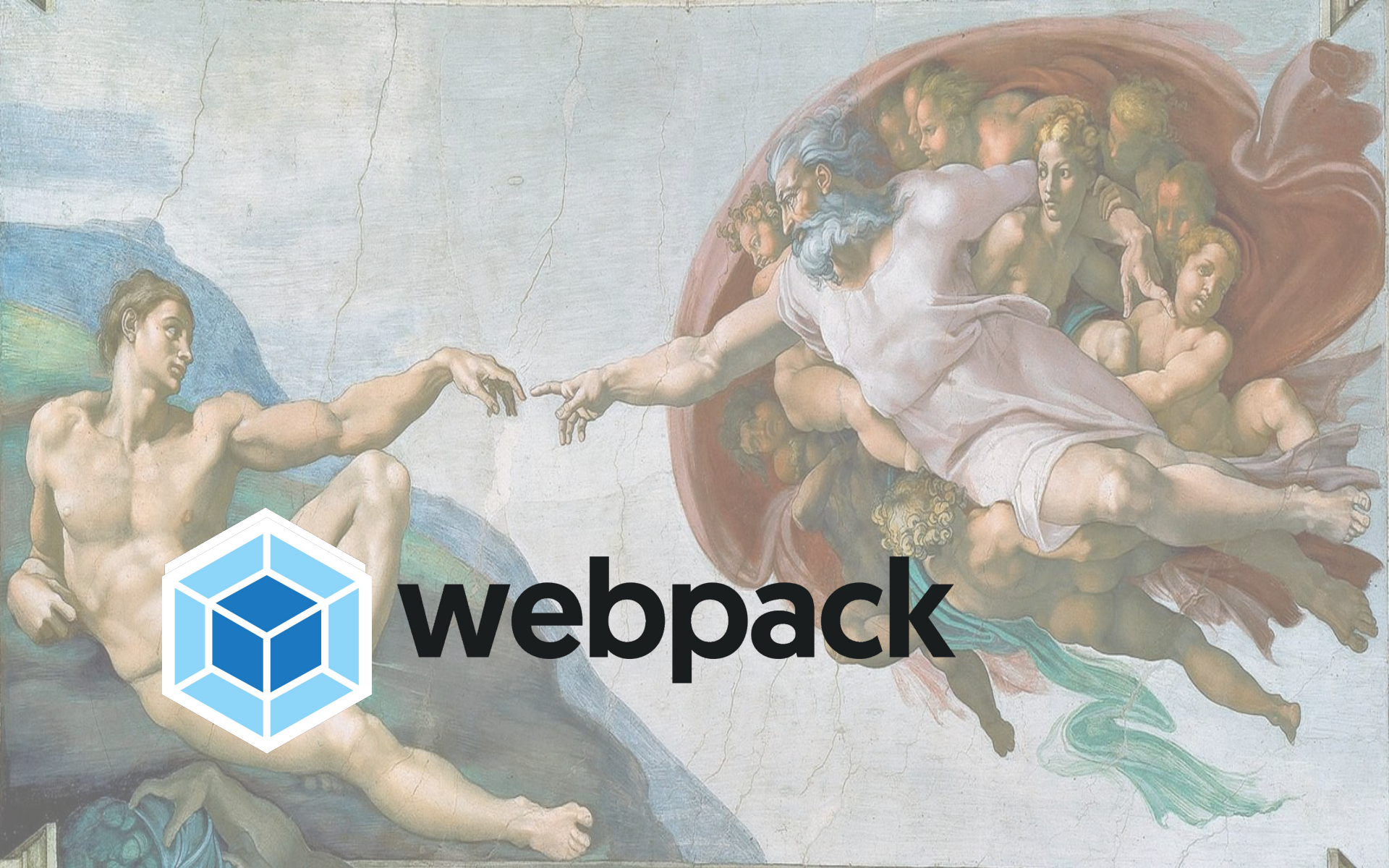
# Every library is different...

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

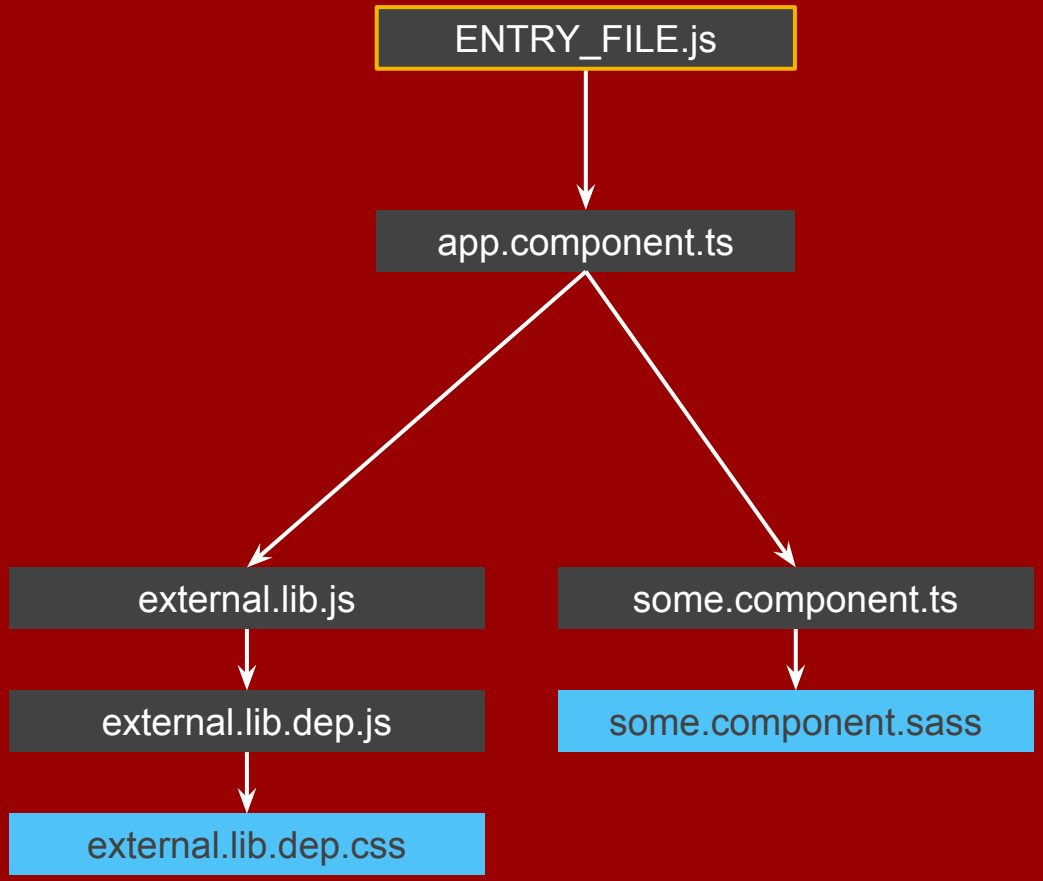
# The Core Concepts

# Entry

# The Core Concepts: Entry

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

webpack uses this as the starting point



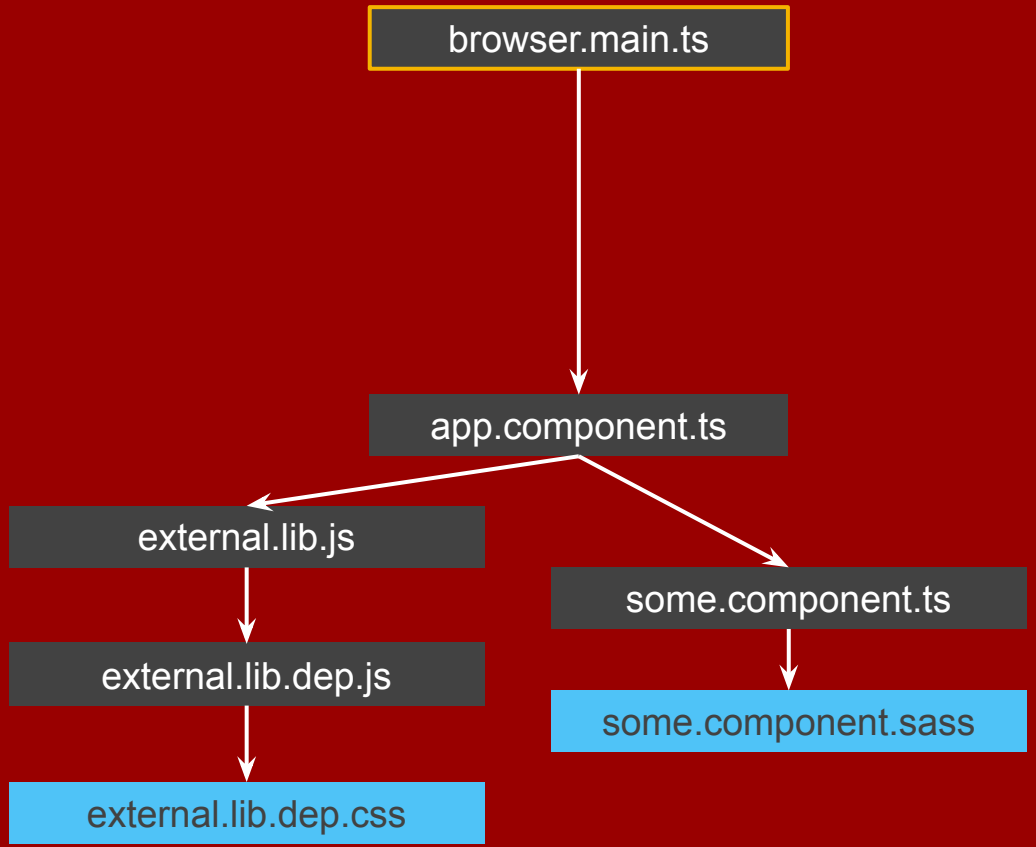
# The Core Concepts: Entry

```
//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

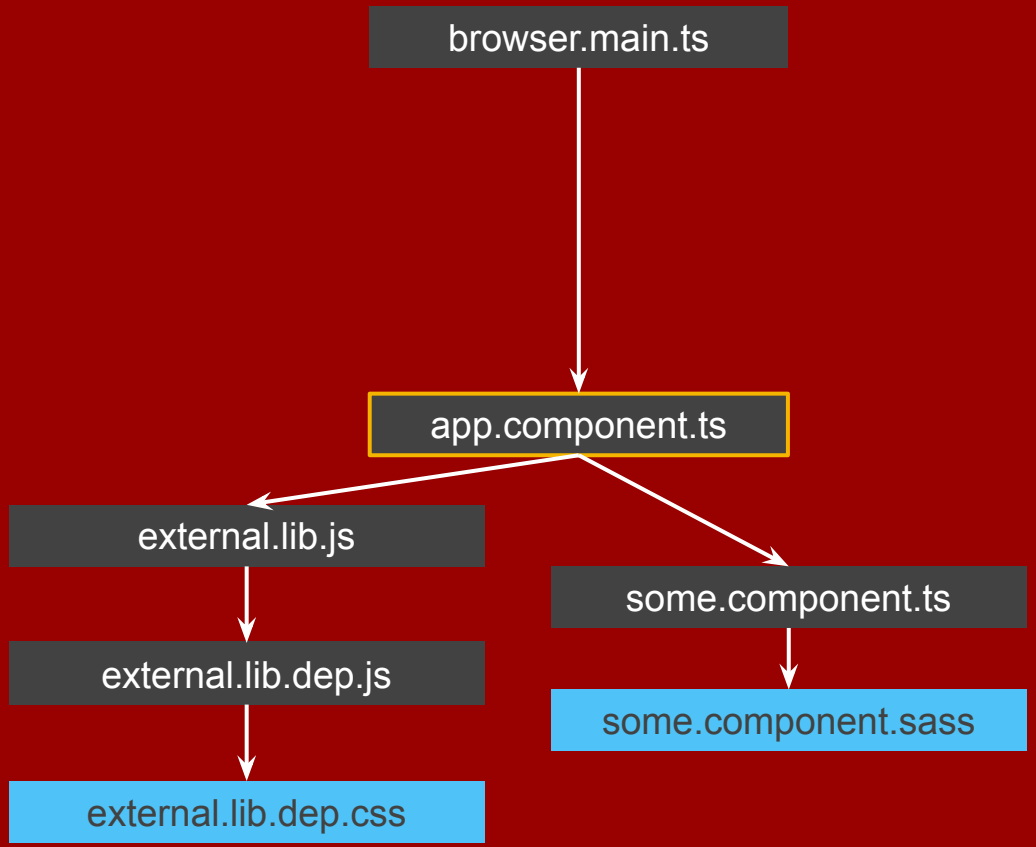
```
//webpack.config.js
module.exports = {
  entry: './browser.main.ts',
  //...
}

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

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

bootstrap(App,[]);

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



# The Core Concepts

## Entry

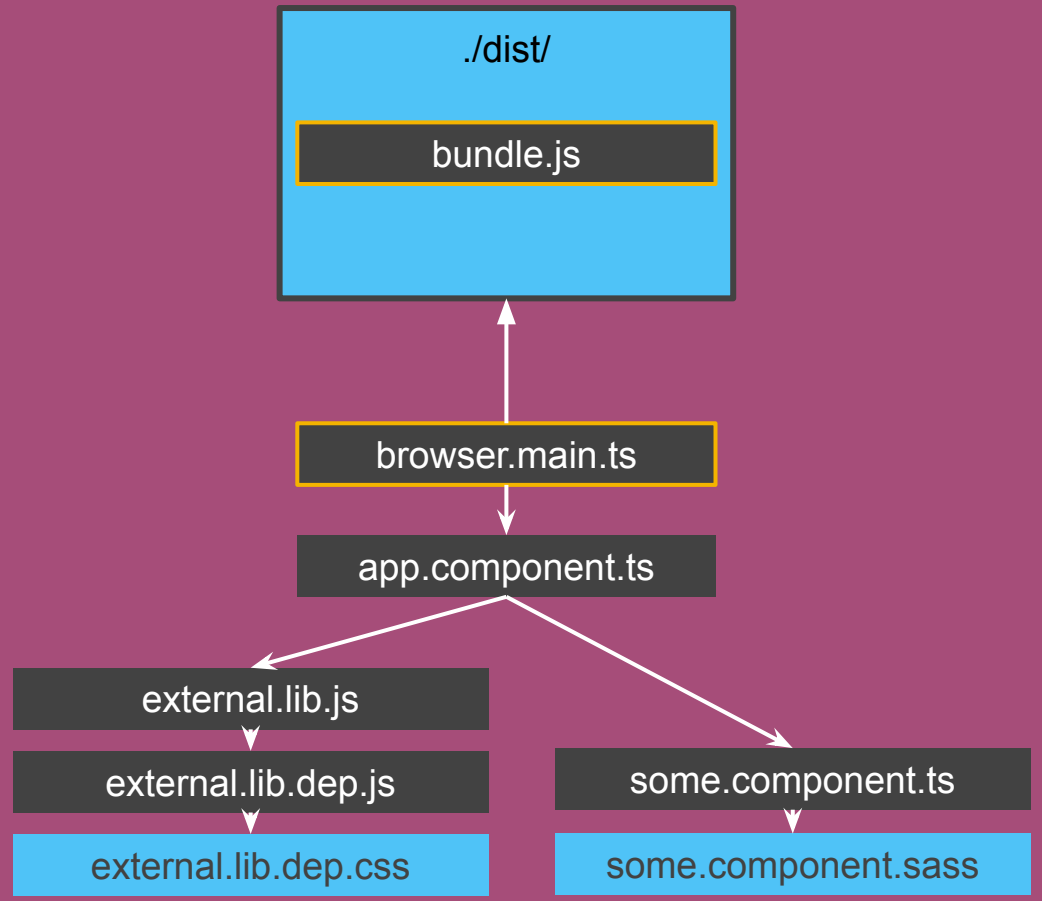
Tells webpack ***WHAT*** (files) to load for the browser;
Compliments the *Output* property.

# 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

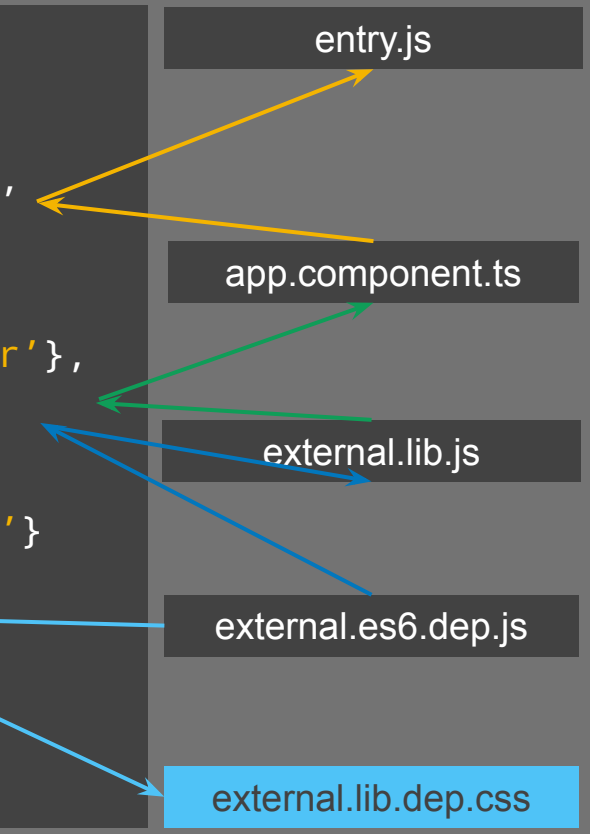
Tells webpack how to modify files before its added to dependency graph

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.*

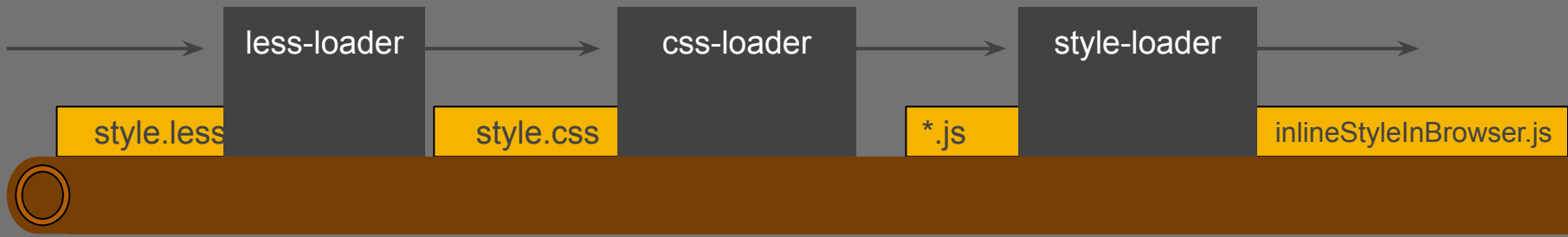
## exclude

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

# The Core Concepts: Loaders

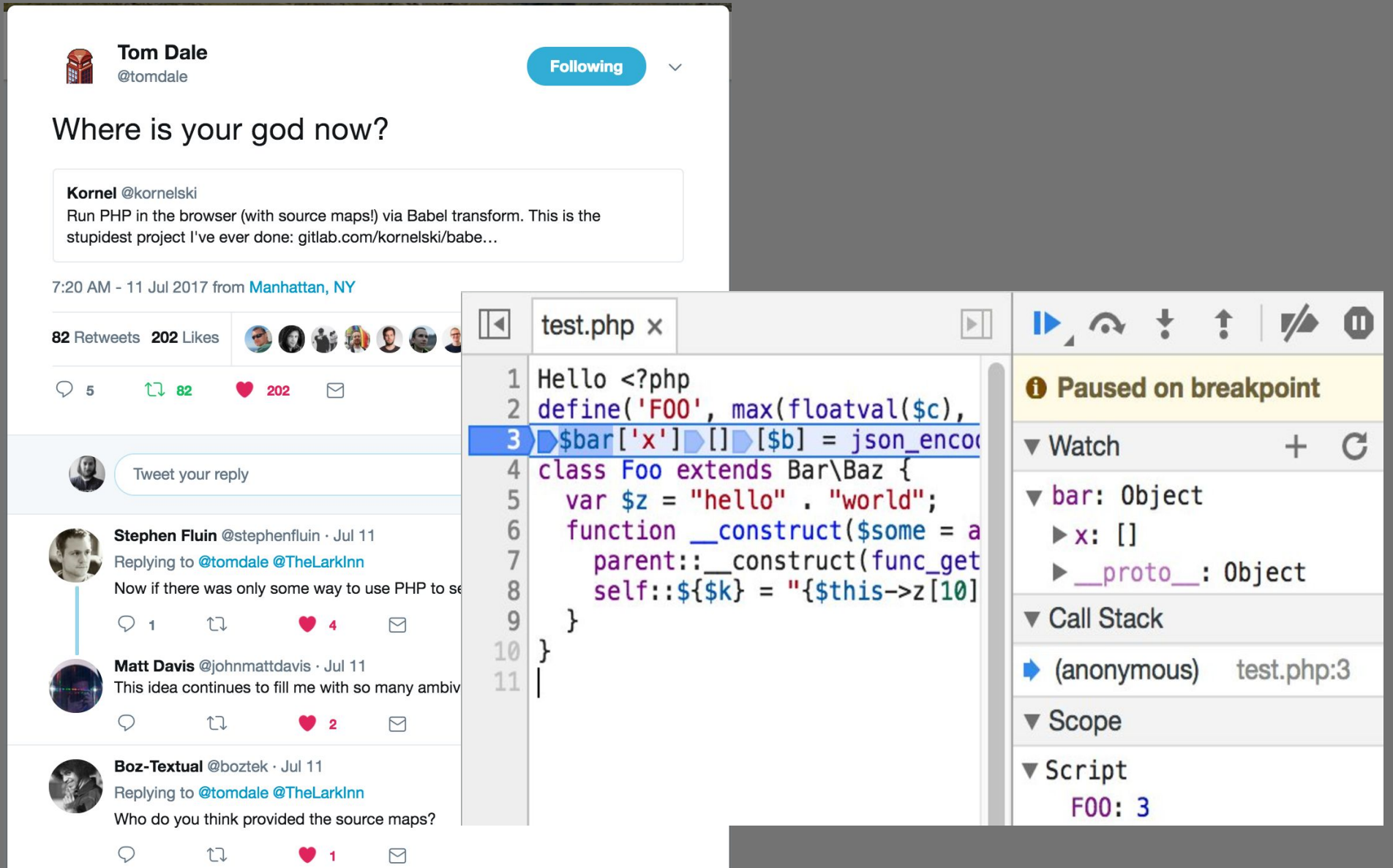
## Chaining Loaders

```
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, ngtemplate, 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…
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 4
Matt Davis @johnmattdavis · Jul 11
This idea continues to fill me with so many ambiv
2
Boz-Textual @boztek · Jul 11
Replying to @tomdale @TheLarkInn
Who do you think provided the source maps?
1
test.php
1 Hello <?php
2 define('FOO', max(floatval($c),
3 $bar['x'][][$b] = json_enco
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
Paused on breakpoint
Watch
bar: Object
x: []
__proto__: Object
Call Stack
(anonymous) test.php:3
Scope
Script
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