

# Front-end build tools

## "All you need to know"

### workshop

Webpack, Gulp, Babel, TypeScript, Sass,  
Hot Module Replacement

# Why to use build tools

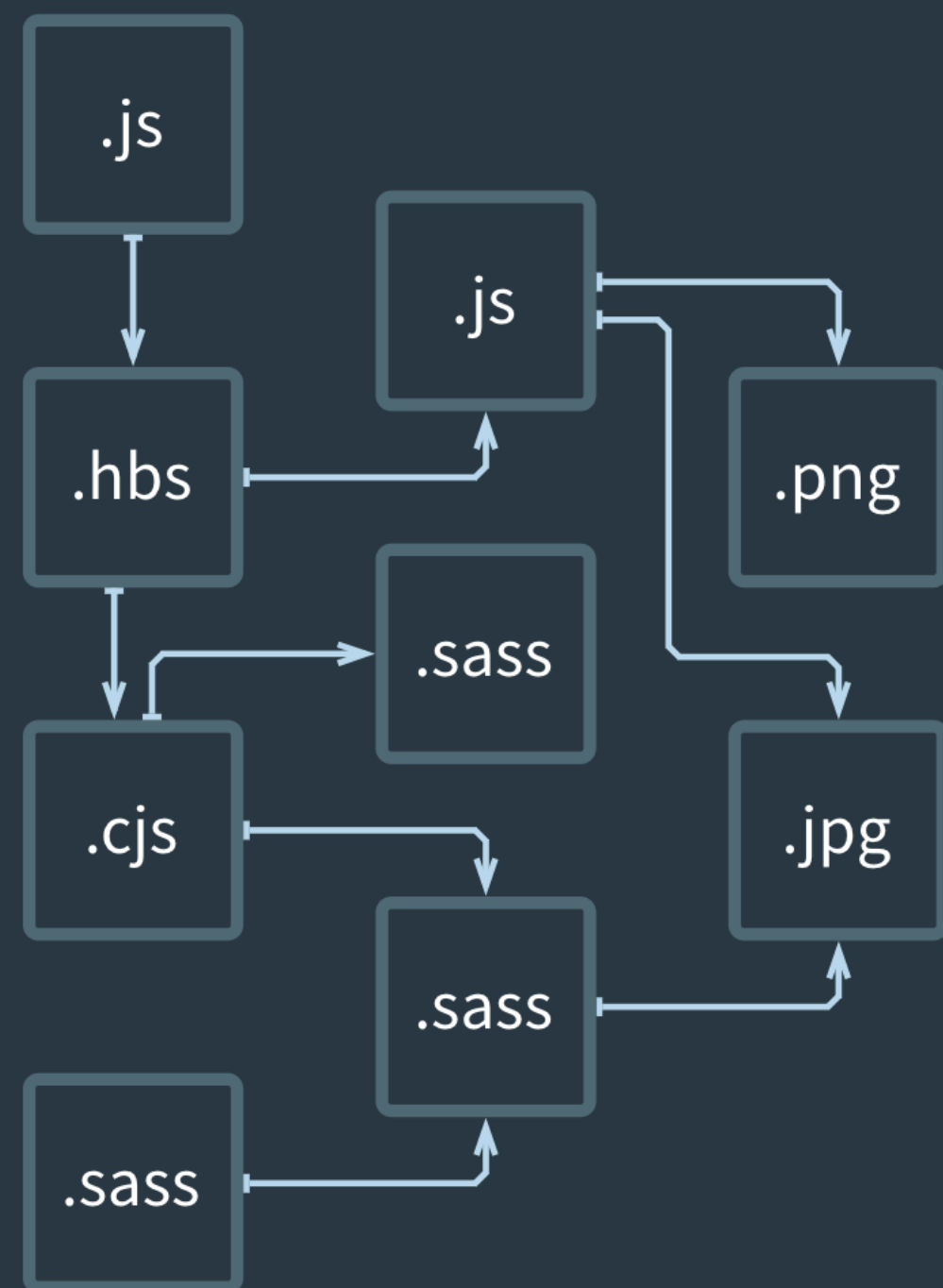
If you're building a complex Front End application with many non-code static assets such as CSS, images, fonts, etc, then yes, Webpack will give you great benefits.

If your application is fairly small, and you don't have many static assets and you only need to build one Javascript file to serve to the client, then Webpack might be more overhead than you need.

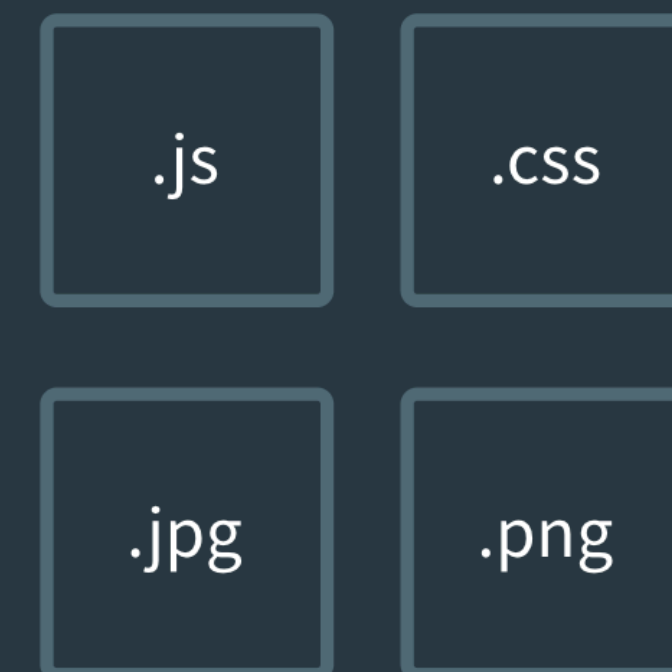
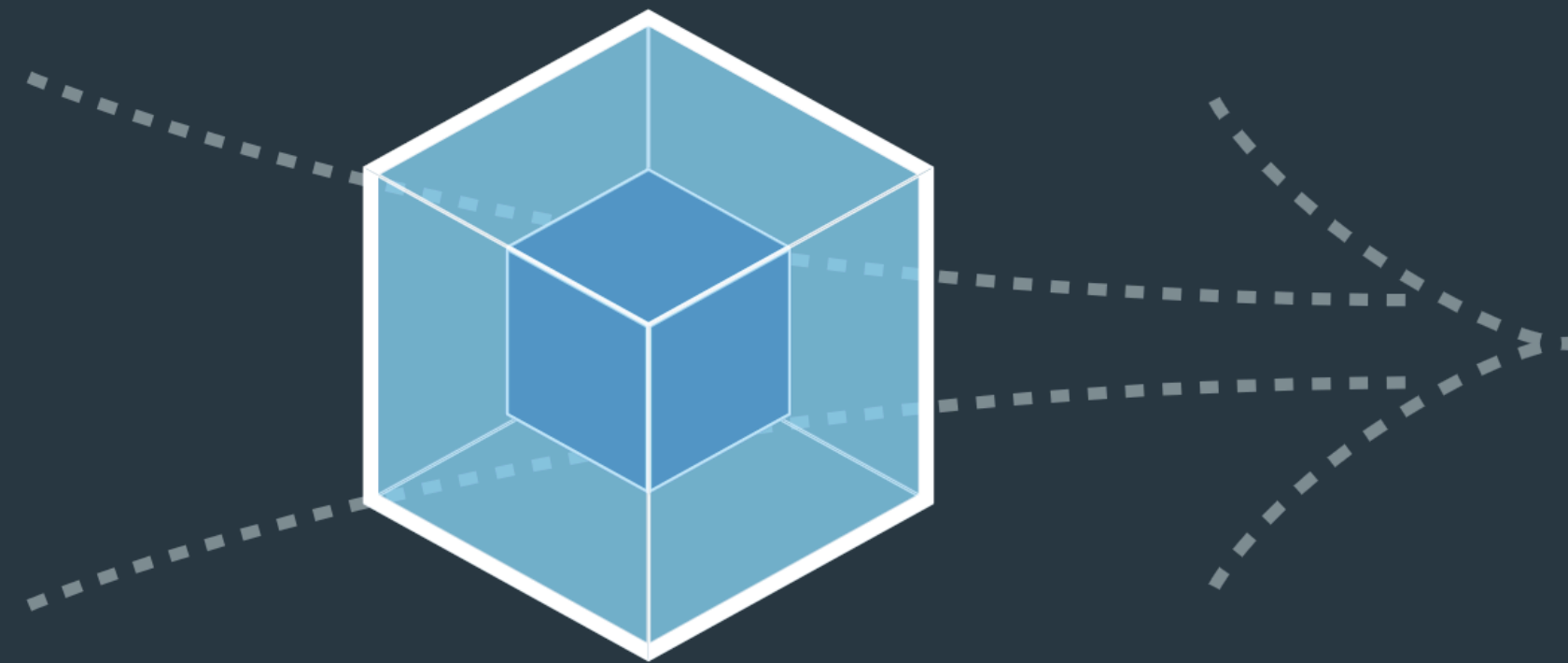
- To use ES6 modules
- To use latest Javascript features
- To use TypeScript
- To use Sass
- To use polyfills for specific target browsers
- To simplify, unify and automate build processes

- To take advantage of production build optimizations
- To speed up local development without the need to refresh the page and lose state
- Dead asset elimination
  - You only build the images and CSS into your dist folder that your application actually needs
- Stable production deploys
  - You can't accidentally deploy code with images missing, or outdated styles

# Webpack



MODULES WITH DEPENDENCIES



STATIC ASSETS

Static module builder

Webpack is a build tool that puts all of your assets, including Javascript, images, fonts, and CSS, in a dependency graph.

When webpack processes your application, it starts from a list of modules defined on the command line or in its config file.

Starting from these entry points, webpack recursively builds a dependency graph that includes every module your application needs, then bundles all of those modules into a small number of bundles - often, just one - to be loaded by the browser.

# Features

- Ability to bundle all sort of different files
- Fairly easy to set up
- Production-ready with a lot of optimizations already built-in
- Hot Module Replacement for fast development
- Lots of advanced customization features
- Mature, widely used and constantly developed

 7,331 commits

 47 branches

 324 releases

 494 contributors

 MIT



# Comparsion

Feature	webpack/webpack	jrburke/requirejs	substack/node-browserify	jspm/jspm-cli	rollup/rollup	brunch/brunch
Additional chunks are loaded on demand	yes	yes	no	<a href="#">System.import</a>	no	no
AMD <code>define</code>	yes	yes	<a href="#">deamdify</a>	yes	<a href="#">rollup-plugin-amd</a>	yes
AMD <code>require</code>	yes	yes	no	yes	no	yes
AMD <code>require</code> loads on demand	yes	with manual configuration	no	yes	no	no
CommonJS <code>exports</code>	yes	only wrapping in <code>define</code>	yes	yes	<a href="#">commonjs-plugin</a>	yes
CommonJS <code>require</code>	yes	only wrapping in <code>define</code>	yes	yes	<a href="#">commonjs-plugin</a>	yes
CommonJS <code>require.resolve</code>	yes	no	no	no	no	
Concat in <code>require</code> <code>require("./fi" + "le")</code>	yes	no♦	no	no	no	
Debugging support	<b>SourceUrl,</b> <b>SourceMaps</b>	not required	SourceMaps	<b>SourceUrl,</b> <b>SourceMaps</b>	<b>SourceUrl,</b> <b>SourceMaps</b>	SourceMaps



Feature	webpack/webpack	jrburke/requirejs	substack/node-browserify	jspm/jspm-cli	rollup/rollup	brunch/brunch
Debugging support	SourceUrl, SourceMaps	not required	SourceMaps	SourceUrl, SourceMaps	SourceUrl, SourceMaps	SourceMaps
Dependencies	19MB / 127 packages	11MB / 118 packages	1.2MB / 1 package	26MB / 131 packages	?MB / 3 packages	
ES2015 <code>import</code> / <code>export</code>	yes (webpack 2)	no	no	yes	yes	yes, via <a href="#">es6 module transpiler</a>
Expressions in require (guided) <code>require("./templates/" + template)</code>	yes (all files matching included)	no♦	no	no	no	no
Expressions in require (free) <code>require(moduleName)</code>	with manual configuration	no♦	no	no	no	
Generate a single bundle	yes	yes♦	yes	yes	yes	yes
Runtime overhead	243B + 20B per module + 4B per dependency	14.7kB + 0B per module + (3B + X) per dependency	415B + 25B per module + (6B + 2X) per dependency	5.5kB for self-executing bundles, 38kB for full loader and polyfill, 0 plain modules, 293B CJS, 139B ES2015 System.register before gzip	none for ES2015 modules (other formats may have)	
Watch mode	yes	not required	<a href="#">watchify</a>	not needed in dev	<a href="#">rollup-watch</a>	yes

# Babel

Javascript compiler / transpiler

*BABEL*

# Features

- Developers can use latest JavaScript features
- Polyfills features that are missing in target environments
- Transforms syntax
- Supports plugins for specific environments e.g React JSX
- Highly customizable target browsers

# DEVELOPMENT

Setting up Webpack build  
and Babel configuration

# TypeScript



JavaScript with types



# Features

- Adds support for type annotations
- Adds object oriented features to JavaScript
- Compile-time checking
- Interfaces, enums, generics, tuples etc...
- Decreases potential of bugs
- Increases confidence in code refactor

# TypeScript vs Flow.js

## FLOW VS. TYPESCRIPT

### Flow

- **Checker**
- Non-nullable by default
- Focused on **Soundness**
- Written in OCAML
- **Works without any annotations**
- Works out of the box with React

### TypeScript

- **Compiler**
- Nullable by default
- Focused on **Tooling & Scalability**
- Written in TypeScript
- Great IDE/Editor integration
- Used as default by more and more libraries



# DEVELOPMENT

## Setting up TypeScript compilation

# Sass



CSS with superpowers

# Features

- Enables writing CSS as modules in separate files
- Paired with BEM methodology increases clarity greatly
- Adds scripting support to CSS
- Variables, nesting, mixins, partials, SassScript etc...

# DEVELOPMENT

## Setting up Sass compilation

# Webpack Dev Server (WDS) + Hot Module Replacement (HMR)

Hot Module Replacement exchanges, adds, or removes modules while an application is running, without a full reload.

# How HMR works

The following steps allow modules to be swapped in and out of an application:

1. The application asks the HMR runtime to check for updates.
2. The runtime asynchronously downloads the updates and notifies the application.
3. The application then asks the runtime to apply the updates.
4. The runtime synchronously applies the updates.

# DEVELOPMENT

## Setting up WDS and HMR

# Gulp.js



Task runner



# Gulp task

**gulp.task(name[, deps], fn)**

```
gulp.task('js', function() {  
  gulp.src('app/js/main.js')  
    .pipe(uglify())  
    .pipe(concat())  
    .pipe(gulp.dest('build'))  
});
```

# Features

- More efficient than npm scripts for IO operations
- Possible to define custom tasks that cannot be performed otherwise

DEVELOPMENT

Setting up Gulp

# Linting



The diagram consists of three identical rectangular boxes arranged horizontally. Each box has a thin white border and contains a text label in white. The first box on the left contains the text 'ESLint', the middle box contains 'TSLint', and the third box on the right contains 'SassLint'.

ESLint

TSLint

SassLint

A linter refers to tools that analyze source code to flag programming errors, bugs, stylistic errors, and suspicious constructs.

# Features

- Agreed upon code syntax and methodologies per team/service
- Reduction in potential bugs
- Tons of rules that can be configured easily
- More readable code

# DEVELOPMENT

## Setting up Linting

# Useful resources

<https://blog.andrewray.me/webpack-when-to-use-and-why/>

<https://github.com/webpack-contrib/sass-loader>

<https://webpack.js.org/configuration/>

<https://webpack.js.org/concepts/modules/>

<https://webpack.js.org/configuration/devtool/>

<https://webpack.js.org/configuration/module/>

<https://babeljs.io/>

<https://github.com/TypeStrong/ts-loader>

<http://chir.ag/projects/name-that-color/>

<https://github.com/webpack-contrib/css-loader>

<https://github.com/webpack-contrib/css-loader>

<https://webpack.js.org/concepts/hot-module-replacement/>

<https://github.com/webpack-contrib/file-loader>

<https://github.com/jantimon/html-webpack-plugin>

<https://github.com/johnnagan/clean-webpack-plugin>

<https://www.npmjs.com/package/url-loader>

<https://learn.co/lessons/javascript-lodash-templates>

<https://eslint.org/>

# Useful resources

<https://eslint.org/docs/user-guide/configuring>

<https://www.npmjs.com/package/eslint-plugin-react>

<https://www.npmjs.com/package/eslint-plugin-import>

<https://github.com/babel/eslint-plugin-babel>

<https://palantir.github.io/tslint/>

<https://www.npmjs.com/package/sass-lint>

<https://webpack.js.org/comparison/>

<https://engineering.velocityapp.com/webpack-vs-browsersify-vs-systemjs-for-spas-95b349a41fa0>

<https://en.wikipedia.org/wiki/TypeScript>

<https://en.wikipedia.org/wiki/Webpack>

[https://en.wikipedia.org/wiki/Babel\\_\(compiler\)](https://en.wikipedia.org/wiki/Babel_(compiler))

<https://github.com/niileani/typescript-vs-flowtype>

<https://sass-lang.com/>

[https://en.wikipedia.org/wiki/Sass\\_\(stylesheet\\_language\)](https://en.wikipedia.org/wiki/Sass_(stylesheet_language))

<http://getbem.com/>

<https://en.wikipedia.org/wiki/Gulp.js>

[https://en.wikipedia.org/wiki/Lint\\_\(software\)](https://en.wikipedia.org/wiki/Lint_(software))



Thank you!