# Front-end build tools "All you need to know" workshop

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

The goal of this workshop is not to compare different build tools but rather to set up a working front end build environment using the most common tools and giving practical information to apply in your own application.

I have set up a very basic React app. Now we need to make it compile so that browsers can understand and display it.
I hope you all have installed Node and downloaded the repo. Please run npm install if you have not done so yet.

[introducing repo files]
[talking about workshop steps]

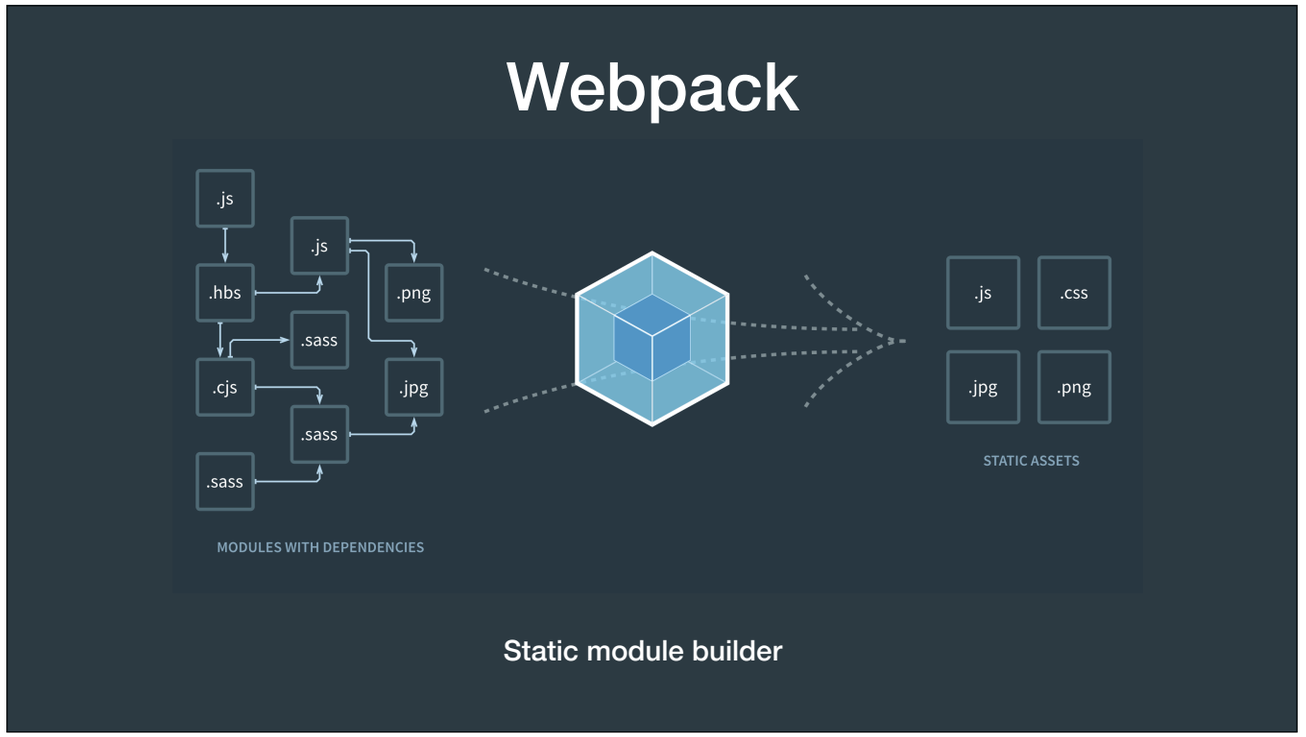
# Why to use build tools

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  - You can't accidentally deploy code with images missing, or outdated styles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 (stylised webpack) is an open-source JavaScript module bundler. Its main purpose is to bundle JavaScript files for usage in a browser, yet it is also capable of transforming, bundling, or packaging just about any resource or asset. Webpack takes modules with dependencies and generates static assets representing those modules.
Through "loaders," modules can be CommonJs, AMD, ES6 modules, CSS, Images, JSON, Coffeescript, LESS, ... and your custom stuff.

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.

Any time one file depends on another, webpack treats this as a dependency. This allows webpack to take non-code assets, such as images or web fonts, and also provide them as dependencies for your application.

# 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

# 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 | System.import | no | no |
| AMD `define` | **yes** | **yes** | deamdify | yes | rollup-plugin-amd | yes |
| AMD `require` | **yes** | **yes** | no | yes | no | yes |
| AMD `require` loads on demand | **yes** | with manual configuration | no | yes | no | no |
| CommonJS `exports` | **yes** | only wrapping in `define` | **yes** | yes | commonjs-plugin | yes |
| CommonJS `require` | **yes** | only wrapping in `define` | **yes** | yes | commonjs-plugin | yes |
| CommonJS `require.resolve` | **yes** | no | no | no | no | |
| Concat in require `require("./fi" + "le")` | **yes** | no♦ | no | no | no | |
| Debugging support | **SourceUrl, SourceMaps** | not required | SourceMaps | **SourceUrl, SourceMaps** | **SourceUrl, SourceMaps** | SourceMaps |

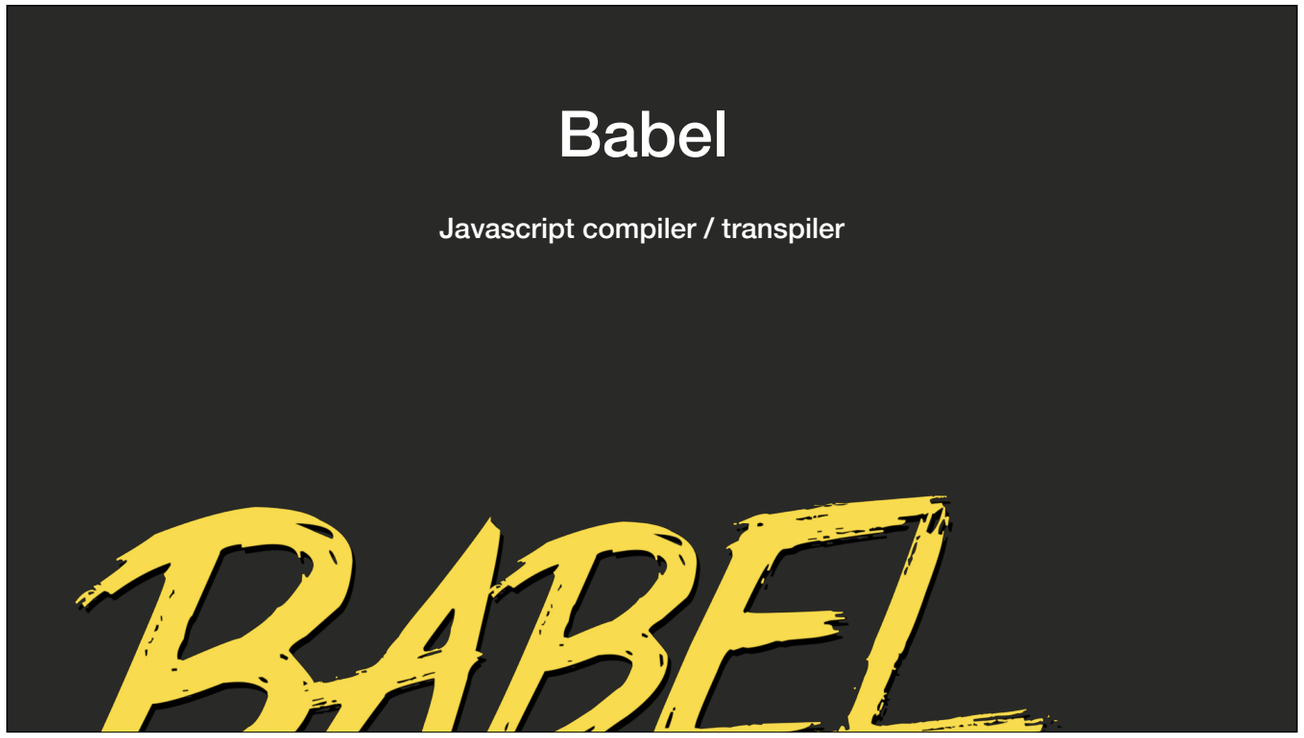
Webpack is the most feature rich build tool and usually most suitable for all sizes of applications that need to be production ready and configurable.

| 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 `import` / `export` | **yes** (webpack 2) | no | no | **yes** | **yes** | yes, via es6 module transpiler |
| Expressions in require (guided) `require("./templates/" + template)` | **yes (all files matching included)** | no♦ | no | no | no | no |
| Expressions in require (free) `require(moduleName)` | 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 | watchify | not needed in dev | rollup-watch | yes |

As they write on their website, tt's important to note some key differences between loading and bundling modules. A tool like SystemJS, which can be found under the hood of JSPM, is used to load and transpile modules at runtime in the browser. This differs significantly from webpack, where modules are transpiled (through "loaders") and bundled before hitting the browser.

# Babel

Javascript compiler / transpiler




Babel or Babel.js is a free and open-source JavaScript compiler and configurable transpiler used in web development.
Its a toolchain that is mainly used to convert ECMAScript 2015+ code into a backwards compatible version of JavaScript in current and older browsers or environments.

# 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

Babel plugins are available to provide specific conversions used in web development. For example, developers working with React.js, can use Babel to convert JSX (JavaScript eXtension) markup into JavaScript using the Babel preset “react".

DEVELOPMENT

# Setting up Webpack build and Babel configuration

# TypeScript


JavaScript with types

TypeScript is an open-source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript, and adds optional static typing to the language.

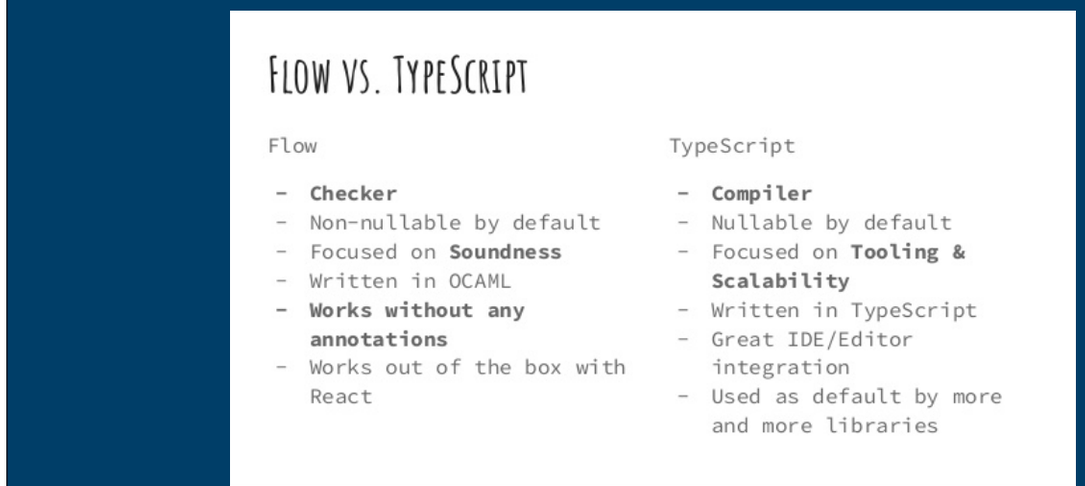
TypeScript is designed for development of large applications and transcompiles to JavaScript. As TypeScript is a superset of JavaScript, existing JavaScript programs are also valid TypeScript programs. TypeScript may be used to develop JavaScript applications for both client-side and server-side (Node.js) execution.

# 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

Types enable JavaScript developers to use highly-productive development tools and practices like static checking and code refactoring when developing JavaScript applications.
TypeScript offers support for the latest and evolving JavaScript features, including those from ECMAScript 2015 and future proposals, like async functions and decorators, to help build robust components.

# TypeScript vs Flow.js




I would recommend using Typescript since it is much more powerful and opens up a new world of possibilites.

DEVELOPMENT

# Setting up TypeScript compilation

# Sass


CSS with superpowers

Sass is the most mature, stable, and powerful professional grade CSS extension language in the world.

# 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…

Sass is a preprocessor scripting language that is interpreted or compiled into Cascading Style Sheets (CSS). SassScript is the scripting language itself.

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.

This can significantly speed up development in a few ways:
Retain application state which is lost during a full reload.
Save valuable development time by only updating what's changed.
Modifications made to CSS/JS in the source code results in an instant browser update which is almost comparable to changing styles directly in the browser's dev tools.

# 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

It is a task runner built on Node.js and npm, used for automation of time-consuming and repetitive tasks involved in web development like minification, concatenation, cache busting, unit testing, linting, optimization, etc

gulp uses a code-over-configuration approach to define its tasks and relies on its small, single-purposed plugins to carry them out. The gulp ecosystem includes more than 300 such plugins.

# 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'))
});
```

Gulp task uses pipes to pass data along and can be used to perform any process using its plugins.

# Features

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

Task-runners like gulp and Grunt are built on Node.js rather than npm, because the basic npm scripts are inefficient when executing multiple tasks. Even though some developers prefer npm scripts because they can be simple and easy to implement, there are numerous ways where gulp and Grunt seem to have an advantage over each other and the default provided scripts.

Not so much used for compiling and transpiling anymore since Webpack can do most of it on its own.

DEVELOPMENT

# Setting up Gulp

# Linting

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/johnagan/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://github.com/niieani/typescript-vs-flowtype
https://sass-lang.com/
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)

Thank you!