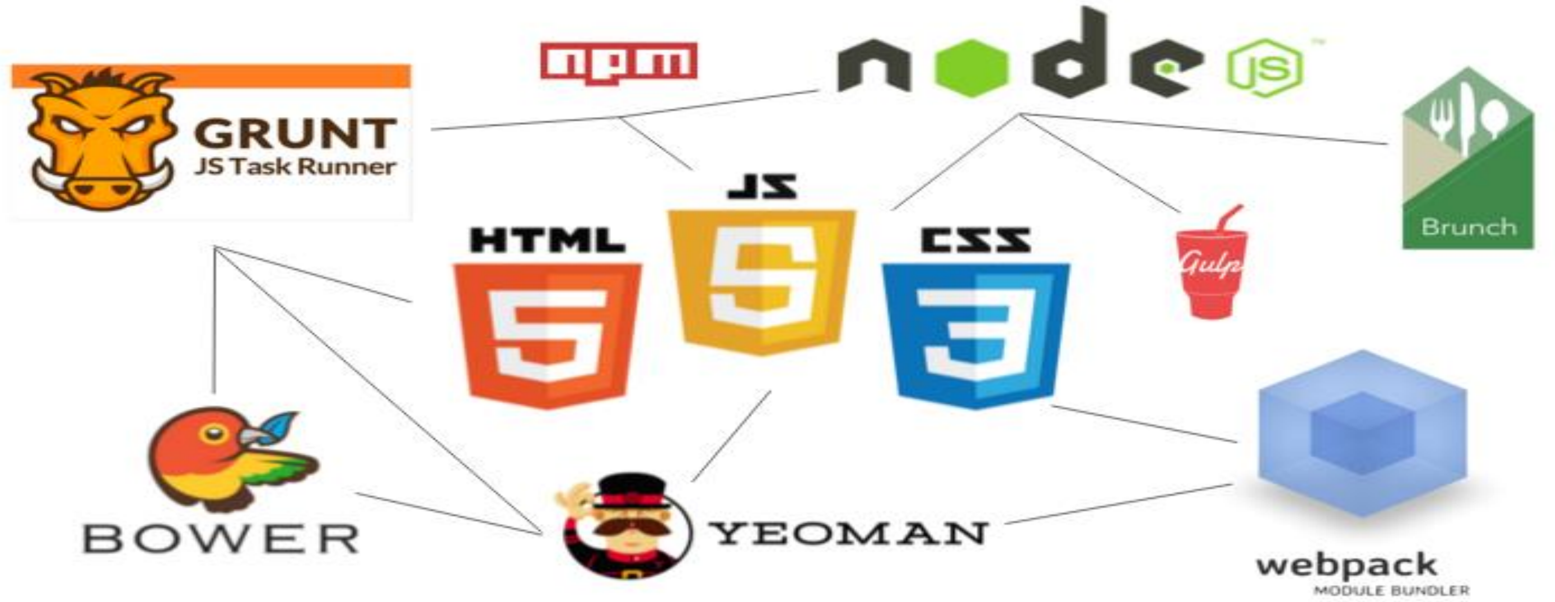
Javascript
Build Tools

@ppgowda4



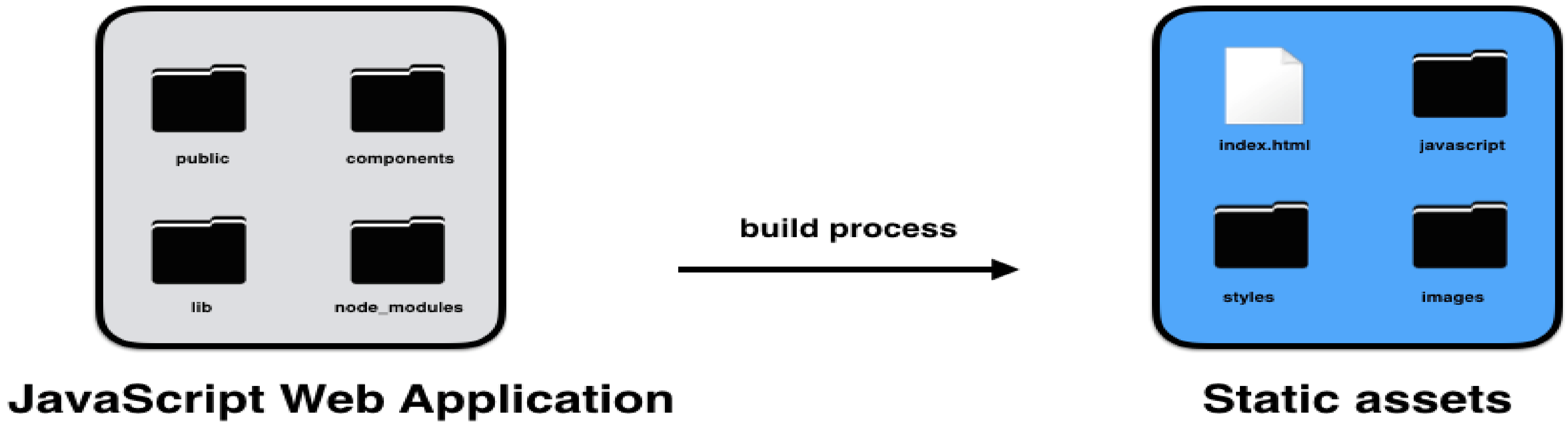
We will talk about

- What is build process?
- Conceptual understanding of each step
- Making sense of build tools
- Most common build tools
- What happens in popular frameworks

What is build process?

The rise of modern development practices has brought significant improvements to speed, security, and scalability. It has also made building websites somewhat more complex.

Promise of build process (tools) is stripping away the complexity involved in optimizing your site’s size, speed, and streamlining the management of 3rd party dependencies.



Conceptual understanding of each step

- Minification
- Asset optimization
- Pre-processing(CSS)
- Linting
- Duplication
- Tree shaking
- Transpilers(TS)
- Compression
- Polyfills
- Documentation

It’s opinionated approach to making sense of build tools

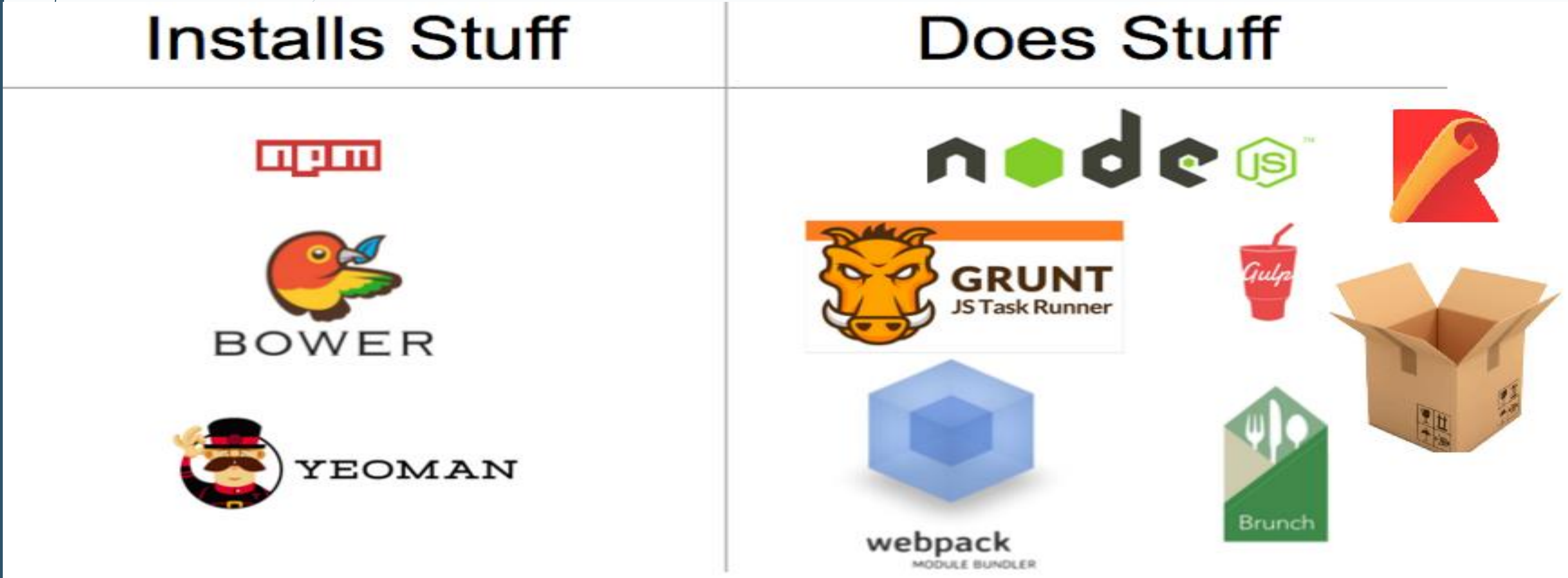
List of Javascript tooling available

- Node
- Yarn
- Yeoman
- Gulp
- Webpack
- Parcel
- Babel
- NPM
- Bower
- Grunt
- Browserify
- Brunch
- Rollup
- Microbundle

There are so many front-end build tools out there that it can seem impossible to keep up.

Making sense of build tools

The core dichotomy of build tools is “installing vs. doing”

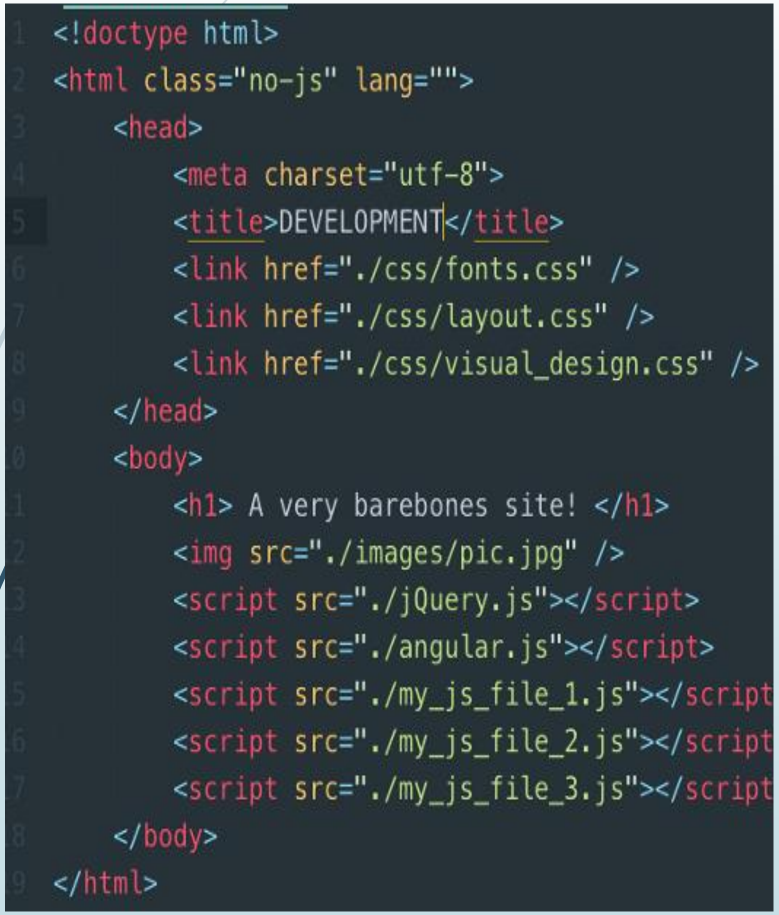


Some examples of ‘Does Stuff’ are

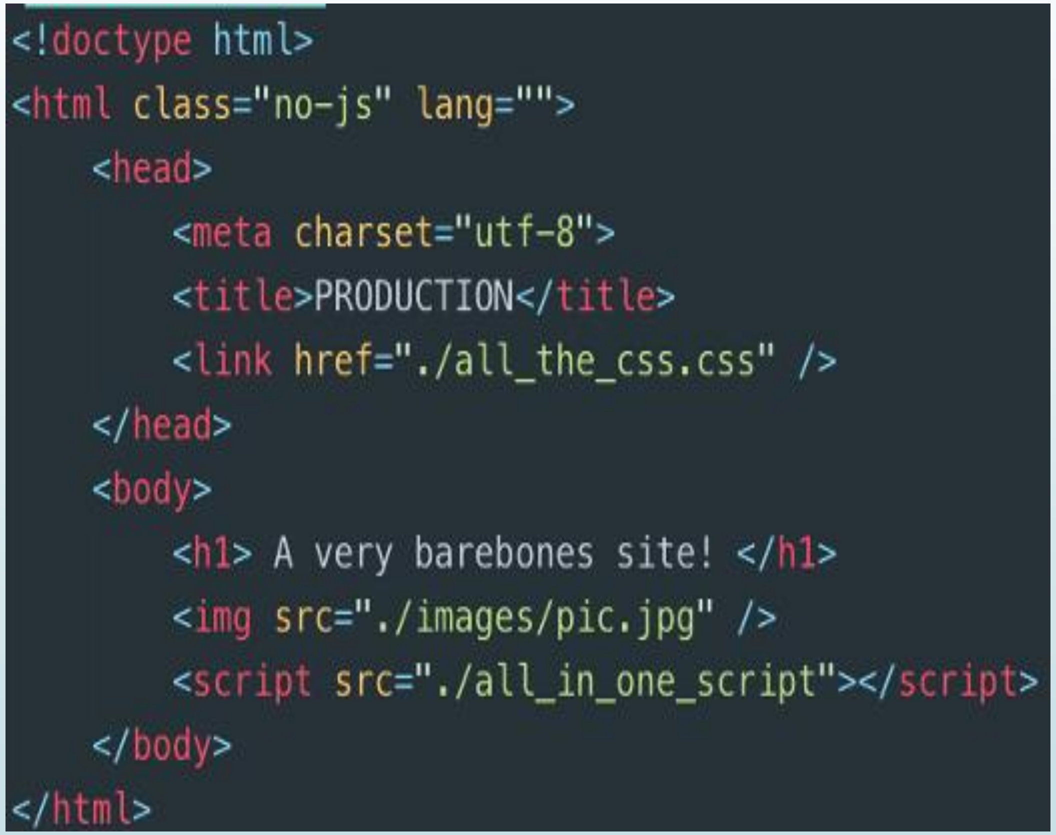
- Replacing a string of text in a file
- Creating folders and moving files into those folders
- Running my unit tests with a single command
- Refreshing my browser when I save a file
- Combining all my JavaScript files into one, and all my CSS files into one
- Minifying my concatenated JavaScript and CSS files
- Modifying the placement of <script> tags on an html page

All build tools are powered by **Node** and **NPM**

A build is just a production ready version of your app



```
<!doctype html>
<html class="no-js" lang="">
    <head>
        <meta charset="utf-8">
        <title>DEVELOPMENT</title>
        <link href="./css/fonts.css" />
        <link href="./css/layout.css" />
        <link href="./css/visual_design.css" />
    </head>
    <body>
        <h1> A very barebones site! </h1>
        <img src="./images/pic.jpg" />
        <script src="./jQuery.js"></script>
        <script src="./angular.js"></script>
        <script src="./my_js_file_1.js"></script
        <script src="./my_js_file_2.js"></script
        <script src="./my_js_file_3.js"></script
    </body>
</html>
```

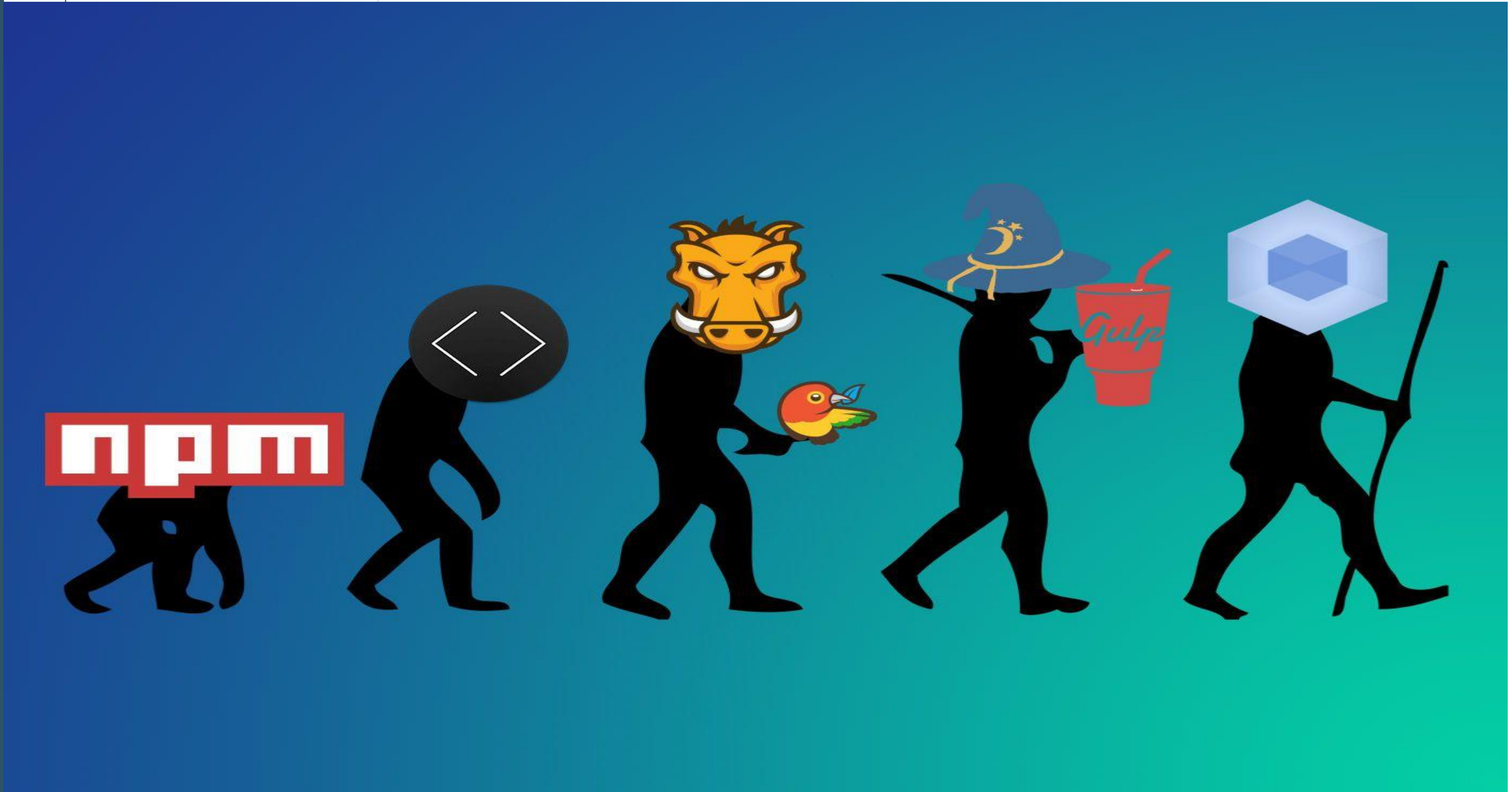


```
<!doctype html>
<html class="no-js" lang="">
    <head>
        <meta charset="utf-8">
        <title>PRODUCTION</title>
        <link href="./all_the_css.css" />
    </head>
    <body>
        <h1> A very barebones site! </h1>
        <img src="./images/pic.jpg" />
        <script src="./all_in_one_script"></script>
    </body>
</html>
```

Some more points..

- The lines between “install” and “do” can be blurry
- There is no one right combination of tools
- Build tools have a steep learning curve, so only learn what’s necessary
- All build tools share the same goal: to make Dev’s life simple by automating a lot of menial tasks
- It’s not just you. The documentation often is terrible.

Evolution of build tools

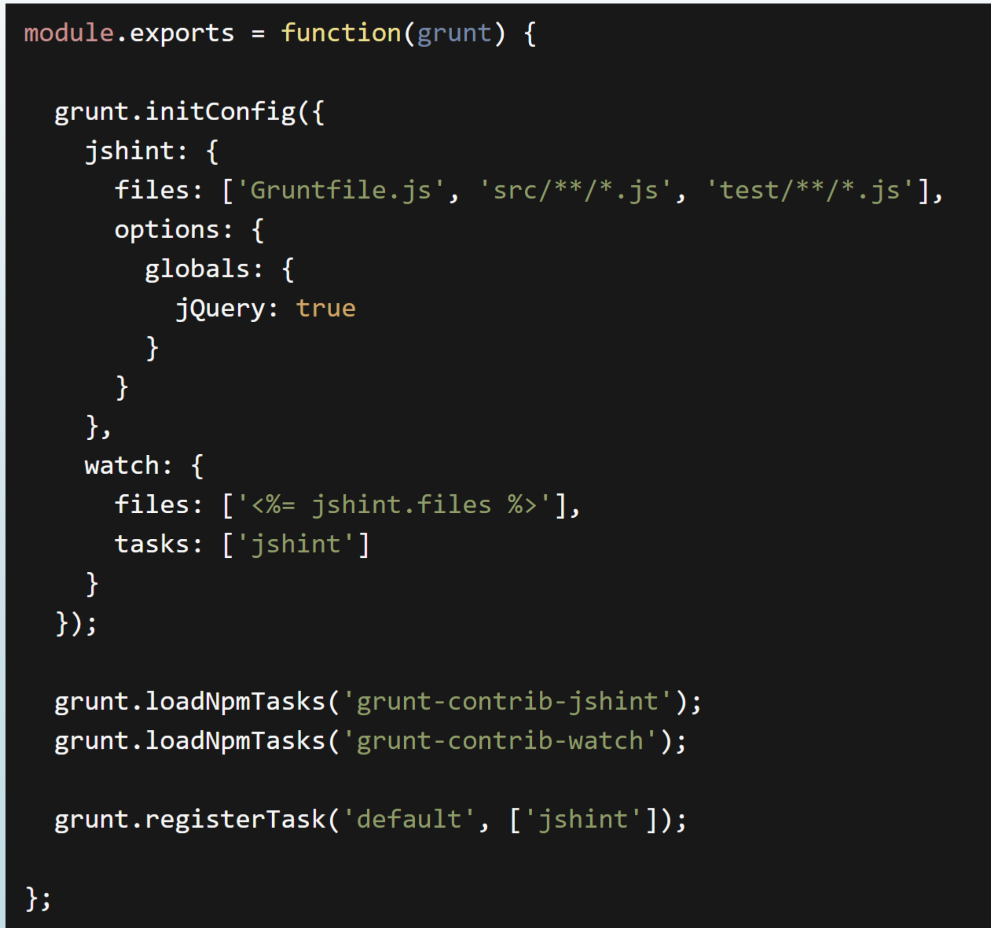


Most common build tools

GRUNT

Grunt is a JavaScript-based command line build tool that allows developers automate all of those boring repeated processes.

- Performing repetitive tasks like minification, compilation, unit testing, linting, etc.
- Configuration
- Great community

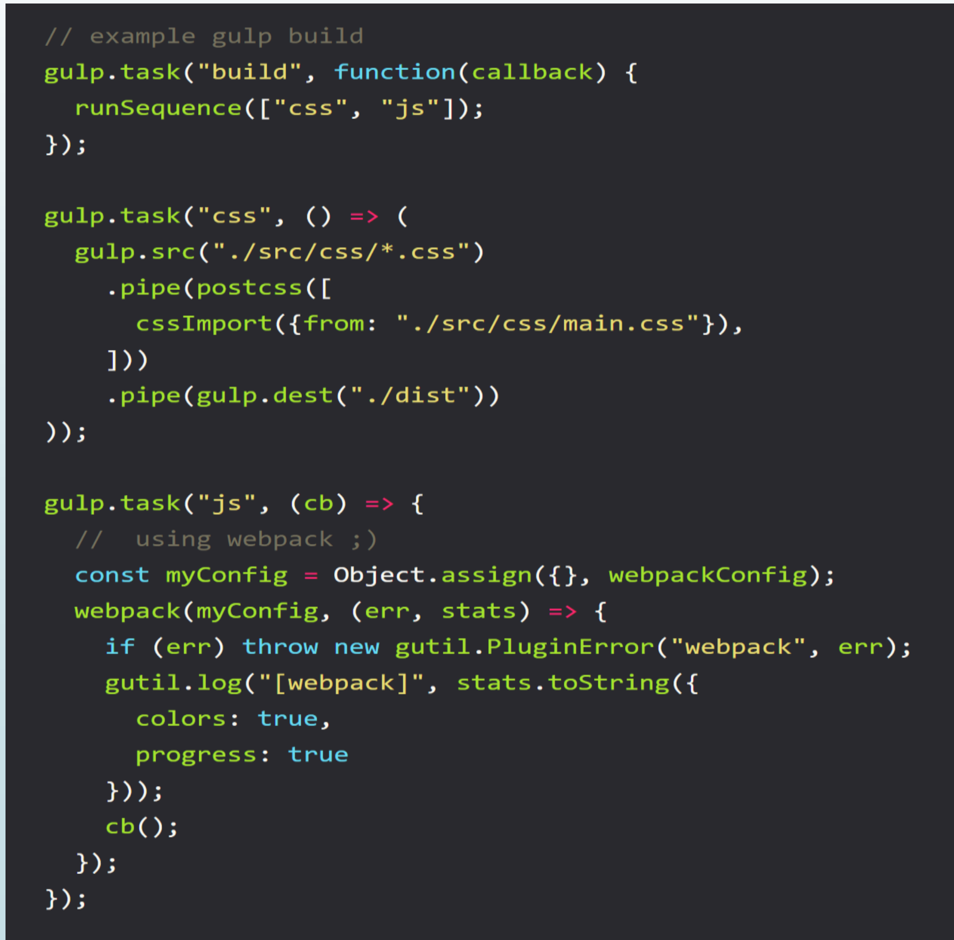


```
module.exports = function(grunt) {

  grunt.initConfig({
    jshint: {
      files: ['Gruntfile.js', 'src/**/*.js', 'test/**/*.js'],
      options: {
        globals: {
          jQuery: true
        }
      }
    },
    watch: {
      files: ['<%= jshint.files %>'],
      tasks: ['jshint']
    }
  });

  grunt.loadNpmTasks('grunt-contrib-jshint');
  grunt.loadNpmTasks('grunt-contrib-watch');

  grunt.registerTask('default', ['jshint']);

};
```

Gulp

Gulp defines tasks as JavaScript functions instead of configuration objects and runs time-consuming tasks

- Code-over-configuration
- Build speed & Readable
- Works great with Browserify
- Lots of plugins
- Node streams



```
// example gulp build
gulp.task("build", function(callback) {
  runSequence(["css", "js"]);
});

gulp.task("css", () => (
  gulp.src("./src/css/*.css")
    .pipe(postcss([
      cssImport({from: "./src/css/main.css"}),
    ]))
    .pipe(gulp.dest("./dist"))
));

gulp.task("js", (cb) => {
  //  using webpack ;)
  const myConfig = Object.assign({}, webpackConfig);
  webpack(myConfig, (err, stats) => {
    if (err) throw new gutil.PluginError("webpack", err);
    gutil.log("[webpack]", stats.toString({
      colors: true,
      progress: true
    }));
    cb();
  });
});
```

webpack

Webpack is a build tool that is built on 4 main concepts: Entry, Output, Plugins, and Loaders.

- Most powerful bundler
- Flexible configuration
- Code splitting
- Lots of plugins
- Built-in dev server with live-reload
- Can handle all types of assets

```
// example of a webpack.config.js

module.exports = {
  entry:'./index.js',
  output: {
    filename: 'bundle.js'
  }
}
```

rollup.js

Rollup provides much simpler configuration over webpack and has a host of pre-configured plugins that are a breeze to incorporate into your project

- It uses the new standardized format for code modules included in the ES6 revision of JavaScript Flexible configuration
- Tree shaking
- Compatibility with CommonJS modules.

```
import babel from 'rollup-plugin-babel';

export default {
    input: './src/main.js',
    output: {
        file: './build/bundle.min.js',
        format: 'iife',
        name: 'bundle'
    },
    plugins: [
        babel({
            exclude: 'node_modules/**'
        })
    ]
}
```

PARCEL

Parcel is a web application bundler, differentiated by its developer experience. It offers blazing fast performance utilizing multicore processing, and requires zero configuration.

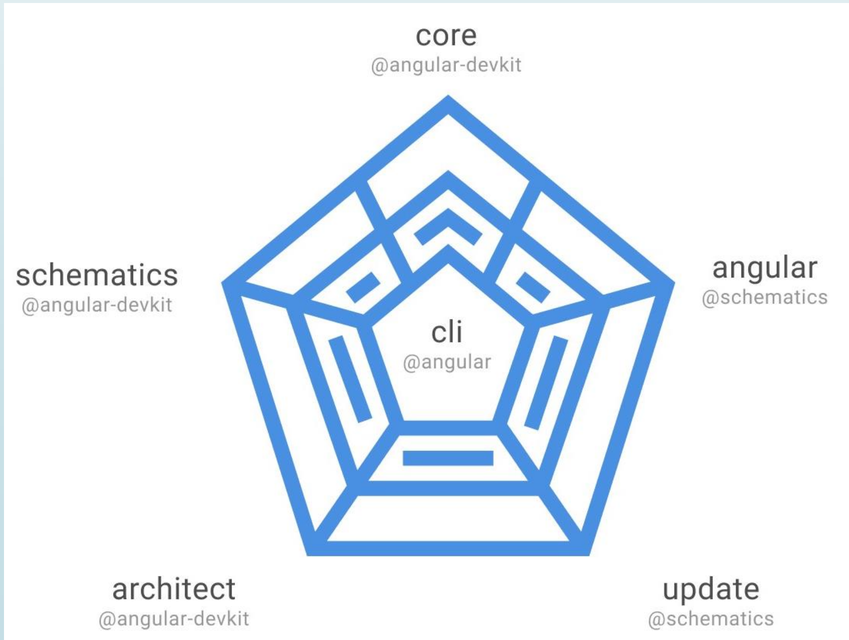
- Blazing fast performance
- Zero configuration
- Tree Shaking for both ES6 and CommonJS modules
- Up to 2x faster file watcher
- Resolved filenames are now cached

What happens in popular frameworks

The **Angular** CLI abstracts a lot of things, from the way you create a new Angular project to generating production build. It setups all the necessary configurations for each tool used for bundling, linting, testing. It allows you to generate code, extracts your i18n strings, handles update and migration.

Angular CLI has 5 concepts in project setup

- workspace
- Schematics
- Architects
- Builders
- angular.json



Thank you

Any Questions?