JavaScript Tooling

Advanced JavaScript

1

tl;dr

- Modern JavaScript development requires a ton of libraries which in turn requires the use of a lot of tools:
  - npm - organizes libraries and automates the build
  - linters - reports bad code practices
  - unit testers - run automated test scripts
  - webpack - minifies, bundles, and automates the build
  - babel or typescript - transpiles to allow all code to run universally

4

Web development in the 90's

- 90's JavaScript development looked like this:

1. Write a series of HTML pages
2. Write a JavaScript file to handle DOM manipulation
3. All data reads/writes would require a request to the server for a new page

6

Web development in the 2000's

- jQuery!
- Write a series of HTML pages
- Download jQuery.js and put in the root of your site.

```
<script src="jquery.js"></script>
```

- Write a JavaScript file to handle DOM manipulation through jQuery
- Make occasional Ajax requests with jQuery

7

Web development in the 2010s

- backbone, ember, angularJS!
- Write a series of HTML pages
- Download your library

```
<script src="angular.js"></script>
```

- Let angularJS handle your DOM manipulation and Ajax.
- Write a JavaScript file to deal with Model and Controller
- We occasionally have a SPA

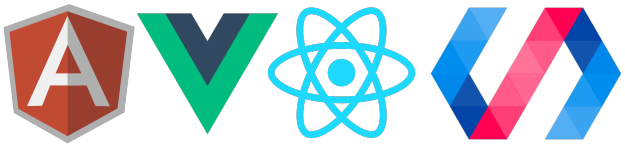
8

In come the libraries!

- Tools - Datejs, Moment, Sylvester, Dojo, Moo
- Graphing libraries - jsCharts, D3, Raphael
- Animation - $fx, jsTweener
- Asynchronous JS - q.js, promises
- Component libraries - jQueryUI, et. al.
- Browser detection and polyfills- Modernizr
- Forms - wForms, qForms, formReform
- Layout - Bootstrap, grid360

9

Modern web development involves frameworks that require a lot of care



These are nearly impossible to develop without tooling.

10

- Modern JavaScript development involves libraries
- Tons!
- Tons of libraries means that have to be pre-processed
- This means either extra work for you or
- Lots of tooling!
- This chapter is about that tooling

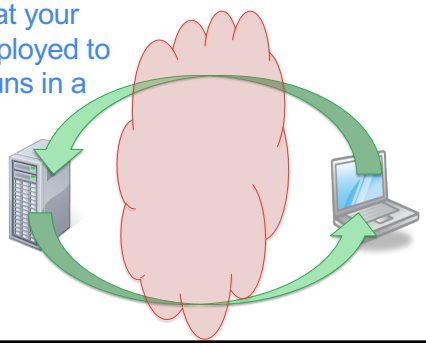
11

Problems	Solutions!
We use tons of libraries and the versions are important!	Package managers
We should unit test everything	Testing suites ... • Test runners • Testing frameworks • Assertion libraries
Too many fetches from the server	Bundlers
Too many bytes sent by the server	Minifiers
We want to use modern features in our code that the browsers don't support yet	Transpilers
It's time-consuming and error-prone to run all of these tools manually	Task runners

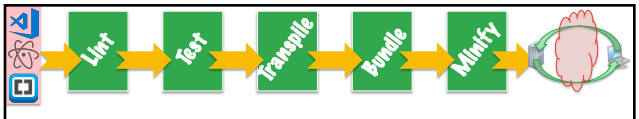
13

Remember that your web app is deployed to a server but runs in a browser

Where do all these tools fit in?



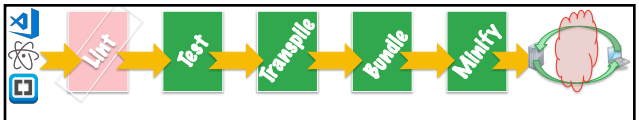
14



You'll write your code in an IDE

- VS Code
- Atom
- Brackets
- WebStorm
- Sublime
- et. al.

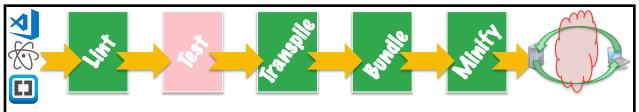
15



Lint

- Tells you when your code is not following best practices.
- Helps to maintain high code quality.
- ESLint - Newer and better
- JSHint - Rules are customizable
- JSLint - The original. Crockford's rules are built-in.

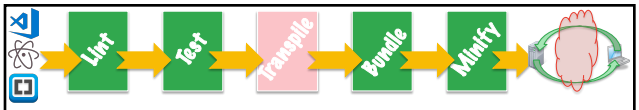
16



Test

- Automated tests are made of unit tests and integration tests
- If we need a browser, it's called a UI test or e2e test
- Much more about this step later!

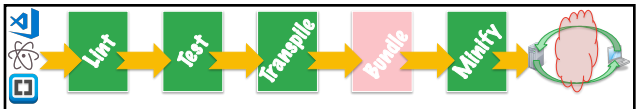
17



Transpile

- Convert one dialect of JavaScript to another
- Babel - ES2018 to ES5
- TypeScript - Proprietary dialect to ES5
- Traceur - ES2015-to-ES5
- CoffeeScript - Proprietary dialect to ES5

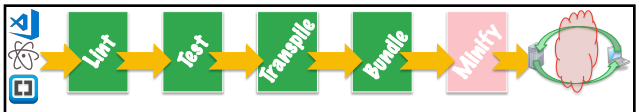
18



Bundle

- Normally we'd list a boatload of <script> tags in the <head>
- Each of those fires off an http GET request.
- And they can block the UI if not handled right
- HTTP requests are expensive to manage
  - Create a socket
  - Handshakes
  - Reserve memory
  - Clean up file handles
- Bundling puts all JavaScript files into one great big one.

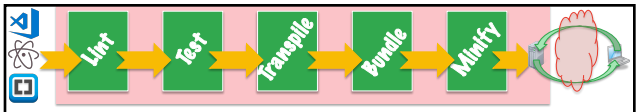
19



Minify

- Makes your JS smaller by removing every character it doesn't absolutely need.
- webpack - Very capable and does more for you.
- Uglify - Once was very popular
- Browserify - Conceptually easier

20



Automating the build

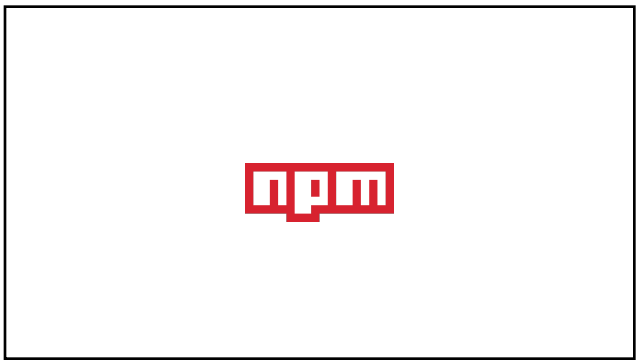
- This is a lot of steps! Thank goodness we don't have to do it manually!
- gulp - Streams - code over configuration
- grunt - Files - configuration over code
- npm run

21

Package managers

- Download and install components in a controlled, easy way
- Like system package managers but for JavaScript
  - bower - Good for static assets. Has been called "abandonware"
  - npm - Most popular
  - yarn - More efficient

22



31

npm does several things

1. Holds project metadata
2. Manages packages (aka libraries)
3. Runs tasks

33

```
{
  "name": "demo",
  "version": "10.2.4",
  "private": true,
  "dependencies": {
  },
  "scripts": {
  },
  "devDependencies": {
  }
}
```

1. Holds project metadata

npm uses package.json as its config file

34

npm as a package manager

35

Certain libraries were written to be installed globally.	Others were written to be local to a folder	Still others can be done either way
• npm itself • npx • phonegap • gulp • grunt	• react • angular • vue • redux • jquery	• @angular/cli • tsc • jasmine • mocha

36

```
{
"dependencies": {
    "bootstrap": "^4.1.2",
    "react": "^16.4.1",
    "react-dom": "^16.4.1",
    "react-scripts": "1.1.4"
  },
"devDependencies": {
    "eslint": "^5.1.0",
    "react-test": "^16.4.1"
  }
}
```

2. npm manages libraries

39

npm installs libraries in a controlled way

`npm install`

- Reads package.json and installs all the packages in the dependencies and devDependencies

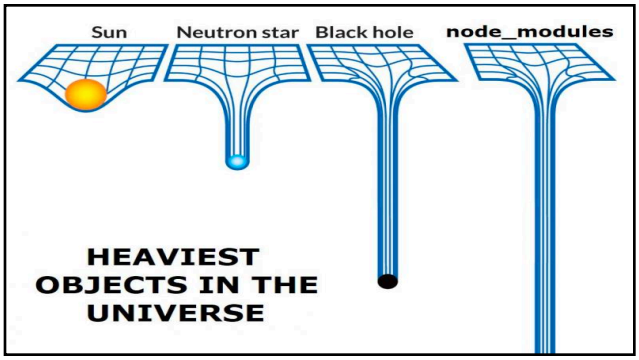
`npm install packageName`

- Adds the package to the dependencies section
- For things to be sent to the browser

`npm install --save-dev packageName`

- Adds the package to the devDependencies section
- For build/compile/development tools

43



49

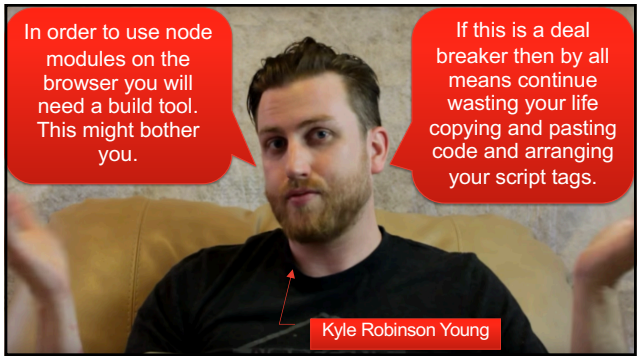
npm as task runner

50

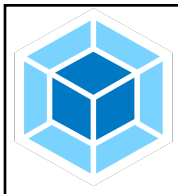
```
"scripts": {
 "test": "jest",
 "e2e": "karma start",
 "build": "eslint && webpack
           && npm run test",
 "server": "node web.js",
},
```

3. npm automates running

51



54



webpack Introduction

Just touching on this build tool

55

webpack's main purpose is bundling

But it can do lots of other things ...

- Module loading
- Minification
- Transpiling
- Compressing images
- Copying new files
- Deleting old files
- Running a development server
- And many more!

56

Wait, what is *module loading*?

- webpack will scan your JavaScript files for require/import and export statements. It will then reorder those files so that they are read in the proper order. Eliminates the errors of listing the script files in the wrong order
- It avoids packing those script files which are not needed
- No need to maintain a separate config file with a listing of all scripts needed for particular sections
- The scripts are now self-documenting (in terms of what each requires).
- No need to list the JavaScript files in <script> tags

57

If webpack baked in all of its capabilities ...

1. It would be huge!
2. It couldn't be extended.

So they separate these capabilities as "plug-ins" and "loaders"

58

loaders	plugins
• Defines rules for bundling • Very simple • Work with a single file • Run during bundling	• Is a whole separate process • Can handle complex logic • Can work with many files • Run after the bundling

They are similar, but different

59

Some example loaders

- sass-loader - To transpile SASS into CSS
- css-loader - To parse css into injectable styles
- style-loader - To inject those styles via JavaScript.
- ts-loader - Transpile TypeScript into JavaScript
- babel-loader - Transpile ES2018 to ES5
- handlebars-loader - Handlebars.js library
- vue-loader - Transpile Vue.js into JavaScript
- jsx-loader - For React
- jshint-loader - For linting

60

Some example plugins

- HTML Plugin - Creation of HTML files
- Copy Plugin - Copies files to another directory
- Bundle Analyzer - Creates a tree map of all files
- Prerender SPA - Creates static pages on the server
- Modules CDN - Can retrieve libraries from a CDN and pack.
- PWA Manifest - Generate a manifest file from contents
- Friendly Errors - Analyzes bundle for common errors
- Dup Package Checker - Tells you if you have a duplicate
- Purge CSS - Tree-shaking for unused CSS rules

61

webpack.config.js

```
const HTMLPlugin = require('html-webpack-plugin');
module.exports = {
  entry: "./src/index.js",
  output: {
    path: "dist", filename: "scripts/bundle.js"
  },
  module: {
    rules: [
      {test: /\.js$/, u
      {test: /\.css/, u
    ]
  },
  plugins: [
    new HTMLPlugin({template:'./index.html'})
  ]
}
```

Setup is done in webpack.config.js

62

webpack.config.js

- entry - What is the topmost JavaScript file?
- output - Where do the bundled file(s) go?
  - path - where to put the built JavaScript
  - file - Name of the .js file
- module - What loaders do we need?
- plugins - What plugins do we need?

63

Automating using webpack devserver

- For simplifying the dev process. Not for production.
- Will monitor your source files and when one changes,
- Rebuild the app
- Restart the web server

```
Raps-MBP:webpack-workshop raps$ npm start

> webpack-treehouse-example@0.0.1 start /Users/
ebPack/webpack-workshop
> webpack-dev-server

Project is running at http://localhost:8080/
webpack output is served from /
[BABEL] Note: The code generator has deoptimise
pt/Class Labs and Demos/WebPack/webpack-worksho
Hash: 01fbde17898516dc0c22
```

64

Match the tool to its task

1. npm	A. Static code analyzer
2. webpack	B. Transpiler
3. Babel	C. Minifier
4. node_modules	D. Task runner
5. TypeScript	E. Testing framework
6. esLint	F. Bundler
7. Jasmine	G. None of the above
8. package.json	

65

tl;dr

- Modern JavaScript development requires a ton of libraries which in turn requires the use of a lot of tools:
  - npm - organizes libraries and automates the build
  - linters - reports bad code practices
  - unit testers - run automated test scripts
  - webpack - minifies, bundles, and automates the build
  - babel or typescript - transpiles to allow all code to run universally

66