**Pt 1 - What is web pack?**

In a typical html file, we may have a number of script files that we are loading in, which generally slows performance. Instead, you can have 1-2 script files.

We cam use webpack to bundle these different script files into that one file.

It takes all of our assets and outputs them to a production-ready bundle.

It can also process SASS / LESS files into CSS and only use them where we need it and convert JSX and ES2015 into vanilla JS, which browsers understand.

**Part 2 – how to load this?**

Npm init to create a package.json which will have info about project and track our dependencies, which will include webpack.

Npm install webpack –save-dev

This causes node modules to appear with all dependencies, and devDependeics will feature webpack.

Now use it:

Create an html with a link to ./bundle.js

Script.js:

Var message = require(“./script2.js”)

Alert(message);

Script2.js:

Module.exports = “tempus fuits”;

Run webpack to bundle the two files together: webpack script.js ./bundle.js

This will bundle the two into bundle.js

Webpack will grab script.js, which calls script2.js, and transform it and its dependencies, into ./bundle.js

Webpack entrypoint output

**Part 3 Webpack.config file**

Instead of writing the whole 3 word command, we write this config file

Create new file called “webpack.config.js”

Here:

Module.exports = {

// define entry point – the one that requires all the modules

entry: “./src/script.js’,

// define output point

output: {

path: ‘dist’, // this is our distribution folder that will be created

filename: ‘bundle.js // filename we want to output to..

}

}

create a source folder (the ‘src’)

place source file in there and output them into a distribution folder when done.

Move the scripts to source folder and delete bundle.js

Now run in the terminal:

Webpack

This runs webpack, and creaes the ‘dist’ folder that holds the bundle.js but index.html needs to have it’s src changed to “./dist/bundle.js”

**Part 4 – bable loaders**

A loader helps us perform transformations on our code. Convert es15 and JSX (a combo of JS and XML used in react) and SASS

1. Install bable-core and babel-loader via npm
2. Install presets we will be using to perform transformations. If we want to change es15, we need to install the es15 preset, etc
3. Configure webpack.config to tell webpack to use babel to transform our code.

Es15, we can use ‘const’ but it won’t be understood, so try that in the above the code.

Npm install babel-core babel-loader babel-preset-es2015 –save-dev

Check package.json file

Now configure config by adding a module:

Output: { …},

Module: {

Loaders: [

{

test: /\.js$/,

exclude: /(node\_modules)/,

loader: ‘babel-loader’,

query: {

presets: [‘es2015’]

}…

}

]

run webpack

tell wbpack about this via module property

loaders – an array to put all loaders you want in here

each loader is an object

the first one – test

test – test particular file – loader shouldn’t run on html, css, etc – just files that have a .js extention – so use regex to find this…

exclude – don’t run loader on every js file, such as node\_modules folder – ignore

loader – id the loader

query:{presets: [… – what kind of code do you want to transform with this?

Part 5 – CSS loaders

Css loaders to load only the styles we need to certain parts of our apps.

If our framework dynamically injects contents into website, then js different components, and each component loads only css necessary

Css-loader – loads css into js file

Style-loader adds our css into the DOM

Go to index.html

<div class = “intro”></div>

<h1>css loaders are awesome</h1>

enter src folder and create folder css

file: introcomp.css

in css file:

.intro { padding: 20px; background: maroon; color: #fff}

now install

npm install style-loader css-loader –save-dev

configure config file

in loaders array – first object already in there, so nex object

{ test… },

{ test: /\.css$/, loader: ‘style-loader|css-loader’}

pipe- runs both together.

Enter script.js

Require(‘./css/intro.css’);

Run webpack -🡪

Part 6 – SASS loader

Webpack can turn sass code into css –

Transform css into sass into css?

Change extension of intro.css to intro.scss

Then add this:

$mainColour: #7777bb;

intro{

…

background: $mainColour;

…

this targets the <h1> - its’ a nested style, it’s inside the class=”intro”

h1 {

text-transform: uppercase;

}

now install

npm install node-sass sass-loader –save-dev

configure - >

in the last test, just tack on ‘s’ to .css -> .scs

then pipe sass-loader to loaders

also in script, change .scss

run webpack ---