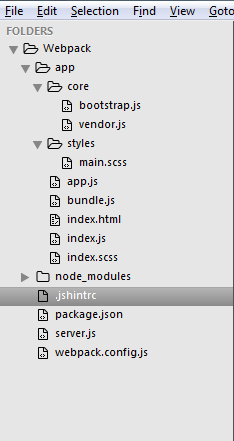
**AngularJs with Webpack**

**Creating angular project with webpack:**

**create a project folder**

**/Webpack**

**Folder structure should be shown as below**

****

**(Ref:** [**http://www.shmck.com/webpack-angular-part-1**](http://www.shmck.com/webpack-angular-part-1) **)**

1. **Installing Webpack**

>npm init

>npm install -D webpack

create /index.html

/index.html

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title>Angular with webpack</title>

</head>

<body><div>Angular working {{1+1==2}}</div></body>

</html>

create /index.js

alert('loaded');

1. **create webapp.config.js**

/webpack.config.js

'use strict'

var webpack = require('webpack');

var path = require('path');

//var APP = \_\_dirname + '/app'; // Don't use this, this will effect livereload.

var APP = path.join(\_\_dirname + '/app');//Use this instead

//We need to give only entry file name and output file name.

//We need to include outfile in index.html

module.exports = {

context: APP,

entry: {

app: './index.js'

},

output: {

path: APP,

filename: 'bundle.js'

}

}

1. **Run webpack in cmd to build bundle.js**

>webpack

it will create bundle.js, include this in index.html

/index.html

<!DOCTYPE html>

<html>

<head lang="en">

<meta charset="UTF-8">

<title>Angular with webpack</title>

</head>

<body>

<script src="bundle.js">

</script>

</body>

</html>

1. **Create Webpack dev server**

Webpack -dev-server is quick and easy Node.js/express/socket.io app that creates your bundle.js on the fly and reloads it on change.

>npm install -D webpack-dev-server //install it as dev dependency locally.

**Hot mode**

Use hot mode plugin to live reload the module, only what changed. No need to load entire project.

Update thewebpack.config.js file

entry: {

app: ['webpack/hot/dev-server', './core/bootstrap.js']

},

plugins: [

new new webpack.HotModuleReplacementPlugin()

]

Again, install webpack globally

>npm install webpack-dev-server -g

No run the command

>webpack-dev-server --content-base app --inline --hot

They say with --hot , live reload should work, but it's not working.

Use --inline for live-reload, without this live reload won't work.

Add this command to the package.json for easy run

"scripts": {

"start": "webpack-dev-server --content-base app --inline --hot"

}

then run

>npm start

Now your webpack server is up and running.

Just try to modify index.js, it should automatically reflect in the browser.

**Webpack with AngularJs**

1. **Install Angular**

>npm install angular

1. **Create file /app/core/bootstrap.js**

/\*jshint browser : true\*/

'use strict'

//load the angular, no need to inject the file in the html, this does automatically

require('angular');

//load the main angular app file

var appModule = require('../index');

//bootstrap angular app

angular.element(document).ready(function(){

angular.bootstrap(document, [appModule.name], { // appModule.name will be taken automatically from the angular app which is configured in the index.js

//stricDi: true //To strictly check the dependency injection is done

});

});

1. **Create file app/index.js**

module.exports = angular.module('app', []);

1. **Make out bootstrap.js as an entry point in the webpack.config.js**

entry: {

app: ['webpack/hot/dev-server', './core/bootstrap.js']

}

1. **We will add all dependencies in seperate file. Create app/core/vendor.js**

module.exports = function(){

require('angular');

};

1. **Add require('./vendor')(); in the bootstrap.js in the first line.**
2. **Now run the >npm start, you should see 'Angular Working true' in the webpage.**

**Working with loaders**

**We must use loaders for live reload.**

1. **Add loaders plugin for scss/css, js, html, images**
2. **Use require('filepath') to load js,html,css/scss and images to make livereload to work.**

module: {

loaders: [

{

test: /\.scss$/,

loader: 'style!css!sass'

},

{

test: /\.js?$/,

exclude: /(node\_modules)/,

loader: 'babel',

query: {

presets: ['es2015'],

plugins: ['transform-runtime']

}

},

{

test: /\.(jpe?g|png|gif|svg)$/i,

loader: 'url-loader?mimetype=image/png'

},

{

test: /\.html$/,

loader: "html-loader"

}

]

}

To load the images inside the html also,we need to add image

1. **Use require('filepath') to load the js, image, css and html .**

**Load the css using require('../index.scss') instead of adding in html using link tag.**

**Eg:** require('../index.scss');

**Directive:**

Create a file **../**app/core/directives/graph-directive.js

module.exports = function(){

return {

restrict: 'AE',

scope: {

data: '=',

options: '='

},

template: require('core/views/graph/graph.html'), //load html file using require in the directive to make livereload work.

link: function(scope, attrs, ctrl){

}

}

}

Create a file **../**app/core/controllers/graph-controller.js

module.exports = function(GraphService){

var graphVm = this;

GraphService.getBarGraphData().then(function(data){

graphVm.data = data;

});

graphVm.options = {

chart: {

type: 'discreteBarChart',

height: 450,

margin : {

top: 20,

right: 20,

bottom: 60,

left: 55

},

x: function(d){ return d.label; },

y: function(d){ return d.value; },

showValues: true,

valueFormat: function(d){

return d3.format(',.4f')(d);

},

transitionDuration: 500,

xAxis: {

axisLabel: 'X Axis'

},

yAxis: {

axisLabel: 'Y Axis',

axisLabelDistance: 30

}

}

};

}

Create a file ../app/core/view/graph-example.html

<div>

<graph-directive data="graphCtrl.data" options="graphCtrl.options"></graph-directive>

</div>