USER – INTERFACE MAGIC

1. when to use require() and when define()? Difference between require and define.

Ans:

It is a functionality of requiresJS. if you want to use your module in any other module then define() your module with define() and With require we "just" load/use a module or javascript file that can be loaded by require.js for execution.

***Example:***

***main.js*** *:*

*/\*\*  
 \* Created by Priyanka Acharya on 5/8/15.  
 \*/****require***([**"jquery"**,**"employeeManager"**,**"archieve/employeeData"**], **function** ($,employeeManager,employeeData) {  
 alert(**"main.js funtion"**);  
 **var** main= employeeManager.loadEmployeeData();  
 alert(**"main.js end"**);  
  
 **return** main;  
})  
  
In the above code main.js we are using this module for execution and calling loadEmployeeData(); which is available in **employeeManager** module and we are not using this module anywhere in code.

***employeeManager .js***

*/\*\*  
 \* Created by Priyanka Acharya on 5/8/15.  
 \*/*

***define([”jquery”,”build/myHandleBar”], function($,handlebars) {  
 var Employee\_METHOD ={  
 handlerData:function(resJSON){***

***$(‘#mydiv’).html(handlebars(resJSON));  
  
},  
loadEmployee : function(){***

***$.ajax({***

[***url:"http://localhost:63342/NewnodeJsProject/public/data/studentData.json",***](url:%22http://localhost:63342/NewnodeJsProject/public/data/studentData.json%22,) ***method:’get’,  
 success:this.handlerData***

***})  
 }  
 };  
return Employee\_METHOD;***

***});***

In this case a module has been created using define Specifying the dependencies and it is getting used in main.js .

**Difference:**

The **define()**function accepts two optional parameters (a string that represent a module ID and an array of required modules) and one required parameter (a factory method).

The return of the factory method **MUST** return the implementation for your module (in the same way that the [Module Pattern](http://addyosmani.com/resources/essentialjsdesignpatterns/book/#modulepatternjavascript) does).

The **require()** function doesn't have to return the implementation of a new module.

Using **define()**you are asking something like *"run the function that I am passing as a parameter and assign whatever returns to the ID that I am passing but, before, check that these dependencies are loaded"*.

Using **require()**you are saying something like *"the function that I pass has the following dependencies, check that these dependencies are loaded before running it"*.

The **require()**function is where you use your defined modules, in order to be sure that the modules are defined, but you are not defining new modules there.

1. How to use handlebars? How to compile, how to execute.

Handlebars templates look like regular HTML, with embedded handlebars expressions. Written in JavaScript, Handlebars.js is a compiler that takes any HTML and Handlebars expression and compiles them to a JavaScript function. This derived JavaScript function then takes one parameter, an object—our data—and it returns a string with the HTML and the object properties’ values inserted into the HTML. So, we end up with a string that has the values from the object properties inserted in the relevant places, and insert that string as HTML on the page.

The following example shows how to compile and execute handlebars.JS

I am basically building a small app, which will read values from a json file and display it with the help of handle bar.

The Basic App:

in index.html:

<body>

<div id="banner"><p>Enjoy the Power</p>

</div>

<br><br> <br><br><br>

<div id="mydiv" > </div>

<!--Handlebar templates start-->

<script id="employee-template" type="text/x-handlebars-template">

<table id="t01">

<thead>

<tr>

<th>Name</th>

<th>Email</th>

<th>Phone</th>

</tr>

</thead>

<tbody>

{{#each employeesArray}}

<tr>

<td>{{ this.name }}</td>

<td>{{ this.phone }}</td>

<td>{{ this.email }}</td>

</tr> {{/each}}

</tbody>

</table>

</script>

<!--Handlebar templates end-->

</body>

</html>

Basically from the script id= employee-template handlebars is getting initiated and now lets take a look at employeeManager.js file

/\*\* Created by Priyanka Acharya on 5/7/15. \*/

define([”bower\_components/jquery/dist/jquery”,”bower\_components/handlebars/handlebars”],function ($,handlebars) {

var Employee\_METHOD ={

handlerData:function(resJSON){

var templateSource = $(“#employee-template”).html(),

grunt.js

template = handlebars.compile(templateSource),

studentHTML = template(resJSON);

$(‘#mydiv’).html(studentHTML);

console.log($(“#employee-template”))

},

loadStudentData : function(){

$.ajax({

[url:"http://localhost:63342/NewnodeJsProject/public/data/studentData.json](url:%22http://localhost:63342/NewnodeJsProject/public/data/studentData.json)”, method:’get’,

success:this.handlerData

})

}

};

return Employee\_METHOD;

});

now in employeeManager.js file we are compiling the script using handlebars and then looping through the values.

1. How to use .hbs template and pre-compile it into .js file using Grunt for faster templating? How to use Grunt? How to add any task? How to install Plugin?

Grunt is very useful for faster precompiling. Grunt is very useful tool to convert .hbs file to js file and that is faster .

For doing that we need to follow some steps:

1. install -g grunt-cli (if doesn’t work or come out with errors use

sudo install –g grunt-cli

1. Then require a package.json file

{  
 "name": "NewnodeJsProject(Name of the Project)",  
 "version": "0.1.0",  
 "devDependencies": {  
 "grunt": "~0.4.5",  
 "grunt-contrib-handlebars": "^0.10.2",  
 "grunt-contrib-jshint": "~0.10.0",  
 "grunt-contrib-nodeunit": "~0.4.1",  
 "grunt-contrib-uglify": "~0.5.0"  
 }  
}

1. Next from command line npm install, which will rather install all the dependencies by its own and will notice a folder called node\_modules in project folder
2. next goal will be making Gruntfile.js file creation.. Here we need to use several options depending upon requirement. Options like: uglify (Zip, Unzip, Minify), Handlebars(for handlebar templates)

/\*\*  
 \* Created by Priyanka Acharya on 5/8/15.  
 \*/  
  
'use strict';  
  
module.exports = function(grunt) {  
  
 // Project configuration.  
 grunt.initConfig({  
 pkg: grunt.file.readJSON('package.json'),  
  
 handlebars: {  
  
 compile: {  
 files: {  
 'build/myHandleBar.js': 'template/myHandleBar.hbs'  
 }  
 },  
 options: {  
 namespace: false,  
 amd: true  
  
 }  
  
 }  
  
  
 });  
// Load the plugin that provides the "uglify" task.  
 // grunt.loadNpmTasks('grunt-handlebars-compiler');  
 grunt.loadNpmTasks('grunt-contrib-handlebars');  
// Default task(s).  
 grunt.registerTask('default', ['handlebars']);  
  
};

here It is helping to convert hbs file to js file using Grunt. Once converted it will convert hbs file and store it in the path provided and in the file name which is mentioned.

Here 'template/myHandleBar.hbs' is getting converted to myHandleBar.js' and getting stored in build folder.

After creating Grunt.js from command line need to run

grunt

So finally the .hbs file content:

<table id="t01">  
 <thead>  
 <tr>  
 <th>Name</th>  
 <th>Phone</th>  
 <th>Email</th>  
 </tr>  
 </thead>  
 <tbody>  
 {{#each employeesArray}}  
 <tr>  
 <td>{{ this.name }}</td>  
 <td>{{ this.phone }}</td>  
 <td>{{ this.email }}</td>  
 </tr>  
 {{/each}}  
 </tbody>  
</table>

and after grunt ‘s compilation done got a file called myHandleBar.js. Here we go for the compiled contents:

define(['handlebars'], function(Handlebars) {  
  
return Handlebars.template({"1":function(depth0,helpers,partials,data) {  
 var alias1=this.lambda, alias2=this.escapeExpression;  
  
 return " <tr>\n <td>"  
 + alias2(alias1((depth0 != null ? depth0.name : depth0), depth0))  
 + "</td>\n <td>"  
 + alias2(alias1((depth0 != null ? depth0.phone : depth0), depth0))  
 + "</td>\n <td>"  
 + alias2(alias1((depth0 != null ? depth0.email : depth0), depth0))  
 + "</td>\n </tr>\n";  
},"compiler":[6,">= 2.0.0-beta.1"],"main":function(depth0,helpers,partials,data) {  
 var stack1;  
  
 return "\n\n <table id=\"t01\">\n <thead>\n <tr>\n <th>Name</th>\n <th>Mark</th>\n <th>Subject</th>\n </tr>\n </thead>\n <tbody>\n"  
 + ((stack1 = helpers.each.call(depth0,(depth0 != null ? depth0.employeesArray : depth0),{"name":"each","hash":{},"fn":this.program(1, data, 0),"inverse":this.noop,"data":data})) != null ? stack1 : "")  
 + " </tbody>\n </table>\n";  
},"useData":true})  
  
});

After that we can easily grab the response in our js file from this precompiled one just by these lines:

define(["jquery", "build/myHandleBar"], function ($, handlebars) {  
 var Employee\_METHOD = {  
  
 handlerData: function (resJSON) {  
 $('#mydiv').html(handlebars(resJSON));  
  
 },  
 loadStudentData: function () {  
  
 $.ajax({

url: "http://localhost:63342/NewnodeJsProject/public/data/studentData.json",  
 method: 'get',  
 success: this.handlerData

})  
 }  
 };  
return Employee\_METHOD;  
  
  
});