**Introduction to using Handlebars in Node.js**

**References:**

* <https://handlebarsjs.com/guide/> This is where you download handlebars (or install from npm). The guide moves pretty quickly – it's better viewed as a reference than as a tutorial.
* <https://www.sitepoint.com/a-beginners-guide-to-handlebars/> An excellent place to start
* <https://code-maven.com/introduction-to-handlebars-javascript-templating-system>
This uses older style innerHTML code, but the examples are clear and well explained.
This is handlebars *in the browser.* Another good intro, in the browser, is at <https://sabe.io/tutorials/getting-started-with-handlebars-js>
* <https://www.geekstrick.com/create-website-with-handlebars-express-nodejs/#Installing_Express_Handlebars> is a clear tutorial on using handlebars with Express.
* <https://waelyasmina.medium.com/a-guide-into-using-handlebars-with-your-express-js-application-22b944443b65> Takes you a little further. Also at <https://www.youtube.com/watch?v=HxJzZ7fmUDQ>
* <https://www.youtube.com/watch?v=erfN7fH7A6s> has a series of short videos.

**The important ideas:**

1. Handlebars is a 'templating engine' – i.e. it allows you to create a basic template which many pages on your site can use (even incorporating CSS etc.).
You can then 'fill in the blanks' -i.e. the placeholders – with value which you send to the template.
2. Handlebars may be used either in the browser or in Node (e.g with Express) . You use it slightly differently in each place.

As the sitepoint page explains, here is how handlebars work *in the browser*:

Handlebars takes a template with the variables and compiles it into a function.

This function is then executed by passing a JSON object as an argument. This JSON object is known as context and it contains the values of the variables used in the template

On its execution, the function returns the required HTML after replacing the variables of the template with their corresponding values

In other words, you will write a template (inside a script), then you will ask the Handlebars.compile() function to turn your template into a f*unction*. You will pass the values of your variables as json (this json is called the *context)*  to the function and it will return some HTML – which you can put on your page.

1. When you use Handlebars with Express and Node, then you don't need to explicityly compile the template ---- Express takes care of this for you.
Obviously, however, you need to install various modules, tell Express that you are using Handlebars, etc. (see below).
You may also need
 **const bodyParser = require('body-parser');**
To use handlebars you will also need

 **const exphbs = require('express-handlebars');**

1. In Express it is customary to have certain *folders:* **public** --- will hold your static pages, including images and css files (& possibly
 your landing page). In other words, this folder will hold everything
 which does not change.
 **routes** – will have one file for each subsite; each file will specify how to handle a
 get, post etc. and it may handle several pages on the subsite

 **views** – will have the templates for whatever template engine you use. There
 may be a main page and a folder with layouts.

In order to have a place for your css etc. you will need to get to the public folder – so you will need the path module.

1. **All the things you need to start –** this is all in your **index.js** page, the entry point for your project:
//Setting up for Express
 **const express = require('Express');
 const app = express();**

//require any modules with data or middleware your created

 **const path = require('path');
 app.use(express.static(path.join(\_\_dirname, 'public'))); //set up public folder**

 //for any middleware modules you created and require'd, here you would say
 //app.use([optional route,] module\_name);

 //for any express middleware you will use, such as express.json, or
 //express.urlencoded({extended:true}) here you say

 //app.use([optional route,] express.whatever);

 **const bodyParser = require('body-parser'); //may be needed
 const exphbs = require('express-handlebars');**

app.use(bodyParser.json())

 app.use(bodyParser.urlencoded({ extended: true }))

 **app.set('view engine', 'hbs'); //sets the view engine
 app.engine('hbs', exphbs{
 extname:"hbs" , //so we can use hbs rather than handlebars**

 **defaultLayout:index //default is index.hbs in the layouts folder** //In waelyasmina reference you can see this reset

 **});**

1. **When you are done with all this your file structure will look something like:**

MyProject

index.js
other\_files\_such\_as\_data.js
package.json
node\_modules //you have seen this before; Express is here
middleware
 files\_you\_wrote\_to\_use\_in\_Express.js
public
 static page files for Express
 css and images files

routes
 files\_for\_routes\_on\_subsites
views
 main.hbs //holds the main template you will put content into
 layouts
 index.hbs
 partials
 more\_hbs\_files //holds things like nav or footer which you
 //can inject into your layouts; also other
 //parts of a page e.g. specific to a subsite.

Note: When you set up app.engine you can optionally change the layoutsDir, partialsDir etc but I recommend sticking with the defaults.

1. Now we wil create some templates.
	1. Partials
	If you have a header, nav, footer, etc. just make a file for each in the partials folder and name it header.hbs, nav.hbs etc. The file will just hold the html for that part of your original page.
	When you want that to appear in your index.hbs you will code
	**{{> header}}**
	2. Material sent from the body of our page – e.g. {{{body}}}
	3. Variable values – in the context, which is the second parameter for render().
	These are values which will be substituted in **main, not in the layout.**
	4. Putting it all together:
	The main.hbs page will look like:
	 usual <!DOCTYPE html> and head section, with links to scripts and css
	 <body>
	 {{ > header }}
	 {{ > nav }}
	 {{{body}}} //This is the file you will ask to render
	 //It will go into the default layout specified in main.hbs
	 {{ > footer}}
	 </body>
	 </html>

	The {{ > nav}} etc. partials will be filled in from the nav.hbs etc files in the partials folder.
	The {{{body}}} will be filled in from the index.hbs page (or whatever page you ask Express to render), using any variables from the json object which is the second parameter in the res.render() – e.g.
	 res.render('index', {fname: 'Kamala', lname:'Harris});

Please notice that instead of res.send() we will use the res.render() method.
It has two parameters --- the first parameter is main, which has the real content.
It will be rendered using the layout in our defaultLayout (the layout in layouts/index.hbs).
The second parameter is {key1:value1, key2:value2} which passes in values to be substituted – e.g. wherever the layout says {{key1}} you will get name1, etc.
Note: Handlebars allows you to iterate through an array with
 {{#each}} …/{{/each}} and the element you are at in the array is referenced with this. For example, if you have an array of objects, and one of the objects' properties is id then you may refer to this.id

*While handlebars describes itself as logicless, it does have an* ***if*** *and can loop with* ***each****.*

|  |
| --- |
| **Agenda for our day on Handlebars** |

1. Review this document up to this point

2. Finish the Brad Traversy video- he introduces Handlebars about at minute 59

<https://www.youtube.com/watch?v=L72fhGm1tfE&t=3483s>

3. Note that handlebars has 'helper' functions all of which are of the form
    <#helper>   ...... </helper>
   See <https://handlebarsjs.com/guide/builtin-helpers.html#if>

4.As time allows look at Esterling Accime videos<https://www.youtube.com/watch?v=erfN7fH7A6s&list=PLurIMwd6GdCi3ssXNAcjZ2l5mYaTfYPhf&index=1>

starting at the end of Video 1  & then on to videos 2 and 3.  This series will get you thru all the bells and whistles.  See video 7 on partials <https://www.youtube.com/watch?v=tb7081fzfdE&list=PLurIMwd6GdCi3ssXNAcjZ2l5mYaTfYPhf&index=7>