Open-Source Technology Use Report

Proof of knowing your stuff in CSE312

Handlebars

General Information & Licensing

Code Repository	github.com/wycats/handlebars.js
License Type	MIT
License Description	• Permission to reuse modify code to suit own needs even if code is proprietary software • License originated at MIT
License Restrictions	• Virtually no restrictions, may reuse and profit off of it. • The only requirement is to include the license and copyright notices in parts of the project where applicable.
Who worked with this?	Patryk Kasza

Handlebars.compile

Purpose

- This method will create a template that we can use to input our own data. It takes a string of html with unfilled data like this {{data}}. It outputs a function that can take a javascript object with data needed to fill the template.
- const templateFun1 = HandleBars.compile(templateHTML1);
  - Line 235 and 263
  - We need to create a template to render our data on the login page and home page.
- var data = templateFun1(templateDict1);
  - Line 236 and 264
  - This is the output function taking in javascript object templateDict1 which will fill the template with the data in the javascript object.

Magic ⋆★˚｡⋆୨୧˚ ☽ ˚⁀➴｡˚★彡⋆⁺₊ ༄

- This technology will allow us to render an html page. When receiving a HTTP request, we must build a response. This function will create a template and fill it with data so that we can build a http response with the html data we want.
- https://github.com/handlebars-lang/handlebars.js/blob/master/lib/handlebars/compiler/compiler.js
    - Line 50-74 is where the compile method is called
    - Starting at line 16 a function prototype is being created called Compiler. After the function compile at line 50-74 is run then the prototype becomes a function called Compiler. Which we said in the purpose will take in a context argument which can be used to render the template.
- Line 451-476

Magic

https://github.com/handlebars-lang/handlebars.js/blob/b0f1a62ddb7516694892485c365d97f090b58230/lib/index.js#L7

On the line above we require a folder that doesn't exist in the github repo, or at least I cannot find it. Then after on line 13 of the same file it exports it for us to use. The rest of the lines of code we don't use, as it provides another way of using the handlebars. We can import handlebars to instead export the compile function built within the extension function defined on line 16.

Because of the nature of the github repo(not having the files that we actually run, dist/cjs and its contents) I will link the node module lines to explain what is going on.

FILE IN QUESTION:
node_modules/handlebars/dist/cjs/handlebars.js
https://github.com/JunWeiCJW/TeamJJDDP/blob/main/node_modules/handlebars/dist/cjs/handlebars.js

Here we call: Line 39
```
_handlebarsCompilerCompiler.compile(input, options, hb);
```

And we pass in hb, which goes to node_modules/handlebars/dist/cjs/handlebars.runtime.js to create a new runtime object, which calls the function

FILE IN QUESTION:
node_modules/handlebars/dist/cjs/handlebars.runtime.js
https://github.com/JunWeiCJW/TeamJJDDP/blob/main/node_modules/handlebars/dist/cjs/handlebars.runtime.js

create() on line 40(called on line 57). Which creates and configures an object out of ->

node_modules/handlebars/dist/cjs/handlebars/base.js which basically sets up a bunch of helper functions for logging and such.

Returning a HandlebarsEnvironment(from now on as env)

Moving on to executing this function.
FILE IN QUESTION:
node_modules/handlebars/dist/cjs/handlebars/compiler/compiler.js
https://github.com/JunWeiCJW/TeamJJDDP/blob/main/node_modules/handlebars/dist/cjs/handlebars/compiler/compiler.js

Line 491, parses any options we passed through, we don't so we have {}.

Line 493, checks for invalid input. We pass valid so we move on

Line 497, we call _utils.extend({}, options);

FILE IN QUESTION:
node_modules/handlebars/dist/cjs/handlebars/utils.js
https://github.com/JunWeiCJW/TeamJJDDP/blob/main/node_modules/handlebars/dist/cjs/handlebars/utils.js

Our code doesn't go past Line 30, as both of our parameters are empty dictionaries, so

return an empty dictionary.
Based on line 31, if we did, we could call functions to load values and conditionals to parse for later.

FILE IN QUESTION:
node_modules/handlebars/dist/cjs/handlebars/compiler/compiler.js
https://github.com/JunWeiCJW/TeamJJDDP/blob/main/node_modules/handlebars/dist/cjs/handlebars/compiler/compiler.js

Line 498, empty dictionary so we end up setting the key ‘data’ to true in options

Line 505-533 we set up the function that will be called when we actually parse our template with the values we pass in.

We now return the ret function, with sub functions back to our main program in handlerGet.js line 235

Line 236 we call our ret function with a dictionary of values that we want to be displayed in our html

Back in compiler.js

Now since compiled is still undefined in this function scope, we call compileInput() on line 523 to assign it to compile.

Now we call env(our HandlebarsEnvironment from before).parse with our input and options(which is just ‘data’:true)

In base.js

Line 52 takes is to parseWithoutProcessing(input, options);

Line 35 is false as our input is text.

Line 46 is when we call parse from the parser object(line 7-388)
Here we define a bunch of properties to for our parser but most importantly is the fact that is all part of the parser variable which we see does a complex enum matching to switch statements to determine the action of the parser.

At the end of this we have an array of parts of our original html. Split up into ‘ContentStatements’ or ‘BlockStatements’ to determine the action of the parser when it is applying our data. BlockStatements have a reference to the variable name(under param) that you put in between curly braces to then match with the dictionary that you pass in later.

So parseWithoutProcessing, does just that. Parses the html we pass in a splits into a format that the parser can then just follow with the values from the defined enum determining its behaviour.

Line 55 in base.js essentially strips the whitespace from the ast variable.

Now in compiler.js

Line 509, we then compile it from the compile() object.(prototype to add functionality with mustache). Then we start compile on line 64

Line 64-84 we add functionality through _utils.extend with properties in line 74-82 and then return the new Environment object with new properties based on the ast variable and options.

Then we create a JavaScript compiler, and then call compile on line 73.(javascript-compiler()) which returns a some functions to be called when parsing our template.

We then continue to call our ret function on line 511, loading it with new properties and then compiling it through our parser variable.

template(context, options)

Purpose

- This is the function that Handlebars.compile() returns. This function replaces the data from the javascript object input.
- var data = templateFun1(templateDict1);
    - Line 236 and 264

Magic

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- How does this technology do what it does for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.
  - If there is more than one step in the chain of calls *(hint: there will be)*, you must provide links for the entire chain of calls from your code, to the library code that actually accomplishes the task for you.
  - Example: If you use an object of type HttpRequest in your code which contains the headers of the request, you must show exactly how that object parsed the original headers from the TCP socket. This will often involve tracing through multiple libraries and you must show the entire trace through all these libraries with links to all the involved code.

*This section may grow beyond the page for many features.

- After the function is