title
Chapter 7: Objects, JSON, and Templates

Recall from Chapter 3 this data type:

{% highlight javascript %} scotus = { "name": "Supreme Court of the United States", "justices": ["Roberts", "Scalia", "Kennedy", "Thomas", "Ginsburg", "Breyer", "Alito", "Sotomayor", "Kagan"] } {% endhighlight %}

This "object" `scotus` has two keys: `name` and `justices`, with corresponding string and an array values, respectively.

Let's create a new object, called `rules`.[^1]

{% highlight javascript %}

rules = { 101:{ "heading": "Scope; Definitions", "text": "(a) Scope. These rules apply to proceedings in United States courts. The specific courts and proceedings to which the rules apply, along with exceptions, are set out in Rule 1101.\n(b) Definitions. In these rules:\n(1) “civil case” means a civil action or proceeding;\n(2) “criminal case” includes a criminal proceeding;\n(3) “public office” includes a public agency;\n(4) “record” includes a memorandum, report, or data compilation;\n(5) a “rule prescribed by the Supreme Court” means a rule adopted by the Supreme Court under statutory authority; and\n(6) a reference to any kind of written material or any other medium includes electronically stored information." }, 102:{ "heading": "Purpose", "text": "These rules should be construed so as to administer every proceeding fairly, eliminate unjustifiable expense and delay, and promote the development of evidence law, to the end of ascertaining the truth and securing a just determination." } }

{% endhighlight %}

Like the `scotus` object above, this new `rules` object has two keys: "101" and "102". But the value of each key is also an object with two keys: "heading" and "text". So, the value of `rules[101]` is an object with the data associated with Federal Rule of Evidence 101. Going deeper: `rules[101]["heading"]` would result in "Scope; Definitions".

Another thing to note in the above `rules` object is the use of `\n`. This is called a "new-line character", and it is an instruction to the computer to add a line between what comes before and what comes after the character. The convention of using a backslash should look familiar to you, as that is something you learned about in Chapter 1 (recall `\d` and `\w`).

Now, let's use the `fre` function from the last chapter and pass the `rules` object as an argument to the function.

{% highlight javascript %}

// Define the rules object rules = { 101:{ "heading": "Scope; Definitions", *** } }

// Define the function with a "rule" parameter function fre (rulesObj, rule) {

```
// Access the value of the rule
r = rulesObj[rule]

// Check to see if the function input is a known rule of the Federal Rules of Evidence
if (r) {

    // Return a string in html (see chapter 2) with the applicable heading as a header and with the rule text as
    return "<h2> Rule " + rule + ". " + r["heading"] + "</h2><pre>" + r["text"] + "</pre>";
}

// Return a default, error message
return "I'm sorry, I don't know that rule. Please try again. There are " + rulesObj.length + "rules; surely somet
```

}

// Call the fre function and pass the rules object and an integer as arguments // Print the returned value to the console console.log(fre(rules, 101))

{% endhighlight %}

Here, we are taking advantage of the fact that, once we've defined the `rules` object, we can use the `fre` function to extract information from the `rules` object. This is a significant improvement over the 'if-else if' logic that the previous chapter contemplated. But, let's see if we can take it one step further.

Extending into the browser

One of the reasons we are now using javascript is because modern web browsers allow a website to execute javascript. [^2]

NOTE: Should what follows be in the console? Should it be in jsfiddle? Something else? I want to introduce the user to JSON and illustrate that it's powerful to define the data once and then allow the user to use it to modify the DOM. Is that too ambitious for a single chapter?

ALSO NOTE: I am not sure whether gh-pages supports CORS. If not, I may need to make this live on S3.

FINAL NOTE: I don't really have the time or energy to copy and paste the full FRE into this object. Does anyone know if it exists somewhere else? If not, why not?

The next snippet of code is a bit involved, but try reading it through and see if you get the gist. We'll explain each part in detail. To see it in action, see http://codingforlawyers.com/examples/ch7_fre.html.

{% highlight html %}

Federal Rules

Federal Rules of Evidence

{% endhighlight %}

What did we here? As it turns out, those little lines of code do a number of really advanced things that we'll cover in future chapters.[^3] But for now, here's what you need to know:

[A description of all the stuff that happened in the script above]

Summary

If this chapter seemed like you were drinking from a fire hose, that's ok. There's a lot to unpack here. It may be helpful to re-read the chapter. It's my belief that, once you understand this chapter, the world of the internet really starts to open up. In the next chapter, we're going to ...

---

[^1]: A full copy of this object is available at http://codingforlawyers.com/examples/fre.json.

[^2]: Javascript can also be executed outside of web browsers using a tool called "node". I personally love node. It's exceptionally fast and, after you get the hang of it, it is sort of fun to work with. That said, it can be very frustrating and is far from intuitive to the uninitiated. All of this is to say, a lot of what node does doesn't work directly in the browser, so you don't need to worry about it now. Just know that it exists.

[^3]: As a preview, these lines of code implicate a HTTP request, dependencies, and callback functions.