Rev. 25Jan2019

Exercise 6

Getting a Value Dynamically

OK, the last exercise was kind of tough. Let's see if we can make this a little easier. You're familiar with this formulation:

```
let firstName = prompt("What's your first name")
```

We called that a *variable assignment* and said that it's made up of the keyword, `let`, followed by a variable name, followed by the `=` sign and then a value.

The value was the result of a user typing in their first name: whatever they typed became the value of firstName. That was an example of getting a value *dynamically*. What does that mean? Let's look at a simpler way of creating a variable:

```
let myName = "Hal"
```

We call that *hard-coding* a value. The computer doesn't have to calculate the value or ask the user for their name and *then* use that as the value for a variable name. Instead we *hard-code* a *static* value when we're writing code.

Most of the time, we don't have the luxury of knowing ahead of time what the value of a variable is. Since not everyone is named "Hal", hard-coding usually isn't a viable strategy. So most of the time, we're going to derive values through calculation of some kind — that is, dynamically.

In the example where we used `prompt` to get the value, we used dynamic assignment all on one line of code. Often, that's fine, but especially when we're working with complex code, it's helpful to break things into multiple lines of code: it just makes it easier to think about what we're doing.

Let's just work with examples of breaking things into multiple lines of code.

Here's where we want to end up:

> Congratulations! You're on your way to a new secure, profitable career as a programmer, Hal Helms

Looking at this, what can we say about it — especially in terms of dynamic *v*. static parts of the code? Is it all dynamic? Is it all static? Is it a mix of the two?

Just judging from this little data, I think a good guess would be that the congratulatory part is static; it probably doesn't change depending on the names provided. Of course, the names are definitely dynamic.

- ☐ In your `index.html` file, create a `<div>` with an `id` of `welcome` just before the `<script>` tag.

dynamic
- ☐ Over in `index.js`, create two variables, `firstName` and `lastName`, by `prompt`ing the user for that information

static
- ☐ Now, create a variable named `message` using the following text for the value: `"Congratulations! You're on your way to a new secure, profitable career as a programmer, "`

Now, we want to combine the two together. Take a look at this snippet:

```
let total = 5 ? 10
```

I want to add `5` and `10` together. What symbol should I replace the question mark with? You know this: it's the `+` symbol.

- ☐ We use the same `+` symbol when we want to combine text, so make another variable, `greeting`:

Combining text together is known as "concatenation". You were dying to know that, right?

```
let greeting = message + firstName + " " + lastName
```

The *value* of the variable `greeting` is the result of *dynamically* evaluating `message` and `firstName` and `lastName`. (I put the static `" "` in there to provide a space between first and last names.)

We can put the value of greeting in the `<div id="welcome"></div>` element you created earlier.

- ❑ You've done this several times before:

```
document.getElementById("welcome").innerHTML = greeting
```

Another way of doing this is to break this into multiple lines. In the first line, we just grab hold of the HTML element:

```
let welcome = document.getElementById("welcome")
```

In the second line, we set `welcome`'s `innerHTML` to `greeting`

Astute readers may say "Didn't you forget the 'let'?" No, we use let only when we're creating variables. Here, we've already created "welcome" and now we're just using it.

```
welcome.innerHTML = greeting
```

- ❑ Try that out. As always, if you get stuck, refer to `answer.js`.