JavaScript CodeAlong

1. Build a page and add these parts:

```
<script type="text/javascript">                                     1. container

//      this is a comment!
document.write("Hello. This is Javascript.<br>");                 2. write/writeln

for (var number = 1; number <= 10; number++) {                         3. loop; var
        document.writeln("Counting to" + number + "<br>");      4. concatenation
};

</script>
```

Lessons:

- Needs to go inside **<script>** container with **type="text/javascript"**
- variables do not have a **$**
- end lines with **;**
- no **echo**; rather call the **write()** or **writeln()** method on the **document** object (we'll come back to this: it's called the Document Object Model, or DOM)
- the object accessor is a dot (unlike PHP's **->**)
- the concatenation symbol is a **+**
- JavaScript doesn't use variable interpolation (in fact, JavaScripters find the whole concept of interpolation to be bizarre)
- declare variables with **var**
- HTML tags can be concatenated in strings (But just wait—there are better ways to work with HTML.)
- comments are same as PHP: **//** or **/* */**
- has normal **if / else** like PHP
- JavaScript cannot read or write to your hard drive or to a database (for safety!)
- However, we will learn Ajax in another week or two, which solves that problem.

2. Array

```
animals = ['zebra', 'aardvark', 'monster'];                             1. array
for (var i = 0; i < animals.length; i++) {                2. arrays have properties
        document.writeln(animals[i] + "<br>");
};
```

An array is a built-in object that has built-in properties and methods:
http://www.w3schools.com/jsref/jsref_obj_array.asp

```
animals = ['zebra', 'aardvark', 'monster'];
animals.sort();                                          method on Array object
for (var i = 0; i <= animals.length - 1; i++) {
        document.writeln(animals[i] + "<br>");
};
```

Change to:

```
animals.sort().reverse();                                can chain references
```

---

3. A different way to iterate an array:

```
animals = ['zebra', 'aardvark', 'monster'];
animals.sort().reverse();

for (var property in animals) {                          Keep this in mind!
        document.writeln(property + ": " + animals[property] + "<br>");
};
```

---

4. Open JavaScript console to inspect elements

Safari:	Develop > Show Error Console
Chrome:	View > Developer > JavaScript Console
Firefox:	Tools > Developer > Web Console

You can inspect element and write to the console:

```
console.log("Hello to the console!”);                   <— console lives on window
```

---

5. Put .js in an external document

Create a new empty document, save it as **library.js**, and add this to the page:

```
<script type="text/javascript" src=“library.js”></script>
```

We’ll start to put some more complex things in the library. After any edits, remember to reload your page.

---

6. JSON == Associative Array:

```
var person = {                                           1. Note curly braces!
        first: 'Bob',                                    2. Note colon and ending comma
        last: 'Smith',
        email: 'bs@canada.com',
        phone: null                                    3. Note: null is a value
    }
```

What is that variable?

```
    > typeof(person)                                   <— It's an object!
    > typeof person                                    <— Parentheses optional on this one
    > person.first                                     <— and it has accessible properties!
    > for (var property in person) {                   <— iterate that with a loop
        console.log(property + ": " + person[property]);
    }
```

JavaScript has no associative arrays, but uses on-the-fly objects instead.

Moreover, although Javascript has objects, it has no classes (but this odd situation will change in a future version of Javascript). For now, the strategy is to build a **Prototype** object and clone it. (We'll do that after we learn functions.)

JavaScript has no equivalent of **protected** or **private** visibility.

7. Functions

```
    var string = 'Romeo and Juliet';
    document.write( italicize(string) + "<br>");

    function italicize(string) {
        return "<i>" + string + "</i>";
    }
```

8. I can put a function in a variable:

```
    var italicize = function(string) {
        return "<i>" + string + "</i>";
    }

    document.write( italicize("Hamlet") + "<br>" );
```

What kind of thing are we working with here?

```
    document.write( typeof(italicize) + "<br>");       <— function
```

9. Add a method to a JSON object:

```
var person = {                                      <— Same as JSON we saw before
    first: 'Bob',
    last: 'Smith',
    email: 'bs@canada.com',
    phone: null,
    fullname: function() {                          <— Assign function to property; note colon!
        return this.first + ' ' + this.last;        <— this is a pronoun variable
    }
}
```

Try it out in the console:

```
> person.fullname()
```

10. Access the functions via the console:

Try this in the console with the function code we wrote earlier in #8:

```
> italicize                          <--• lists the function
> italicize('goofus')                <--• executes the function with this input
> italicize()                        <--• calls the function with no input
    "<i>undefined</i>"
```

11. Using a Prototype Function to build an object

Prototypes will be replaced by full-fledged classes in the future, but for now they're the best way to build lots of objects that are exactly the same:

```
function Person(first, last, email, phone) {
    this.first = first;                         <— note equals sign here:
    this.last = last;                           <— different from the JSON syntax!
    this.email = email;
    this.phone = phone;
    this.fullname = function() {
        return this.first + ' ' + this.last;
    }
}
```

Now try it out in the console:

```
> var writer = new Person('Will', 'Shakespeare', 'ws@eliz.net', '555-4567')
> writer.fullname()                                    <— executes the method
> writer.fullname                                      <— just lists the method
```

Next up: Events and the DOM

View DOM:

Safari:	Develop > Show Page Resources
Chrome:	View > Developer > Developer Tools
Firefox:	Tools > Developer > Inspector