The JavaScripting English Major Cheat Sheet

Files & Basic Syntax (ch 2)

Write JavaScript in your `scripts.js` file, HTML in your `index.html`, and CSS (styles) in your `styles.css`. You can use different file names, but the JavaScript files should always end in `.js`.

End every statement (typically a line of code) with a `;`. Comment text out with `//`. All of the subsequent code on that line will not be executed.

Define variables using `let variableName;` and then assign it with `variableName = "some string";`, for example.

Data Types (ch 2)

`string`	Regular text, surrounded by `""`.
`number`	A number. Not surrounded by `""`.
`boolean`	`true` or `false`.
`array`	A list, surrounded by `[…]`.
`object`	Always surrounded by `{…}`.

If Statements (ch. 3)

If-then-else logic is central to program decision making.

```
if (true) {
  console.log("This will always print.");
}
```

What follows `if` in parentheses is a truth test that should respond either with `true` or `false`:

```
let a, b;
a = 10;
b = "someString";
if (a > 5) {
  console.log("This will print.");
}
if (a <= 5) {
  console.log("This will not print.");
}
if (b === "someString") {
  console.log("This will print.");
}
// set an inverse command with else:
if (b === "someString") {
  console.log("b is equal to 'someString.'");
} else {
  console.log("b is not equal to 'someString.'");
}
```

Functions (ch 4 & 6)

JavaScript provides the verbs of the web. If a page does something because of you, that's JavaScript. The verbs of JavaScript, however, are functions. Functions receive parameters and `return` a value:

```
let myFunction, myReturnValue;
myFunction = function(param1, param2){
  return param1 + " " + param2;
};
myReturnValue = myFunction("JavaScript", "is OK!");
// myReturnValue is now "JavaScript is OK!"
```

Parameter names are arbitrary and exist only inside the function. Methods like `.forEach()` are functions:

```
myArray.forEach(function(value, i){
  console.log("index: " + i + ", value: " + value);
});
// "index: 0 value: a"
// "index: 1 value: 1"
// "index: 2 value: string"
// "index: 3 value: 23"
```

Arrays (chs 5, 6, & 11)

Arrays are a list of things (strings, objects, arrays). Every item in the list has an index (that begins with 0) that can be used to access it:

```
let myArray, zerothArrayItem;
myArray = ["a", 1, "string", 23];
zerothArrayItem = myArray[0];
// zerothArrayItem is now "a"
```

Arrays also have a `.length` property and useful functions built-in as methods, like `.map()`, which takes the array and builds a new one.

```
let newArrayLength, newArray;
newArrayLength = newArray.length;
// newArrayLength is now 4
newArray = myArray.map(function(value) {
  return value + 1;
});
// newArray is ["a1", 2, "string1", 24]
```

The `.forEach()` method works similarly to `.map()` and iterates over the array—say, a set of points on a map—in order to execute useful commands on each item. See above.

Other Resources

These basics of JavaScript should lead you much of the way towards being able to look up other methods and looking up questions online. Here are some useful references:

1. `https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference` — MDN JavaScript reference
2. `http://api.jquery.com` — jQuery documentation
3. `http://leafletjs.com` — Leaflet homepage
4. `https://getbootstrap.com/docs/` — Bootstrap documentation
5. `https://stackoverflow.com/questions/tagged/javascript` — Stack Overflow JavaScript questions. Priceless resource
6. `https://www.w3schools.com/TAGs/` — HTML reference
7. `https://www.w3schools.com/cssref/` — CSS reference
8. `https://www.w3schools.com/colors/colors_picker.asp` — Color picker

Objects (chs 5, 6, & 11)

Objects are generic types and have arbitrary properties that you can assign. They are the multi-purpose blank slate of JavaScript.

```
let myObject, myName;
myObject = {
  name: "JavaScript",
};
// Access a property:
myName = myObject.name;
// myName is now "JavaScript";
```

Properties can be any data type, including functions:

```
myObject = {
  name: "JavaScript",
  favNumbers: [1, 3, 5, 6],
  favGreeting: function(name){
    return "Ahoy, ahoy, " + name + "!";
  }
}
myObject.favNumbers.map(function(i){
  return i + 2;
});
// returns: [3, 5, 7, 8]
myObject.favGreeting("Bhalu");
// returns: "Ahoy, ahoy, Bhalu!"
```

jQuery (chs 8, 9, 11, & 14)

jQuery lets you select parts of the webpage with `$("entity")` selector and manipulate them:

```
$("#response").html("New <em>HTML</em> text.");
// Changes the value of <div id="response"></div>
let pHtml, theParagraph;
pHtml = $("p").html();
// pHtml is the contents of the first <p></p>.
theParagraph = $(".a-class").html();
// theParagraph is the contents of
// <p class="a-class"></p>
```

`.html()` is a method, but there many, many useful ones in JQuery. Here are two more:

```
$("p").click(function(){
  // do something when you click on <p></p>.
});
$.getJSON("some.url.of/file.json", function(obj){
  // obj is the JSON object & you can manipulate it:
  $("p").html(obj.someProperty);
  // change the value of the first <p></p> to the
  // value of someProperty
});
```

CSS (ch 8)

CSS (cascading style sheets) is the language we use to control how things look on the page, in terms of colors, fonts, sizes, margins, etc. Styles are defined in a file ending in `.css`.

Styles use the same syntax for tags, ids, and classes as jQuery uses in its `$("entity")` selector.

```
p {
  color: #657b83;
  background-color: #fdf6e3;
}
.some-class-name {
  font-size: 24px;
}
#some-id-name {
  margin: 20px;
}
#leaflet-map {
  height: 500px;
}
```

CSS is magic, so I recommend using pre-defined styles like those in Bootstrap and tweaking them. Colors are defined either as hex values or rgb values. See the other resources section for more.

HTML (ch 8)

HTML is a relatively simple language made up of a `<tag>` that contains information inside and are then closed with a similar tag: `</tag>`. Here is a sample `index.html`:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>My Page Title</title>
    <link rel="stylesheet" href="styles.css">
  </head>
  <body>
    <h1 class="header">This is my project!</h1>
    <div id="response">
      <h2 class="header">This is a subhead</h2>
      <p>
        This is a paragraph inside the #response div.
      </p>
    </div>
    <div id="leaflet-map">Leaflet map</div>
    <script src="https://code.jquery.com/jquery
     -3.2.1.min.js"></script>
    <script src="scripts.js"></script>
  </body>
</html>
```

Information about the page and pointers to CSS stylesheets go between the `<head></head>` tags. The content of the page goes in the `<body></body>` tags, with the JavaScript files loaded at the bottom inside `<script></script>` tags.

Inside a tag, we can set attributes with values, like `id="response"` in the `<div>` in the example. In addition to `id=`, `class=` and `src=` (for "source") are common attributes. `id=` should give a unique name to an HTML object, while many objects can have the same `class=`.

Some other useful tags include:

`<p></p>`	paragraph
`<a href="http://…"></a>`	anchor, for making links
`<img src="http://…" />`	images
`<h2></h2>`	2nd-level heading (down to `<h6>`)
`<div></div>`	a generic block of content
`<span></span>`	a generic span of inline content
`<em></em>`	emphasis (typically italics)
`<strong></strong>`	strong (typically bold)

The jQuery selector can grab HTML objects based on the tag type, the id, and the class. See the jQuery section for more.

Leaflet (chs 10, 11, & 14)

Leaflet requires some added stylesheet and script additions in your HTML file (see ch 10). The map also needs to be drawn in a `<div>` with a pre-defined height (see the CSS section here). Defining a map then requires that `<div>`, a center point, and a tile resource to draw the background map.

```
let map, center, tileLayer;
// draw the map in <div id="leaflet-map"></div>:
map = L.map("leaflet-map");
// define the center as [latitude, longitude]:
center = [40, -72];
// set the tileLayer to a tile url:
tileLayer = L.tileLayer("some.url.of/tiles.png", {
      attribution: "&copy; rights holders",
      subdomains: "abcd",
      maxZoom: 18
}).addTo(map);
// now set the view with 9 as the zoom level:
map.setView(center, 9);
```

Once the basic map is drawn, you can add markers and lines to it, which can be styled in a way similar to CSS. Every Leaflet method is of the form `L.someMethod()`.

```
let marker, line;
// draw a circleMarker at the center:
marker = L.circleMarker(center, {
  radius: 5,
  fillColor: "#00aa00", // green
  fillOpacity: 0.8 // make it a bit transparent
}).addTo(map);
// draw a line between the circleMarker and a
// new point at [42, -74]:
line = L.polyline([center, [42, -74]], {
  color: "#aa0000", // red
  weight: 5 // line width in pixels
}).addTo(map);
// change the radius and fillColor of marker:
marker.setStyle({
  radius: 20,
  fillColor: "#0000aa" // blue
});
// add a popup that shows when you click on marker:
marker.bindPopup("This is a <em>popup</em>");
```

Leaflet also can respond to events like clicking:

```
// show the coordinates whereever you click:
map.on("click", function(clickEvent) {
  alert("You clicked at " + clickEvent.latlng.lat +
     ", " + clickEvent.latlng.lng);
});
```