Frontend Building Blocks for Web Mapping

Shruti Mukhtyar @mapchitra

Spatial Data Science Bootcamp

May 22, 2015

Before we start

Open a terminal, and

cd /home/oski/BootcampMaterials

git clone https://github.com/berkeley-gif/webtools-tutorial.git

cd webtools-tutorial

python -m SimpleHTTPServer

Open Chrome browser. Type in url -> localhost:8000

Tasks

1. Basic Tools - Code Editor & Chrome Developer Tools
2. Quick Review of Client & Server
3. HTML
4. CSS
5. DOM in simple english
6. Create a good looking web page in minutes
7. JavaScript - some parts
8. Tools to improve your productivity

1. Basic Tools - Code Editor & Chrome Developer Tools

Basic Tools - Code Editor

- Exercise 1a

Open Sublime Text

File - Open Folder - /home/oski/BootcampMaterials/webtools-tutorial

Click on any file in the exercises folder

- Other editors - PyCharm, Notepad++, Vim, IDLE, Gedit, TextMate
- Syntax highlighting, indentation, autocomplete, bracket matching
- Run interpreters, debuggers

Basic Tools - Chrome Developer Tools

- Exercise 1b

In Chrome go to tab where you are running this slideshow

Press Ctrl+Shift+I (or F12) to open developer tools

Check out the different panels

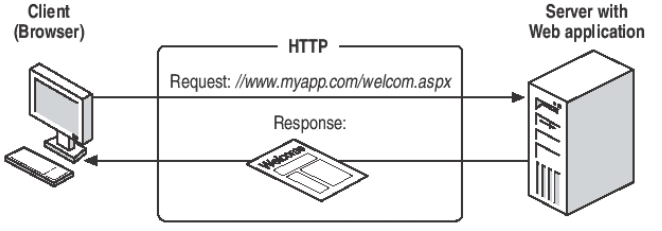
In Console panel type:

```
console.log('Hello World')
```

- More info on Chrome Developer tools here

2. Quick Review of Client & Server

Quick Review of Client & Server



- Exercise 2

Open Network Panel on Chrome Developer Tools

Reload the page. Look at all the activity there, and this is a simple page!

- HTTP, language of the web
- Browser sends HTTP GET Request. Server sends response for each request with content and/or status message.

How Does the Web Work?

3. HTML

Hyper Text Markup Language (HTML)

- Role » Describes content (not presentation)
- Tags, elements, attributes
- Block tags & Inline tags
- Nesting tags creates hierarchy
- Classes & ID's
- HTML Reference. Open this link in a new tab.

Let's write some HTML

- Exercise 3

Open _html.html in a new tab

Open file _html.html in Sublime Text

We'll look at it together

Create an ordered list in the body

Add a class attribute to titles of blog post

Add a id attribute to span element

4. CSS

Cascading Style Sheet (CSS)

- Role » CSS is the presentation of the content
- Selectors, properties, and values
- Cascading » style properties cascade down through all child elements
- Box model » how elements are positioned
- CSS Reference. Open this link in a new tab.

Ways to add CSS to a HTML document

In between the head tags:

```
<style>
  h1 {
    font-family: "Helvetica", "sans-serif";
    font-weight: bold;
    background-color: #000;
  }
</style>
```

As an external file:

```
<link rel="stylesheet" href="myStyle.css"/>
```

Inline style:

```
<p style="font-size:16px;"/>This is a paragraph</p>
```

Let's write some CSS

- Exercise 4

Open _css.html in a new tab

Open file _css.html in Sublime Text

We'll look at it together

Add following styles in head for:

Ordered list, see ul style there for reference

id "tagline"

class "post-title"

Add inline style to:

div with image

5. DOM

Document Object Model (DOM)

- Structured representation of the document (a tree) created by the browser (not you!)
- Programming interface for HTML, XML and SVG documents
- Connects web pages to JavaScript
- DOM in simple English, please!

Let's manipulate the DOM

- Exercise 5

Open _css.html in a new tab

Open Chrome Developer Tools

In Console panel type or copy-paste:

```
var tag = document.getElementById("tagline");
```

Change the text inside this element:

```
tag.innerHTML = "My new tagline";
tag.innerHTML = '<span id="tagline" style="color:blue;">My new tagline in blue</span>';
```

6. Create a good looking web page in minutes

CSS Frameworks

- Software frameworks that allow easier, standards-compliant web design
- Most popular CSS Frameworks -> Bootstrap, Foundation
- Exercise 6

Open _bootstrap.html in a new tab

We'll look at it together

7. JavaScript - Some Parts

JavaScript

- "Easy to learn, hard to master"
- Role » Creating interaction
- Interpreted by your browser
- JavaScript Reference

Variables

- Container for storing values
- Loosely typed
- Always use var when declaring for first time
- Variables can store values, expressions, objects, functions

No exercise files. We will type JavaScript directly into Console panel in Chrome's Developer Tools. You can copy paste from html slideshow. I encourage you to type if code blocks are small. Use Shift + Enter keys for multiline code blocks, e.g. functions.

Variables

- Exercise 7a

Define variables.

```
var count = 5;
var text = "bananas";
console.log(count); (or count;)
typeof(text);
```

Try out + operator, * operator

```
count + text;
count * text;
```

Try out == operator, === operator

```
var countStr = "5";
typeof(countStr);
countStr  == count //??
countStr  === count //??
```

Objects {}

- Container for your data
- Collection of properties
- Property is an association between a name and a value (key : value pairs)
- A property's value can be any type - numbers, strings, functions, arrays, even other objects
- A property whose value is a function is known as a method

Objects

- Exercise 7b

Define an empty object. Many ways to define, we will use Object Literal Notation.

```
var mySample = {};
typeof(mySample);
```

Add a new property using Dot Notation

```
mySample.name = 'MW-1';
```

Access the property using Dot Notation and store it in a variable

```
var well = mySample.name;
```

Assign the name property a different value.

```
mySample.name = 'MW-5';
well; //??
```

Objects

- Exercise 7b (more)

Another way to add a new property - Bracket Notation

```
mySample["height"] = 50;
```

Property names are stored as strings inside the object. Let's look what the last 2 lines in the next code snippet returns.

```
var key = "height";
mySample[key]; //??
mySample.key; //??
```

With bracket notation you can use functions, evaluate expressions.

```
var myFunc = function (){
  return "height";
}
mySample[myFunc()]; //??
```

Objects

- Exercise 7b (still more)

Add a new property which is an object

```
mySample["location"] = {
  "lat": 37.87,
  "lon": 122.27
}
```

Add a new property which is a function

```
mySample.getLatitude = function(){
  return mySample.location.lat;
}
```

Objects

- Exercise 7b (lots more)

Object Literal Notation

```
var mySample = {
  "name" : "MW-1",
  "height" : 50,
  "location" : {
     "lat": 37.87,
     "lon": 122.27
  },
  "getLatitude" : function (){
      return mySample.location.lat;
  }
}
```

Objects

- Exercise 7b (even more)

Iteration. Get a list of property names.

```
for (var key in mySample) {
  console.log(key);
}
```

Get a list of property values.

```
for (var key in mySample) {
  console.log(mySample[key]);
}
```

Arrays []

- Exercise 7c

We are going to keep working in console. Define an array

```
var arr = [];
```

Assign values

```
arr[0] = "sample";
arr[1] = false;
arr.push({ "name" : "MW-1"});
```

Access values

```
arr[0];
var i = 2;
arr[i];
```

Arrays

- Exercise 7c (more)

Arrays are objects!

```
typeof(arr);
arr["size"] = 9;
for (var key in arr) {
  console.log(key);
}
arr.length;
```

Use this syntax for looping over arrays

```
for (var i=0; i < arr.length; i++) {
  console.log(arr[i]);
}
```

Functions (){}

- Functions are objects too! Assign them to variables. Pass them around as data.
- Can be written in many ways.
- Anonymous function

```
function(x, y){
  return x + y;
}
```

- Named function

```
function multiply (x, y){
  return x + y;
}
```

Functions

- Anatomy of a function expression

```
var multiply = function(x, y){
  return x + y;
}
multiply(2,3);
```

multiply » Function name

x, y » Parameters

return x + y; » Function body

{} » Function definition/declaration

2,3 » Arguments

multiply(2,3) » Invocation, calling the function

Closure

- You can nest a function within a function
- The inner function can be accessed only from statements in the outer function
- The inner function forms a closure
- The inner function can use the arguments and variables of the outer function, while the outer function cannot use the arguments and variables of the inner function

Closure

- Exercise 7d

Copy to console

```
function outside(x) {
  function inside(y) {
    return x + y;
  }
  return inside;
}
```

Type these statements. Use typeof and console.log to examine fn_inside

```
fn_inside = outside(3);
result = fn_inside(5);

result1 = outside(3)(5);
```

Callbacks

- A function passed to another function (let's name it anotherFunction) as a parameter
- You are only passing the function definition, it is not getting executed in the parameter
- Function is executed inside anotherFunction
- Callbacks are also closures
- You will use this almost right away the first time you make a web map.

Callbacks

- Exercise 7e

Copy to console. Define function fullName. Takes 3 arguments - last one is callback

```
function fullName(firstName, lastName, callback){
  console.log("My name is " + firstName + " " + lastName);
  callback(lastName);
}
```

Define function greeting. This will be our callback function.

```
var greeting = function(ln){
  console.log('Welcome Mr. ' + ln);
};
```

Call fullName, pass it greeting without parentheses. greeting will be executed later

```
fullName("Jackie", "Chan", greeting);
```

Module Pattern

```
var MYMODULE = (function () {

    // variables and functions private unless attached to API below
    // 'this' refers to global window

    // private array
    var array = [];

    // add a number into array
    function add(a) {
        log("add "+a);
        array.push(a);
    }

    // return copy of the array
    function get_array() {
        log("copy_array");
        return array.slice();
    }

    // a private debug function
    function log(msg) {
        console.debug(msg);
    }

    // define the public API
    var API = {};
    API.add = add;
    API.get_array = get_array;

    return API;
}());
```

Module Pattern

- Way to group functions and data in a private namespace
- By using Javascript's most powerful feature, closures, it is possible to create an anonymous function, invoke it immediately and create a closure of functions and data
- "Learning JavaScript Design Patterns" - Andy Osmani (free ebook)

Source: Blog post

JSON

- Exercise 7f

Copy to console. Let's try to get at the data

```
var obj = {
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": ['Jack','Jill','Bo'],
  "spouse": null
}
```

GeoJSON

- Exercise 7g

Copy to console. Let's try to get at the data

```
var obj =
{ "type": "FeatureCollection",
    "features": [
      { "type": "Feature",
        "geometry": {"type": "Point", "coordinates": [102.0, 0.5] },
        "properties": {"prop0": "value0"}
       },
      { "type": "Feature",
         "geometry": {
           "type": "Polygon",
           "coordinates": [
             [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
               [100.0, 1.0], [100.0, 0.0] ]
           ]
         },
         "properties": {
           "prop0": "value0",
           "prop1": {"this": "that"}
         }
      }
    ]
}
```

- geojson.io, Validate your geojson, Mapshaper

8. Tools to improve your productivity

Tools to improve your productivity - Bower

If you want to start somewhere, use Bower first. What is Bower? A Package Manager for front-end, like pip for python packages

Bower requires Node and npm and Git. Easy to install on Windows, Ubuntu, Mac. Typical Ubuntu installation looks like this:

```
sudo apt-get install nodejs
sudo apt-get install npm
sudo npm -g install bower
```

To use it:

```
cd your-project-directory
bower init
bower install leaflet --save //installs leaflet
bower install git://github.com/user/package.git
bower install http://example.com/script.js
```

Install, remove, search packages. Package names are saved to bower.json.

Tools to improve your productivity

Next checkout Grunt or Gulp. What is Grunt? A JavaScript Task Runner

What does it do? Automate your workflow. Clean up code, compress CSS, contcatenate and minify JS files, Optimize images, etc.

Typical installation. First create package.json with npm init in your project folder.

```
cd your-project-directory
npm init
npm install grunt-cli --save
```

--save, --save-dev flags write dependecies to package.json file.

Next checkout Testing Frameworks - Jasmine, Moncha, Karma, etc.

Resources

- Getting Started Web Development Guide
- Google search: favor results from stack exchange, Mozilla, CSS-Tricks
- Web Design - A List Apart
- CodeAcademy - JavaScript, HTML/CSS (Beginners)
- Courses for all levels on CodeAcademy, Udacity, Coursera
- JavaScript: The Good Parts by Douglas Crockford (Intermediate)
- Frontend Masters (All levels but not free)