

JavaScript ICE I

JavaScript is an interpreted, object-oriented programming language that enables interactive, dynamic web pages. It is the only language supported by all browsers for front-end (client-side) scripting, and it can also be used for back-end (server-side) scripting, effectively making it the “language of the web.” JavaScript can also be used in non-browser environments, such as node.js. JavaScript includes the formal specification for the core language (technically called ECMAScript 6.0), the Document Object Model (DOM), and many Web APIs (e.g., WebGL, device APIs hardware features like Geolocation, Communication APIs like WebSockets, and Data management APIs).

This exercise will introduce you to JavaScript, which has become an essential tool in a programmer's toolkit. No prior experience in JavaScript is assumed, although we do assume some comfort with basic programming concepts learned in Python, such as declaring variables, expressions, functions, selection and iteration statements, string operations, arrays and other basic data structures.

Also in this lesson, we will set up a web server on your VM in EECSnet, set up Remote FTP in Atom, and build a web page locally and deploy it to the web server.

Set up Local Folder

Extract the compressed folder containing the HTML, CSS, and images for the ICE¹ to a folder that is local to your computer. Launch Atom, and select **File > Add Project Folder ...** to add this folder as a project.

Install the Web Server

We will be installing Apache2 (<https://httpd.apache.org/>) web server which is one of the most popular web servers. VPN into your VM and open up a terminal using Putty. The following commands will install Apache, verify that the service is running, and configure the owner and permissions appropriately for the folder that our web server will host the web site from:

```
sudo apt update
```

```
sudo apt install apache2
```

```
sudo service apache2 status
```

```
sudo chown -R $USER:$USER /var/www/html
```

```
sudo chmod -R 755 /var/www/html
```

To verify that your web server is actually running, open a web browser (Chrome or Firefox) and enter the host name of your VM (firstname.lastname.cy355.eecs.net) into the address bar. You should see the default Apache web page.

You can see the web page file by

```
cd /var/www/html
```

```
ls
```

```
cat index.html
```



We can change and add to our web site by placing the appropriate files into this folder.

Adding Remote-FTP

In Atom, install the package **Remote FTP**. This package will allow us to upload/sync a local project to a remote package (i.e. our VM). Once you have installed the package, select **Packages > Remote FTP > Create SFTP Config File**. This will create the file **.ftpconfig**. Set the host name to your VM host name (i.e. "firstname.lastname.cy355.eecs.net"), the user name to "cadet", and the password to your password. In the remote directory, set it to **"/var/www/html"**. This allows us to work on our web pages locally and then upload them to web server (local > remote). Next, select **Packages > Remote FTP > Toggle** to see both the **Project (local)** and **Remote (remote)** folders. Finally, you will need to connect the folders (you could do this by selecting **Packages > Remote FTP > Connect**. **NOTE**: Make sure that you have just the **ONE** project (local) open that you are working in (**multiple open projects will cause connection problems**). Sync **index.html** (local > remote) to overwrite the default Apache page. Also upload the **images** and **styles** folders. Reload your web browser, and you should see the updated web page.

Using a Web Console

JavaScript can be written directly in an HTML file using script tags or in a separate file with a .js extension. You can also enter JavaScript commands directly into a browser console, although everything is lost once the page is reloaded or the browser is closed. The console makes great place to experiment though, so that's how we'll start.

Open a web console in either Chrome (**Ctrl+Shift+I**, select **Console** tab) or Firefox (**Ctrl+Shift+K**). Avoid using Internet Explorer. You can undock into a separate window in both Chrome (select the  icon on the top right, and then first option next to **Dock side**) and Firefox (select the  icon on the top right).

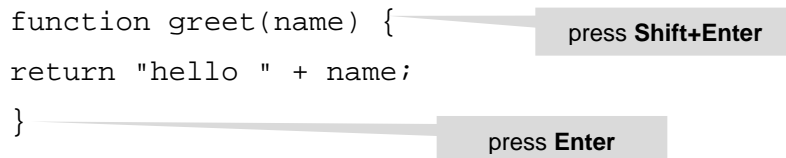
Test entering some simple expressions, pressing **Enter** at the end of each line:

let x=2	press Enter , see result:	undefined
x+4/x		4
let greeting = "hello"		undefined
alert(greeting)		("hello" pop up)

```
alert(prompt("Enter text"))           ("Enter text" dialog)
                                     followed by (text pop up)
```

To enter a multi-line expression:

```
function greet(name) {
  return "hello " + name;
}
```



Then invoke the function:

```
alert(greet("world"))
alert(greet(prompt("Enter your name")))
```

If you want to build and test multiple lines of JavaScript, you can invoke the ScratchPad (Shift+F4). You can then save this JavaScript to a file.

JavaScript declarations, expressions, and statements

You probably noticed that you can declare a variable with the keyword `let`. You can also use `var`, but it can have some unpredictable results due to what's called *hoisting*. This is not a concept that beginners should wrestle with, so you are strongly advised to just declare variables with `let` —it will always give you the intended results.

JavaScript in HTML

The JavaScript written in the console is non-persistent; it will be gone once we refresh the page. If we want our JavaScript to persist then we must place it either in the HTML document or in a separate file. Let's first look at the HTML document.

We can place our JavaScript in either the `<head>` or `<body>` elements. The design decision is whether we want the JavaScript to completely load (i.e. place in `<head>`) prior to loading the remainder of the page, load the page and then the JavaScript (i.e. place at the end of `<body>`), or somewhere in between. For this example we will place the JavaScript in the `<head>` element.

First, let's use the Pythagorean Theorem to calculate the hypotenuse of a right triangle with sides of the right angle equaling 3 and 4. We will then output the result to the console:

```
<script>
  let a = 3;
  let b = 4;
  let c = Math.sqrt(a*a + b*b);
```

```
        console.log("c = " + c);  
</script>
```

We can even insert an alert to let us know to check our web console by adding:

```
alert("Check your web console");
```

JavaScript in a Separate File

Just like CSS, if we think that we will reuse JavaScript across multiple pages then it makes sense to create a separate JavaScript file with the file extension of .js. In your ICE folder, create a folder called `scripts`. Then create a new file called `main.js`. Change the `<script>` element in your HTML file to the following:

```
<script src="scripts/main.js"></script>
```

 and place it right before the `</body>` closing tag.

Notice that all our previous content within the script tags is gone.

We would like to create a button that when we click on it, changes the picture from Firefox to Army West Point. First, we need to create a button element. Right before the `</script>` closing tag, write the following HTML:

```
<button id="changePictureBtn">Change Picture</button>
```

Save your HTML file, and refresh your browser to see the button with “Change Picture” at the bottom of the page. You can click the button, but the button does not do anything yet. The next step is to give our image an id attribute so that we can later refer to it. Find the `` element and set its id attribute to “mainPicture” as seen below:

```

```

Save your HTML file, and go to your JS file. Enter the following code:

```
let changeButton = document.getElementById('changePictureBtn');  
let mainPicture = document.getElementById('mainPicture');  
changeButton.onclick = function(){  
    mainPicture.setAttribute('src', 'images/awp.png');  
}
```

We use the `document.getElementById()` functions to set our variables of `changeButton` and `mainPicture` so that we can reference them later. When `changeButton` is clicked (i.e. the `onclick` event listener is triggered), then our JavaScript changes `src` attribute of `mainPicture` to the Army West Point image. We have changed our picture, but we would like to have the ability to toggle back and forth between images. We will need to add code to check

which image is currently displayed, and then display the other image. Modify our function as follows:

```
changeButton.onclick = function(){
    let curSrc = mainPicture.getAttribute('src');
    if (curSrc === 'images/firefox-icon.png') {
        mainPicture.setAttribute('src', 'images/awp.png');
    } else {
        mainPicture.setAttribute('src', 'images/firefox-icon.png');
    }
}
```

JavaScript Changing HTML Content

Let's now change the main heading `h1` of the page when we click the Change Picture button. First, we will need to get the `h1` elements. We can do this with:

```
let mainHeading = document.getElementsByTagName('h1');
```

This returns an `HTMLCollection` of `h1` elements which is similar to an array but is not an array. We want to change the text of the first (0th) `h1` element to say "Go Army!". We do this by inserting the following into our `if` statement:

```
changeButton.onclick = function(){
    let curSrc = mainPicture.getAttribute('src');
    if (curSrc === 'images/firefox-icon.png') {
        mainPicture.setAttribute('src', 'images/awp.png');
        mainHeading[0].innerHTML = "Go Army!";
    } else {
        mainPicture.setAttribute('src', 'images/firefox-icon.png');
    }
}
```

We can then reset the main heading to Mozilla is cool in a similar manner in our `else` clause. The `.innerHTML` method lets us set the content of HTML elements.

JavaScript Changing CSS Property

Finally, let's add a button to hide the picture. We can hide an HTML element by setting its display property to none. We will need to create a new button that we will use to hide the picture in our HTML file

```
<button id="hidePictureBtn">Hide Picture</button>
```

In our JavaScript file, we will then need to create the function that reacts to the clicking of the Hide Picture Button

```
let hideButton = document.getElementById('hidePictureBtn');  
hideButton.onclick = function() {  
    mainPicture.style.display = "none";  
}
```

As a challenge, can you change your HTML and JS to toggle showing the current picture (HINT: look at the display value of the image in your CSS file)

¹ Source code for HTML, CSS, and Firefox image from Mozilla Developer Network, "JavaScript Basics", https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics, last accessed October 19, 2016.