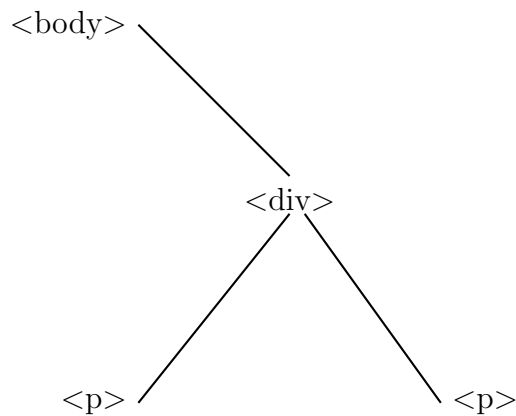


JavaScript Notes

Michael Dick

Lecture 1: DOM Review with Object Oriented Programming

The DOM (Document Object Model) is structured like a tree and every node is a parent. A DOM tree is drawn below:



How the tree works is that the root of the page is at the top which is the *document*. Each *element* is a node in the tree. The *node list* is a array of elements. *Attributes* of an element like an image are the size, lable, etc. Accessing the DOM is done with an API (Application Programming Interface). No matter the browser, scripting language, the API is the same.

Lecture 2: Output

JavaScript doesn't have a typical print function like C++ or other languages do. We will need to implement other means to ouput and debug items using JavaScript.

JavaScript Can:

1. Read and write HTML elements.
2. Reacts to events (i.e., mouse events, keyboard events).
3. Validate data.
4. Detect the visitors brower.
5. Create cookies.

JavaScript Output:

Data is displayed via:

1. Alert box using `window.alert()`
2. Prompting user using `window.prompt()`
3. HTML output using `innerHTML`
4. Browser console using `console.log()`

Using `alert()`: `alert()` creates a pop-up window that displays information. To make this work you will need to use the script tags, i.e., `<script> <\script>`. When you're wanting to use `alert`, use: `alert("string of choice");`

Example 1: Using Alert

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0"
7     >
8     <title>Example 1</title>
9     <script> alert("Hello World");</script>
10 </head>
11 <body>
12 </body>
13 </html>
```

Using `prompt()`: When using `prompt()`, it provides a pop-up like `alert()` does, but it wants user input. To use: `prompt("string of choice");`

Example 2: Using `prompt`

Emitting some of the HTML code, we see:

```
1 <head>
2     <title> Example 2 </title>
3     <script> prompt("This is using a prompt");</script>
4 </head>
```

Using `document.write()`: When using this command we write to the HTML document. What we write in this will write to the webpage itself. We will show that you can use HTML tags and it will display the information according to those tags. To use this: `document.write("string")` or `document.write("<h1>string</h1>")`.

Using `innerHTML`: This is used to change certain elements in the HTML document. Note: do NOT use parathesis for this. Only use equals.

Example 3: Using getElement:

```
1 <head>
2 <title>Example 3</title>
3 </head>
4 <body>
5 <p1 id = "test">This sentence will be turned to hello</p>
6 <script>document.getElementById("test").innerHTML = "Hello";</script>
7 </body>
```

Using `console.log()`: Using `console.log()` it only displays messages to the console using inspect element in a browser. This will be the main tool when using debugging in our code. Use: `console.log("Message you want displayed");`.

Lecture 3: Variables

Using variables in JS is like using variables in any programming language. The syntax is: `var varName = "Name of the variable";` Like any programming language, the item being updated is on the LHS, and the data that is going into the stored location is on the RHS.

Example 4: Assigning Var and Displaying it to the Screen

```
1 <body>
2 <script>
3     var name;
4     name = prompt("What is your name?");
5     document.write(name);
6     var date = Date(); //JS command that writes date.
7     document.write(date);
8     var location;
9     location = window.location; //JS command that write date.
10    document.write(location);
11    document.write("<h1>" + name + "</h1>"); //To write name
12 </script>                                     //using h1 tag.
13 <p>Using JS.</p>
14 </body>
```

Lecture 4: Data Types

In this lecture we went through how JS stores data. We can store numerical values, strings, arrays, and Boolean values.

With numerical values we just assign a numerical value to our variable. We don't need to specify if it is an integer, real number, etc. JavaScript doesn't care about that. Two values that are numerical:

```
var width = window.innerWidth; //A JS command to get window length.
var pi = 3.14;
```

Strings need to be put in double or single quotes. NEVER copy and paste quotes because they usually don't match your programming environment and will give you an error. A single example is given below:

```
var location = window.location //JS command that stores user location.
```

Boolean values in JS are stored as follows: `var status = false;`

```
var windowStatus = window.closed //JS command that set truth value from operation.
```

Objects from the DOM can be stored. These nodes are usually more than a single value because they have attributes.

```
var topic = document.getElementById('myID');
```

Arrays are used to store more than one value. we can use this to store many elements from the DOM.

```
var links = document.getElementsByTagName('a'); //Make sure it's Elements!
```

When using arrays we can display a certain item/ write it to the page: `document.write(links[0]);`

Lecture 5: Operations and Expressions

We've been using statements to execute our JS, and these statements have expressions. These expressions produce values. The format still holds where the LHS is the variable and the RHS generate the value that will be stored to the variables. The following are some expressions you'll see with numerical values.

1. `x=5;` `x++;` `x` is `6`
2. `x=4;` `x--;` `x` is `4`
3. `x=9;` `x+=2;` `x` is `11`

Now we can use the same expressions with strings. Strings become concatenated, meaning they combine both the strings together. Below will be the operations and the expressions they produce.

1. `x="Hi" + "There";` `x="HiThere"`
2. `x="Hi"` `x+5;` `x` is `"Hi5"`

Boolean expressions are also in JS. Something to note, if you have any expression with a inequality, that expression will always give a Boolean value. The example if `x=2;` and we have the expression `x>5`, the expression will return `false`.

More Boolean Expressions:

JavaScript doesn't care for types. When we compare a `x=12;` and `x="12"`, JavaScript will say these two are the same thing even though their types are different.

In JavaScript, to compare two types you'll use the triple equals, i.e., `===`. To examples are given below:

1. Assume `x = 15;`, then using `x == "15";` will return `true`.
2. Assume `x = 4;`, then using `x === "4";` will return `false`.
3. Assume `x = 8;`, then using `x !== "8";` will return `true`.

Lecture 6: Functions

Like in any programming language we use functions. When we build functions we want to make sure they will be used more than once and they will be generic enough so they can be used for multiple occasions throughout the program.

Function Syntax:

The function syntax is as follows:

```
1      function functionName(parameters)
2      {
3          code you want to run.
4      }
```

Writing functions in JavaScript you don't need to specify the return type of the functions. Functions also don't have to have a return type.

Example 5: Function In Use

```
1      <script>
2          function greeting_Msg(msg)
3          {
4              alert(msg); //Greeting the user.
5              var name = prompt("What is your name?"); //Asking for name.
6              return name; //The function returns the name.
7          }
8          var name = greeting_Msg("Hi"); //Runs function.
9          console.log(name); //See if the name was stored correctly.
10     </script>
```

Lecture 7: Code Placement

Placing your JavaScript code can be done in three places: the head, the body, or in an external file. When running your JavaScript in the same file as your HTML, make sure to write your functions in the head and then call your functions in the body.

Example 6: JS in <head> and <body>

```
1 <head>
2   <script>
3     function arrival_message()
4     {
5       alert("Welcome to this webpage!");
6     }
7   </script>
8 </head>
9 <body>
10  <script>
11    arrival_message();
12  </script>
13 </body>
```

As you see above, when using any JavaScript you need to include the script tags.

Example 7: Linking External Sheet

```
1 <head>
2   <script src = "js/js_code.js"></script>
3 </head>
4 <body>
5   <h1>Function Call</h1>
6   <script>
7     message(); //Calling function that is stored externally.
8   </script>
9 </body>
```

Lecture 9: Events

JavaScript uses events to bring some interactivity to your page. The general idea is to assign events, and when these events occur, a function will be called and whatever code that is defined in that function will run.

Events:

- onclick: User clicks on the HTML element and then the event is called.
- onmouseover: User moves the mouse over the HTML element.
- onresize: Event called when browser window is changed.
- onload: Event is called when the browser is finished loading the page. (Note: You'll want to use this one in the body because sometimes your page has a lot going on and the page will load, but all the JavaScript will have already executed.)

How Events Work:

Any HTML element can react to an event. In order for an element to react, you need to add the event to the tag and include what you want to happen. Below will be an example when you click on the `<div>`. When clicking on this tag, a function will be called.

```
<div onclick = "message()">Clicking on this will activate the fuction.</div>
```

Note: When calling the function for an event always use double quotes. The reason for this is when calling a string, you want to use single quotes. If you use single quotes for the function call, the webpage will say that the end of the single quote is the end of the function call. Example below:

```
<div onclick = "message('Hi')">
```

Example 8: Using onload, resize, and onclick

```
1  <script>
2      function message(msg)
3      {
4          document.getElementById("output").innerHTML = msg + " event";
5      }
6  </script>
7  </head>
8      <body onload = "message('LOAD')" onresize="message('RESIZE')">
9          <h1>JavaScript Events - Responging to Evenets</h1>
10         <p onclick = "message('CLICK')">It is possible to
11             call functions based on Events.</p>
12         <p>Clink on the paragraph above to trigger an <em>onclick</em>
13             event. Reload the page to see an onload event.</p>
14         <p id = "output"></p>
15     </body>
16 </head>
```

What is happening here is the `onload` event is calling the JS function when the page is done loading. the function `message(msg)` is finding the ID that matches the string `"output"`, and when it finds this string it writes what ever the input `msg` is and then adds the word `" event"`. When running this, where `id = "output"` is found and the string where the event is called is ran through that function and written where the `id = "output"`. This will happen when you load that page, click on the paragraph where the event is defined, and when you resize your browser window. A snapshot of *Example 8* is below:

JavaScript Events - Responging to Evenets

It is possible to call functions based on Events.

Clink on the paragraph above to trigger an *onclick* event. Reload the page to see an onload event.

CLICK event

Example 9: Using Button Event

In this code we are finding the id named "demo", then we replace the element with the `id = "demo"` with what we have the function defined to do, replacing it with the JavaScript function `Date()`.

```
1 <head>
2 <script>
3     function display_date()
4     {
5         document.getElementById("demo").innerHTML = Date();
6         //This writes over the <p> and diplays the date instead.
7     }
8 </script>
9 </head>
10 <body>
11     <h1>JavaScript Date Fucntion</h1>
12     <p id = "demo">This is a paragrah with id = demo.</p>
13     <button type = "button" onclick= "display_date()">Display Date</
        button>
14 </body>
```

Output:

Before Event

JavaScript Date Fucntion

This is a paragrah with id = demo.

Display Date

After Event

JavaScript Date Fucntion

Mon Oct 11 2021 16:27:26 GMT-0400 (Eastern Daylight Time)

Display Date

Lecture 10: Coding Examples

Example 10: Change ID Using JavaScript

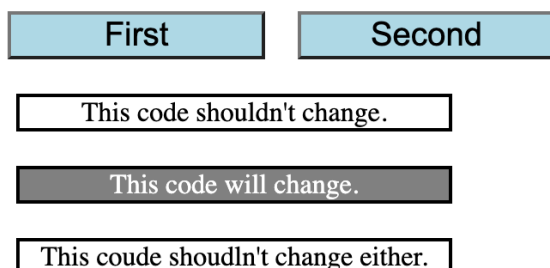
Our first example will deal with changing the second button with JavaScript. The JavaScript looks for the `id` and then changes the idea with the function defined in the `<script>`.


```

1  <head>
2  <script>
3      function clickedButton(buttonNum)
4      {
5          document.getElementById("stuff").innerHTML =
6          "Clicked " + buttonNum + " button."
7      }
8  </script>
9  <style>
10     button
11     {
12         width:150px;
13         height: auto;
14         background-color: lightblue;
15         margin-left: 15px;
16         font-size: 120%;
17     }
18     p
19     {
20         text-align: center;
21         border: 2px solid black;
22         margin:20px;
23         width:250px;
24     }
25     #stuff
26     {
27         text-align: center;
28         color:white; /*color defines text*/
29         background-color: grey;
30     }
31 </style>
32 </head>
33 <body>
34     <h1>Modify the DOM</h1>
35     <button onclick="clickedButton('First')">First</button>
36     <button onclick = "clickedButton('Second')">Second</button>
37     <p>This code shouldn't change.</p>
38     <p id = "stuff">This code will change.</p>
39     <p>This coude shoudln't change either.</p>
40 </body>

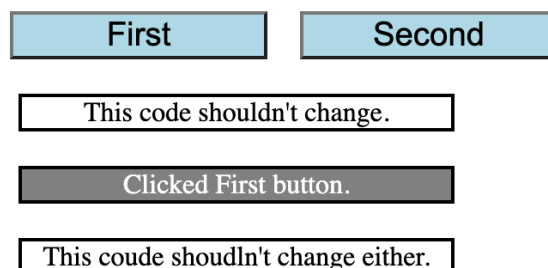
```

Modify the DOM



Before

Modify the DOM



After

Example 11: Using JS to Change CSS

Our next example shows the power of JavaScript to modify CSS. When clicking on the button marked "Open!" the JavaScript will run and go to the CSS and change it to hide to paragrah.

```

1     <script>
2     function clickEvent(buttonNum)
3     {
4         var buttonStatus;
5         if(buttonNum == "two")
6         {
7             buttonStatus = document.getElementById("demo");
8             // buttonStatus.style.display="none"; To edit directly
9             buttonStatus.className="close";
10            //Above chooses the class from stylesheet.
11        }
12        if(buttonNum == "one")
13        {
14            buttonStatus = document.getElementById("demo");
15            // buttonStatus.style.display="block";
16            buttonStatus.className="open";
17        }
18        console.log(buttonNum);
19        console.log(buttonStatus);
20    }
21 </script>
22 <style>
23     button
24     {
25         background-color: lightblue;
26         margin-left : 15px;
27         font-size: 125%;
28         width: 150px;
29         height: auto;
30     }
31     #demo
32     {
33         /*# is for id's.*/
34         color: white;
35         background-color: grey;
36         padding: 5px;
37     }
38     .open
39     {
40         /*. is used for classes.*/
41         display: block;
42     }
43     .close
44     {
45         display: none;
46     }
47 </style>
48 <body>
49     <h1>Changing the Style</h1>
50     <p>JavaScript can change the style of an HTML element.</p>
51     <button onclick = "clickEvent('one')">Open!</button>
52     <button onclick = "clickEvent('two')">Closed!</button>
53     <p id = "demo">Extra details... You can open and close this paragraph
54 </body>

```

Changing the Style

JavaScript can change the style of an HTML element.

Open!

Closed!

Extra details... You can open and close this paragraph

Before

Changing the Style

JavaScript can change the style of an HTML element.

Open!

Closed!

After

Lecture 11: The Keyword `this`

When using `this` in JavaScript, JS knows which tag you're talking about. When using `this` we want to use that certain tag when using our JavaScript. A simple example of using the keyword `this` with an anchor tag.

```
<a onclick = "jsFunction('this')">
```

There will be two examples that show how to use `this`.

Example 12: Using `this` To Get ID Element

```
1  <head>
2      <style>
3      img
4      {
5          width: 200px;
6          height: auto;
7      }
8  </style>
9  <script>
10     function displayId(element)
11     {
12         console.log(element.id);
13         //When running this code and clicking on the div or img will
           display the id.
14     }
15 </script>
16 </head>
17 <body>
18 <img src = "https://s3-us-west-2.amazonaws.com/s.cdpn.io/389177/
           Colleen82.jpg"
19 alt = "Awesome 80's haircut." onclick="console.log(this.alt)">
20 <!-- this.alt returns us the string assigned to alt -->
21 <div onclick = "console.log(this.innerHTML)">Hi there, chchecking out a
           div.</div>
22 <hr>    <!-- This puts a line thru the page. -->
23 <img src = "https://s3-us-west-2.amazonaws.com/s.cdpn.io/389177/
           FamilyChristmas74.jpg"
24 alt = "Another Beautiful Picture" id = "id-1" onclick = "displayId(this)"
           >
25 <div onclick = "displayId(this)" id = "id-2">Another div.</div>
26 </body>
```

- Line 18 uses the console to display the `alt` text to the console screen.
- Line 19 is using the console to write what `this` is referencing to. What happens here we write to the console when this `<div>` is clicked.
- Line 24, when the image is clicked, the id associated with this tag is written to the console. The function is defined in the `<script>`.
- What is happening on Line 24 is the same for Line 25.

Example 13: Using `this` to Call on Certain Tags and Edit in HTML

```

1  <head>
2  <script>
3      function showProperties(element)
4      {
5          document.getElementById('message').innerHTML = element.alt;
6      }
7      function hover_message(element)
8      {
9          document.getElementById('message').innerHTML = "Hover over an
10             image.";
11     }
12 </script>
13 <style>
14     body
15     {
16         background-color: grey;
17         border: 2px solid black;
18     }
19     #message
20     {
21         line-height: 100px;
22         text-align: center;
23         width: 550px;
24         height: 100px;
25         border: 5px black solid;
26         margin: 0 auto; /*Used to center. */
27         margin-bottom: 10px;
28         background-color: blueviolet;
29         font-size: 150%;
30     }
31     .preview
32     {
33         width: 10%;
34         margin-left: 17%;
35         border: 10px solid black;
36     }
37     img
38     {
39         width: 95%;
40     }
41 </style>
42 </head>
43 <body>
44 <div id = "message">Hover over an image to display the alt text.</div>
45 <img class = "preview" alt = "Styling with a Bandana"
46 src = " https://s3-us-west-2.amazonaws.com/s.cdpn.io/389177/bacon.jpg"
47 onmouseover = "showProperties(this)"
48 onmouseleave="hover_message(this)">

```

```

49
50 <img class = "preview" alt = "With my Boy"
51 src = "https://s3-us-west-2.amazonaws.com/s.cdpn.io/389177/bacon2.JPG"
52 onmouseover="showProperties(this)"
53 onmouseleave="hover_message(this)">
54
55 <img class = "preview" alt = "Young Puppy"
56 src = "https://s3-us-west-2.amazonaws.com/s.cdpn.io/389177/bacon3.jpg"
57 onmouseover="showProperties(this)"
58 onmouseleave="hover_message(this)">
59 </body>
60 </html>

```

- The function `showProperties(element)` finds the id that is named `message`, and then changes that tag by using a dot operator to access the alt property. This function changes what the tag says by changing it to the alt name of the image.
- Function `hover_message(element)` finds the id that matches what is in single quotes, `message`, and replaces it with the text stated in the function.

Lecture 12: Arrays

We have been using commands to fetch only one node of the DOM at a time. With the power of arrays we can get a collection of nodes, from anywhere of zero nodes to many. An example of getting one node at a time, we would use something like: `document.getElementById('idName');` and it would fetch that node with that id name.

Using arrays is slightly different. Instead of using `getElement` we'll be getting many nodes so now it will be `getElements`, we just add an 's' to 'element'. The two uses you'll see is:

- `document.getElementsByTagName('tag');`
- `document.getElementsByClassName('class_name');`

To Declare an Array:

Declaring an array is simple in JavaScript. Just put the keyword `var` and then a string of choice, and always use `[]`.

Declaring Different Types of Arrays

```

1   var grades = [72,85,100,82,95];
2   //This is how you declare an array in JS.
3   var images = document.getElementsByClassName['imgs'];
4   //Getting every image by this class name stored in the images array.
5   var list_items = document.getElementsByTagName['li'];
6   //Finds all tags 'li' and stores them in this array.

```

Array Terminology

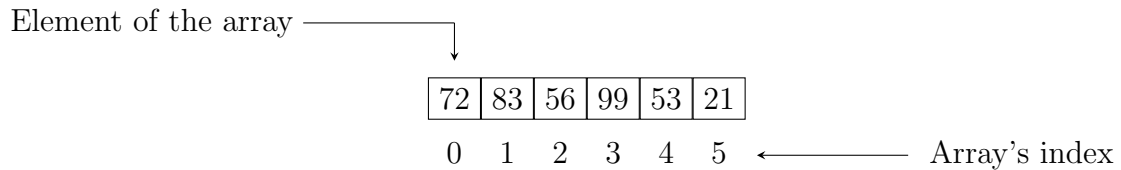


Figure 1.

Note: Every array will start at index 0.

Using the array in Figure 1, let's call this array `grades = [72,83,56,99,53,21];`, if we wanted to return the value that is at index 0, we would have to write: `grades[0];` and this would return the value: `72`.

Arrays in JavaScript don't have to be all the same type. You can have an array with integers, strings, and other datatypes. For example, `var info = [27,'Mike', 49104];`. This practice is not common, but good to know.

Arrays in JavaScript also have attributes and methods that can be used on the array because in JavaScript each array is an object. A list below shows some of the methods that can be used on the arrays. Let us use the array in Figure 1, and let's call this array `grades`.

- `grades.length` - This will return the length of the array. This would return 6.
- `grades.sort()` - `sort()` is a built in JavaScript command that will sort the array from greatest (starting at index 0) to least.
- `grades.push(element)` - This inserts an element at the very end of the array. Say we coded `grade.push(100);`, 100 would be at index 6.

We can combine these methods. Say we want to insert an element at the end of `grades`, but we don't know exactly how long the array is. We can perform: `grades[grades.length] = 84;` This would put 84 at index 6.

Lecture 13: Simple Examples Using Arrays

This code being shown is a very basic example of how to use arrays in JavaScript and add interaction between your site and the user. An overview of this code is that the page loads what is initially defined for the array and there is a button that you can click, JavaScript runs a prompt asking the user to input their favorite fruit. After doing that, the user's favorite fruit is added to the array by a function and this function writes to the HTML document.

Example 14: Asking User for Addition to Array

```
1  <head>
2  <title>Fruits Examples</title>
3  <script>
4      //Create the initial array.
5      var fruits = ['apple', ' orange', ' jack fruit', ' strawberry'];
6      function display_fruit()
7      {
8          //Display what is in the fruits array.
9          document.getElementById('fruits').innerHTML = fruits;
10     }
11     function add_fruit()
12     {
13         //Get users input.
14         var new_fruit = prompt("What is your favorite fruit? ");
15         //Add new fruit to the array.
16         fruits.push(new_fruit);
17         //Write the array to the screen.
18         document.getElementById('fruits').innerHTML = fruits;
19     }
20 </script>
21 </head>
22 <body onload="display_fruit()">
23     <button onclick="add_fruit()">Click to add your favorite fruit!
24     </button>
25     <p id = "fruits"></p>
26 </body>
```

Output:

Click to add your favorite fruit!

apple, orange, jack fruit, strawberry

Lecture 14: JavaScript Iteration

When talking about iteration in JavaScript or most coding languages we are talking about looping. We loop through a condition until our condition is met. Below is the syntax for a for loop in JavaScript.

For Loop Syntax:

```
1      for(i = 0; i < array.length(); i++)
2      {
3          //code goes here.
4      }
```

`i=0;` this is called the initializer, `i < array.length();` is the boolean condition, and `i++;` is the update/increment variable.

For Loop Diagram

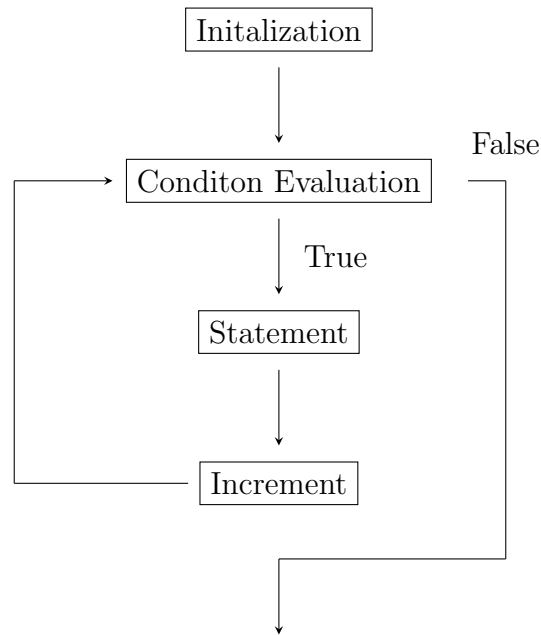


Figure 2.

When the code enters the boolean expression, if the code evaluates to true it will enter the "loop", if it evaluates to false, then the code will break out of the loop.

To write all the contents of an array we can do the following code:

```
1      for(index = 0; index < array.length; index++)
2      {
3          document.write(array[index]);
4          //This writes the contents of the array.
5      }
```

Example 15: Average of an Array Named Grades:

Assume we have an array named grades, and grades is : `grades = [83,82,10,25,100,93,96]`; and we want to find the average among this array. To do this we will perform the follow for loop:

```
1  <script>
2      var grades = [83,82,10,25,100,93,96]; //The array.
3      var sum = 0; //Sum of the total grades.
4      for(index = 0; index < grades.length; index++)
5      {
6          sum += grades[index]; //Adds each element to the total sum.
7      }
8      document.write(sum / grades.length); //Writes sum to HTML doc.
9  </script>
```

Output

69.85714285714286

Lecture 15: Flow of Control

Talking about flow of control we are talking about **if** statements. If statements are a boolean expression that if true will execute the code block, or commonly called block, if false will skip that block. Figure 3 shows the diagram of a if-statement.

Syntax For if Statements:

```
1     if(boolean-expression)
2     {
3         //statement.
4     }
```

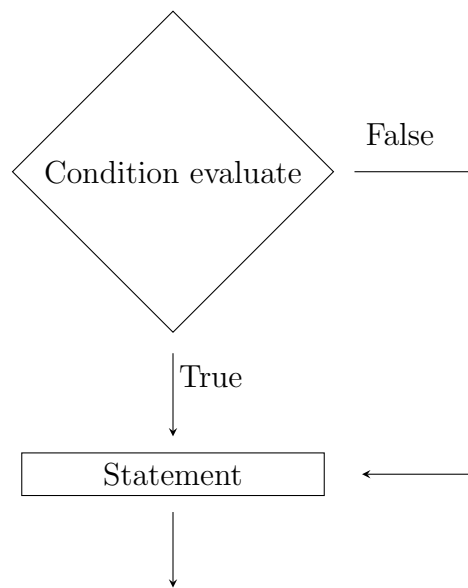


Figure 3.

While using if-statements, there is also an **else** keyword that if the if-statement doesn't execute then it will go to the else block where that code will execute. Shown if figure 4.

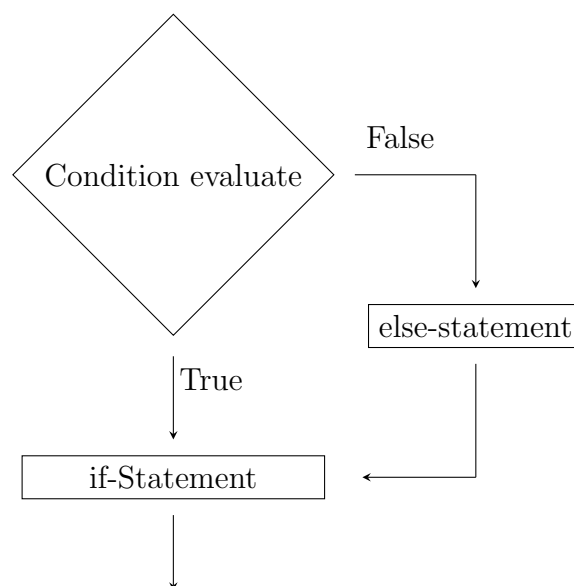


Figure 4.

Example 16: If-Else Statement In this example we will ask the user what their name is. If they enter in a string that is greater than 0, the `if` statement will execute, or if nothing is entered the `else` statement will execute.

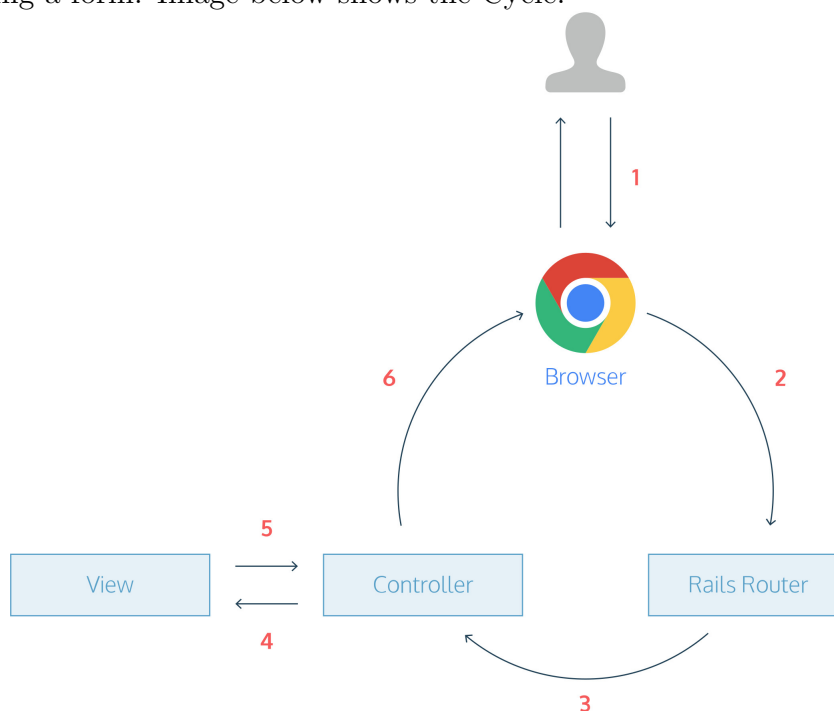
```
1      <script>
2      var name = prompt("What is your name");
3      if(name.length != 0)
4      {
5          document.write("Hello " + name + "!");
6          //This writes directly to the HTML doc.
7      }
8      if(name.length == 0)
9      {
10         document.write("Name entered was not greater than 0.");
11     }
12     </script>
```

Neat Fact: Assume you have a page with four paragraph tags. What is the proper JavaScript code to change the content of the second paragraph to "What does the Fox say?"

Ans: `document.getElementsByTagName('p')[1].innerHTML = "What does the Fox say?";`
You grab the tag and then the index starts at 0 so you go to index 1 and replace it with the current string.

Lecture 17: Forms

When people are working with forms, JavaScript is essential for the process. When we use forms we use what is called a "Request-Response Cycle." This cycle is when a computer/cellphone requests the HTML file from a server and the server responds with the HTML, CSS, and other files. When the computer/cellphone inserts making this request it can attach some attributes using a form. Image below shows the Cycle:



Forms

1. Forms are used to pass data to a server.
2. Different input elements can be used.
3. Forms must have a server with them.

To make a form you want to use three types of elements or tags of HTML:

1. `<form>`
2. `<label>` This matches the text on the screen with the input.
3. `<input>`

Form elements have attributes like: `type`, `name`, `id`. **Input Types:**

1. `input = 'text'` Lets you put any text into a box.
2. `input = 'email'` Brings up email format. On phone, the `@` appears on the phone's keyboard. Using `placeholder = 'me@me.com'` shows in a light print under the input box to give the user an idea of what kind of input we are looking for.
3. `input = 'password'` Makes the input not visible to the user.
4. `input = 'radio'` Puts buttons so you can only select one.
5. `input = 'checkbox'` Makes a checkbox where the user can select none to many.
6. `input = 'submit'` This is the button the user hits to submit the form.
7. `input = 'number'` Only lets the user type in numbers. By using `min` and `max` attributes, the user cannot exceed these limits.
8. `input = 'range'` Is a visible bar and can use the same min and max settings as number.
9. `input = 'color'` Color that a user can submit.
10. `input = 'date'` Puts a calendar up for the user to easily select a date.
11. `input = 'url'` Needs to have `http:` in order to be submitted.

Attributes for Input

1. `name` - Almost all input types should have a `name` attribute. This is needed for backend development.
2. `id` - This is used for labels, and to associate inputs to correct parts of the form. Also `id` is used for JavaScript.
3. `value` - Used for `button`, and `textfield` provides a default value that if not changed goes to the server.

Example 17: Simple Form Example

This example is of a simple form request. You see that the use of `for`, it specifies what the label is bound to. `placeholder` is used to show The user what kind of format the page is looking for. Note: for your button to say what you want it to, you must change the `value`.


```
1 <body>
2   <form>
3     <legend>Personal Information:</legend>
4     <label for = 'name'>Name</label>
5     <input type = "text" id = 'name' name = 'name'><br/>
6     <label>Email<input type = 'email' name = 'email' placeholder="
      me@me.com"></label><br/>
7     <label>Birth Date</label>
8     <input type = 'date' id = 'birthday' name = 'birthday' value = '
      birth'> </br>
9     <input type = 'submit' value="Click here to Submit">
10   </form>
11 </body>
```

Output of Example 17

Personal Information:

Name

Email

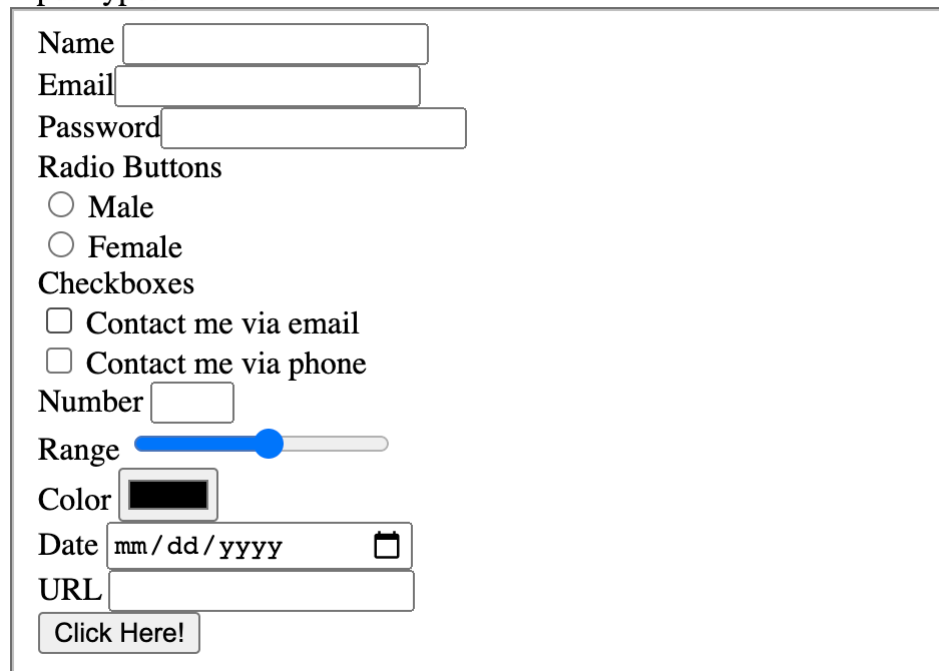
Birth Date 

Example 18: More Inputs on Forms

```
1  <body>
2    <legend>Input Types</legend>
3    <fieldset>
4      <label>Name <input type = 'text' id = 'name' name = 'name'></
        label>
5      <label>Email<input type = 'email' id = 'email' name = 'email'></
        label>
6      <label>Password<input type = 'password' id = 'password' name = '
        password'></label>
7      Radio Buttons </br>
8      <label>
9        <input type = 'radio' name = 'sex' value = 'male'>
10       Male
11     </label>
12     <label>
13       <input type = 'radio' name = 'sex' value = 'female'>
14       Female
15     </label>
16     Checkboxes</br>
17     <label>
18       <input type = 'checkbox' name = 'contact_email' value = '
        contact_email'>
19       Contact me via email
20     </label>
21     <label>
22       <input type = 'checkbox' name = 'contact_phone' value = "
        contact-phone">
23       Contact me via phone
24     </label>
25     <label>
26       Number <input type = 'number' id = 'number' name = 'number'
27       min = 0 max = 10>
28     </label>
29     <label>
30       Range <input type = 'range' id = "range" name = 'range'
31       min = 0 max = 15>
32     </label>
33     <label>
34       Color
35       <input type = 'color' id = 'color' name = 'color'>
36       <!-- Can put min and max on the input attribute but I don't
        know what they do... -->
37     </label>
38     <label>
39       Date
40       <input type = 'date' id = 'date' name = 'date'>
41     </label>
42     <label>
43       URL
44       <input type = 'url' name = 'url' id = url>
45     </label>
46     <label>
47       <!-- value defines what the box will say. -->
48       <input type = 'submit' value = 'Click Here!'>
49     </label>
50   </fieldset>
51 </body>
```

Output of Example 18:

Input Types



Name

Email

Password

Radio Buttons

☐ Male

☐ Female

Checkboxes

☐ Contact me via email

☐ Contact me via phone

Number

Range

Color

Date

URL

Lecture 18: Simple Validation Using HTML

Using HTML you can use element to validate inputs from the user. While using validation can help users make sure that if a site is asking for numbers and user puts in text it will not let the user proceed.

Input Attributes:

1. **required** - Halts the submit process if any of the required elements are empty. If you are testing your code and it keeps requiring you to satisfy the requirements, you can put `<form novalidate>`. Putting this in the form will let you bypass the requirements.
2. **pattern** - works with `input type = 'text'` and requires the input to have a specific form. Ex: `[0 - 9]{5}`, this is saying must use numbers 0 - 9 and there must be five spots taken up. Ex: `[0 - 9]{13 - 16}`, this is saying use numbers and it can be length of 13 to 16. Ex: `[a - zA - Z]+` means use lower and uppercase and use one character.

You can use limiting numbers. Things you can use are **min**, **max**, and **steps**. **steps** is saying things are only in increments; Ex: using **steps** of 5, you can only increase by 5's.

Lecture 19: Comparing Two Inputs

We want to compare inputs when the user is submitting information on a form. We want to make sure that the user put two of the same email or password combination in. Most of this is done with an example.

A new event we learn is `onchange`. We use `onchange` whenever we want to look at something when the user is done typing the information in.

Example 19: Comparing Two Emails:

```
1      <head>
2          <script>
3              function compare()
4              {
5                  var email1 = document.getElementById('email1');
6                  var email2 = document.getElementById('email2');
7                  // We are comparing the values, not the nodes!
8                  if(email1.value != email2.value)
9                  {
10                     console.log('Emails do not match');
11                     alert('Emails must be the same!');
12                     return false;
13                 }
14                 else
15                 {
16                     console.log('Emails do match');
17                     return true;
18                 }
19             }
20         </script>
21     </head>
22     <body>
23         <form>
24             <label>
25                 Preferred email address:
26                 <input type = 'email' name = 'email1' id = 'email1' required>
27             </label>
28             <br>
29             <label>
30                 Repeat enmail address:
31                 <input type = 'email' name = 'email2' required id = 'email2'
32                     onchange="compare();">
33             </label>
34             <br>
35             <!-- When adding a return value, it must be true inorder to proceed.
36                 Also have to add return values into JS. -->
37             <input type = 'submit' value = 'Confirm Emails!' onclick = "
38                 return check();">
39         </form>
40     </body>
```

Note: Notice in example 19 that we are comparing the value of the email node. When we are using these nodes we treat them as objects and we can access certain attributes associated with these nodes.

Lecture 20: Checkbox and Radiobutton

When we are dealing with checkboxes and radiobuttons it's good to know the difference between them. Checkbox can have zero or many boxes selected. Using radiobutton, you can only choose one at a time. With the examples presented ahead, pay attention that these two are grouped by the `name` attribute. This attribute needs to be consistant with what you wanted grouped together.

Example 20: Checkbox and Radiobuttons

We show how to use checkboxes and radiobuttons with this example. Looking at what we have, we see the `name` is consistent with each grouping.

```
1      <body>
2      <!-- Both of these forms need the same name! -->
3      <form>
4      <p>Favorite Foods</p>
5      <label><input type = 'checkbox' value = 'pizza' name = 'food'>Pizza</
        label></br>
6      <label><input type = 'checkbox' value = 'chips' name = 'food'>Chips</
        label></br>
7      <label><input type = 'checkbox' value='kale' name = 'food'>Kale</
        label></br>
8      <input type="submit" value = "Submit!">
9      </form>
10     <hr>
11     <!-- Notice that the name groups these forms together. -->
12     <form>
13         <label><input type = 'radio' value = 'male' name = 'gender'>Male<
            /label></br>
14         <label><input type = 'radio' value = 'female' name = 'gender'>
            Female</label></br>
15         <input type = 'submit' value = 'Submit!!!'>
16     </form>
17 </body>
```

Example 21: Form Verification

With this form verification we are checking if the two emails are the same. Notice when we getting the `id` we are getting the `value` attribute! Because when getting an element by its `id` we are getting more information than what we want. So we want to access the value that is stored in the node. Example 21 will be on page 25.


```

1     <head>
2         <script>
3             function nicknameFunction()
4             {
5                 //We see if the checkbox is checked.
6                 if(document.getElementById('yes_nickname').checked)
7                 {
8                     //Changes the style for this id to make it appear.
9                     document.getElementById('nick').style.display='inline';
10                    //Force user to put nickname in.
11                    document.getElementById('nickname').setAttribute('
                        required',true);
12                }
13                else
14                {
15                    document.getElementById('nickname').removeAttribute('
                        required');
16                    document.getElementById('nick').style.display='none';
17                }
18            }
19        </script>
20        <style>
21            #nick
22            {
23                /* Hides as a default. */
24                display: none;
25            }
26        </style>
27    </head>
28    <body>
29        <fieldset>
30            <legend>Form Verification</legend>
31            <label for = 'first_name'>First Name:</label>
32            <input type = 'text' name = 'first_name' id = 'first_name'
                required></br>
33            <label for = 'last_name'>Last Name:</label>
34            <input type = 'text' name = 'last_name' id = 'last_name' required
                ></br>
35            <label for = 'yes_nickname'>Nickname?</label>
36            <input type = 'checkbox' name = 'yes_nickname' id = 'yes_nickname'
                ' value = 'yes' onclick="nicknameFunction()"></br>
37
38            <div id = 'nick'>
39                <label for = 'nickname'>Nickname:</label>
40                <input type = 'text' id = 'nickname' name = 'nickname'></br>
41            </div>
42            <input type = 'submit' value = 'Verify'>
43
44
45        </fieldset>
46    </body>

```
