**DAY 3: Assignment**

1. **The ‘For’ Loop**

The For Loop comes first because of its simplicity and ease of use. It is a very user-friendly kind of loop that runs with a method of using a counter.

The value is first set with an appropriate condition, which is also called ‘initializing a loop’. Next, the terminal or final value is specified. The For Loop makes things really easy when you need to run a set of codes multiple times.

#### **Using an array**

*var numbers = [ 10, 20, 30, 40, 50]   
for (var i=0; i < numbers.length; i++) {  
   console.log(numbers[i])  
}*

### The ‘For In’ Loop

Another way of looping is the For In Loop. Unlike the For Loop, this loop won’t be using a counter. So this makes the whole process even more simple and hassle-free. In fact, the For In Loop is essentially a simplified version of the For Loop.

#### **Looping through an Object Property**

*var person = {  
   fname: "Nick",  
   lname: "Jonas",  
   age: 26  
};   
for (let x in person) {  
   console.log(x + ": "+ person[x])  
}* **‍**

1. **The ForEach() Loop**

This method is used for looping through an array element. Here’s an example of this:

*var names = ["jerry", "tom", "pluto", "micky", "mini"];  
names.forEach(function1);  
function function1(currentValue, index) {  
   console.log("Index in array is: "+index + " ::  Value is: "+currentValue);  
}* **‍**

# Difference between window,screen and document in javascript

# Window

Each browser tab has its own top-level window object. Each <iframe> (and deprecated <frame>) element has its own window object too, nested within a parent window. Each of these windows gets its own separate global object. window.window always refers to window, but window.parent and window.top might refer to enclosing windows, giving access to other execution contexts. In addition to document and screen described below, window properties include

* setTimeout() and setInterval() binding event handlers to a timer
* location giving the current URL
* history with methods back() and forward() giving the tab's mutable history
* navigator describing the browser software

# JavaScript: window, document and screen

## Window

The JavaScript **window object** sits at the top of the JavaScript Object hierarchy and represents the browser window. The window object is supported by all browsers. All global **JavaScript objects** , functions, and variables automatically become members of the window object. The window is the first thing that gets loaded into the **browser** . This window object has the majority of the properties like length, innerWidth, innerHeight, name, if it has been closed, its parents, and more.

The window object represents the current **browsing context** . It holds things like window.location, window.history, window.screen, window.status, or the **window.document** . Each browser tab has its own top-level window object. Each of these windows gets its own separate global object. window.window always refers to window, but **window.parent** and window.top might refer to enclosing windows, giving access to other execution contexts. Moreover, the window property of a window object points to the window object itself. So the following ststements all return the same window object:

* **Document**

Each window object has a document object to be rendered. These objects get confused in part because HTML elements are added to the global object when assigned a unique id. E.g., in the HTML snippet

<body>

<p id="holyCow"> This is the first paragraph.</p>

</body>

the paragraph element can be referenced by any of the following:

* window.holyCow or window["holyCow"]
* document.getElementById("holyCow")
* document.querySelector("#holyCow")
* document.body.firstChild
* document.body.children[0]

## Document

The **Document interface** represents any web page loaded in the browser and serves as an entry point into the web page's content, which is the DOM tree. When an HTML document is loaded into a **web browser** , it becomes a document object. It is the root node of the HTML document. The document actually gets loaded inside the window object and has properties available to it like title, URL, cookie, etc. HTML documents, served with the **"text/html"** content type, also implement the HTMLDocument interface, whereas XML and SVG documents implement the XMLDocument interface.

# Screen

The window object also has a screen object with properties describing the physical display:

* screen properties width and height are the full screen
* screen properties availWidth and availHeight omit the toolbar

The portion of a screen displaying the rendered document is the **viewport** in JavaScript, which is potentially confusing because we call an application's portion of the screen a window when talking about interactions with the operating system. The getBoundingClientRect() method of any document element will return an object with top, left, bottom, and right properties describing the location of the element in the viewport.

## Screen

Screen is a small information object about physical **screen dimensions** . It can be used to display screen width, height, colorDepth, pixelDepth etc. It is not mandatory to write **window prefix** with screen object. It can be written without window prefix.