Java Script

For Beginners

Variables

Variables

- A variable is defined as a container that is used to hold values.
- declared using the "var" keyword.
- let and const.

```
1    var x;
2    var y;
3    var z;
```

Assignment & Initializing

- We use the assignment operator(=).

```
1    var x = 1;
2    var y = 2;
3    var z = 3;
```

- Declaring a variable with values is called initializing

```
1    var x = 1;
2    var y = 2;
3    var z = 3;
4
5    x = 100;
```

Data Types

- To store in variables single-precision numbers, double-precision numbers, strings, boolean values, objects, date . . .
- Primitive Data Types
  - Number, String, Boolean, Undefined

```
1    var num = 100;
2    var str = "Hello World";
3    var bool = true;
4    var und = undefined;
```

NON - Primitive Data types

- Object, Array, Date

```
1    var obj = {
2        str : "Hello World"
3    }
4
5    var date = new Date();
6
7    var arr = [1,2,3,4,5]
```

Function

Function - Declaring

- Functions are called the main building blocks of a program.
- They provide code reusability and helps reducing time and effort.
- To create a function in JavaScript, we need the '**function**' keyword .

```
1    function demoFunction(){
2
3        console.log("This is a function");
4
5    }
```

- A function in JavaScript is declared using the 'function' keyword followed by the name of the function and two parentheses.

Calling a function

- Declaring a function is not enough. Nothing will happen until the function is called.

```
1   function demoFunction(){
2
3       console.log("This is a function")
4
5   }
6
7   demoFunction();
```

Function - Objective

- The main objective of using functions is to avoid code repetition and offer better code reusability.

```
1   function demoFunction(){
2
3       console.log("This is a function")
4
5
6   }
7
8   demoFunction();
9   demoFunction();
10  demoFunction();
```

Parameters

- The data is passed as parameters or also known as arguments.

```
1   function add(a,b){
2
3       console.log("Sum of a and b is: ", a+b);
4   }
```

```
1    function add(a,b){
2
3        console.log("Sum of a and b is: ", a+b);
4    }
5
6    add(10,20);
```

Variables to a function as parameters

```
1   function add(a,b){
2
3       console.log("Sum of a and b is: ", a+b);
4   }
5
6   var a = 10;
7   var b = 20;
8
9   add(a,b);
10
11  add(b,a);
```

DOM

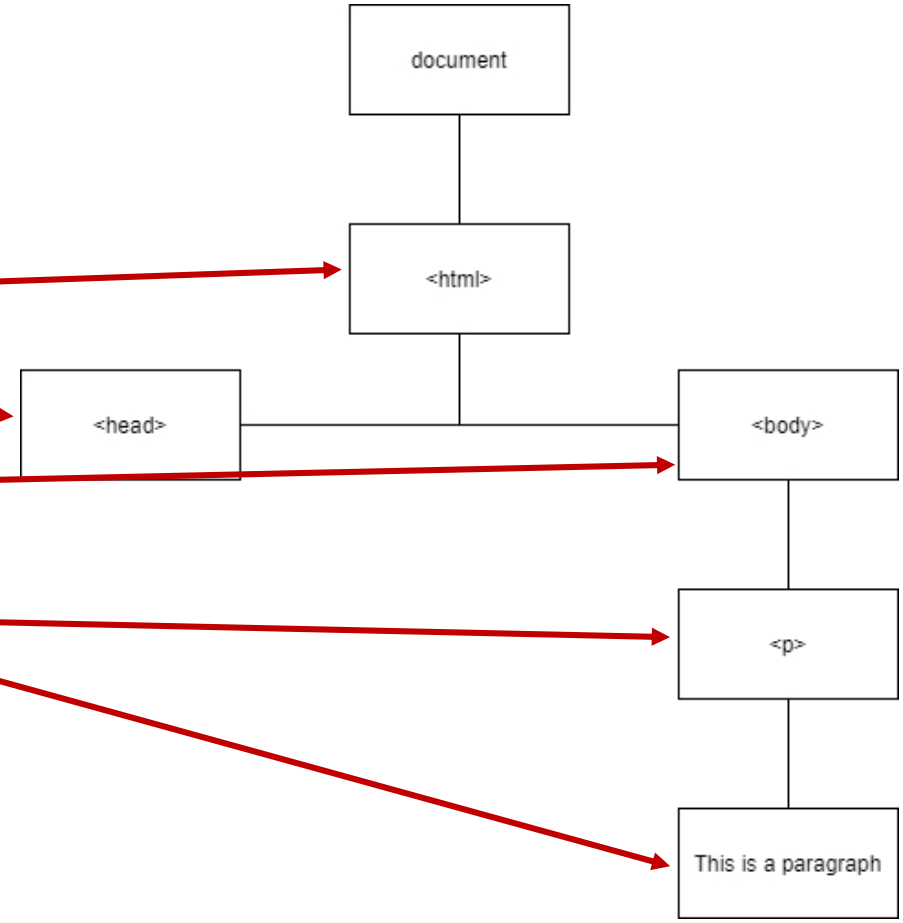
Document Object Model

- When the HTML file is loaded into a browser, a tree-like structure is created.
- This structure has various nodes , and these nodes represent various elements of the HTML document.

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4       </head>
5       <body>
6
7           <p>
8               This is a paragraph.
9           </p>
10
11      </body>
12  </html>
```

Tree Structure

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4       </head>
5       <body>
6
7           <p>
8               This is a paragraph.
9           </p>
10
11      </body>
12  </html>
```



DOM Summary

- DOM is a tree-like structure that is created when an HTML document is loaded in a browser.
- Every node of a DOM tree represents an HTML element.
- DOM can be manipulated to make dynamic changes in an HTML document.

HTML Events

HTML events

- HTML events are attributes that are used to make something happen.
- For example, a button click popping a message. Another example is, popping a message when page loads or when the input changes.
- HTML events are very important because they are used to convert the static HTML elements into dynamic.
- One of the most important uses of these events is that JavaScript functions can be triggered using them.
- Although a very basic DOM manipulation can be done using HTML, serious manipulation is done using functions.

Triggering alert()

- The alert() function is pop up that appears on the screen with a message.
- Mouse events are one of the most commonly used HTML events.
- These events are triggered when the user does something with the mouse.
- For example, clicking on something, double-clicking on something, hovering over something, and many more.

Triggering alert()

- The onclick event is the most basic HTML event. As the name suggests, this event triggers something when an element is clicked on.

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4       </head>
5       <body>
6
7           <button onclick="alert('You clicked on the button')">
8               Click here
9           </button>
10
11      </body>
12  </html>
```

- Remember, the value of an HTML event is always written inside quotes.

Triggering a function

- In the real-time, events are used to trigger JavaScript functions.

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4       </head>
5       <body>
6
7           <button onclick="message()">
8               Click here
9           </button>
10
11      </body>
12      <script>
13          function message(){
14
15              alert('You clicked on the button')
16
17          }
18      </script>
19  </html>
```

- The function is placed inside the <script> tag

Commonly used HTML events

- Mouse events
  - onclick - triggers on the single click of the mouse.
  - ondblclick - triggers on the double click of the mouse.
  - onmouseover - triggers when the mouse moves over an element.
  - onwheel - triggers when the wheel of the mouse moves over an element.
- Keyboard events
  - onkeydown - triggers when a key is being pressed.
  - onkeypress - triggers when a key is pressed.
  - onkeyup - triggers when a key is released.
- Window events
  - onload - triggers when a window is completely loaded.
  - onunload - triggers when a window is closed.
  - onresize - triggers when a window is resized.

Commonly used HTML events

- Form events
  - onchange - triggers when the value of an element is changed.
  - onsumbit - triggers when a form is submitted.
  - onreset - triggers when a form is reset.
- Drag events
  - ondrag - triggers when an element is dragged.
  - ondrop - triggers when an element that is being dragged is dropped.

Summary

- HTML events make something happens. They are used just like other attributes.
- Different types of HTML events are mouse, keyboard, drag, window, media, and form events.
- The value of an event is written inside the quotes. What will happens next depends on the value.
- Generally, functions are triggered using HTML events because they can have multiple lines of codes inside them.

Find HTML elements

Finding elements

- finding elements using various methods and the innerHTML property.
- The innerHTML property is used to get the access to content of the HTML elements.
- Ways to find HTML elements
  - document.getElementById()
  - document.getElementsByTagName()
  - document.getElementsByClassName()

document.getElementById()

- The document.getElementById() method is the most common way to find HTML elements.
- The value passed to the document.getElementById() method is the id of the element that we want to find.

```html
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <p id = "para" onclick="demo()">
            This is a paragraph.
        </p>

    </body>

    <script>
        function demo(){

            var ele = document.getElementById("para");
            console.log(ele)
        }
    </script>
</html>
```

To access the content, we will use the innerHTML property.

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4       </head>
5       <body>
6
7           <p id = "para" onclick="demo()">
8               This is a paragraph.
9           </p>
10
11      </body>
12
13      <script>
14          function demo(){
15
16              var ele = document.getElementById("para").innerHTML;
17              console.log(ele)
18          }
19      </script>
20  </html>
```

document.getElementsByTagName()

- tag name.

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <p>
            This is a paragraph.
        </p>
        <p>
            This is a paragraph.
        </p>
        <p>
            This is a paragraph.
        </p>

        <button onclick="demo()">
            Click here
        </button>

    </body>

    <script>
        function demo(){

            var ele = document.getElementsByTagName("p");
            console.log(ele)
        }
    </script>
</html>
```

document.getElementsByTagName()

- There are three <p> tags and document.getElementsByTagName() will find all of these.
- The document.getElementsByTagName() returns an array with all the elements.
- Similarly, the innerHTML property will also return an array with all the content.

document.getElementsByClassName()

- It uses class name to find elements.
- Similar to the document.getElementsByTagName() method, the document.getElementsByClassName() method is also used to find multiple elements.
- It also returns an array.

```html
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <p class = "para">
            This is a paragraph.
        </p>
        <p class = "para">
            This is a paragraph.
        </p>
        <p class = "para">
            This is a paragraph.
        </p>

        <button onclick="demo()">
            Click here
        </button>

    </body>

    <script>
        function demo(){

            var ele = document.getElementsByClassName("para");
            console.log(ele)
        }
    </script>
</html>
```

Summary

- The innerHTML property is used to get the content of an element.
- There are three ways to find elements in HTML - document.getElementById(), document.getElementsByTagName(), and document.getElementsByClassName().
- The document.getElementById() method finds an element using the id of that element.
- The document.getElementsByTagName() method find elements using the tag name.
- The document.getElementsByClassName() method find elements using the class of the elements.

Content and CSS with JavaScript

Changing content using innerHTML property

Changing values of the attributes

Changing CSS

Summary

Changing content using innerHTML property

- How the methods like document.getElementById(), document.getElementsByTagName() can be combined with innerHTML property to access an element's content.
- The main usage of this property is to change the content
- **Example:**
- It is a dynamic HTML page.
- The page has a button and a paragraph.
- Clicking on this button will replace the text of the paragraph.

```html
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <button onclick="demo()">
            Click here
        </button>

        <p id="para">
            This text will be replaced.
        </p>

    </body>

    <script>
        function demo(){

            var ele = document.getElementById("para");

            ele.innerHTML = "New text!";

        }
    </script>
</html>
```

```html
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <button onclick="demo()">
            Click here
        </button>

        <p>This is a paragraph</p>
        <p>This is a paragraph</p>
        <p>This is a paragraph</p>

    </body>

    <script>
        function demo(){

            var ele = document.getElementsByTagName("p");

            ele[0].innerHTML = "Text replaced!";

        }
    </script>
</html>
```

Changing values of the attributes

- It is also possible to change the value of an attribute with JavaScript. There is no special property to change an attribute's value.
- We can simply use the property name after finding the element and assign it a new value.
- Suppose, there an image on the page and When clicked on it, a new image replaces the old one.
- To do this, we need to assign a new value to the src attribute of the <img> tag.

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <img
        src="./images/forest.jpg"
        width="200"
        height="200"
        onclick="demo()"
        id="image"
        >

    </body>

    <script>
        function demo(){

            var ele = document.getElementById("image");
            ele.src = "./images/mountain.jpg";

        }
    </script>
</html>
```

Changing CSS

- Not only the content and attribute values, but we can also change the CSS with JavaScript.
- First, we have to access the style attribute of the element and then, apply the required CSS property to it with the value.
- Suppose, there is a paragraph whose color is red, and when clicked on it, the color changes to blue.

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <p id="para" style="color : red" onclick="demo()">
            This is a paragraph.
            This is a paragraph.
            This is a paragraph.
            This is a paragraph.
            This is a paragraph.
        </p>

    </body>

    <script>
        function demo(){

            var ele = document.getElementById("para");

            ele.style.color = "blue";

        }
    </script>
</html>
```

Summary

- The innerHTML property is used to change the content of an HTML element.
- To change the value of an attribute, use the attribute name, and assign the new value.
- The CSS can be changed by using the style attribute and then applying the property with the new value.

Creating and removing elements

document.createElement()

document.createTextNode()

appendChild()

insertBefore()

document.createAttribute()

setAttributeNode()

remove()

Create element

- The document.createElement() method is used to create an HTML element.

- In the next example, the document.createElement() method is used to create a <p> element.
- The element name should be passed to this method. Now, we have a <p> tag but there is no content in it yet.
- The next step is to use the document.createTextNode() method to create a text node.

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <div id="main">
            <button onclick="demo()">Click here!</button>

        </div>

    </body>

    <script>
        function demo(){

            var paragraph =  document.createElement("p");

            var text = document.createTextNode("This is a paragaph");

        }
    </script>
</html>
```

appendChild()

- In the last example, we create two nodes - <p> tag and the text.
- We have to append the <p> to the HTML document, but first, we need to append the text node to the newly create <p> tag.
- The appendChild() method appends a new node as the last child node.
- So let's append the text node to the <p> tag and then, we will append this whole element as one of the last child nodes in the HTML document

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <div id="main">
            <button onclick="demo()">Click here!</button>

        </div>

    </body>

    <script>
        function demo(){

            var paragraph =  document.createElement("p");

            var text = document.createTextNode("This is a paragaph");

            paragraph.appendChild(text);

        }
    </script>
</html>
```

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <div id="main">
            <button onclick="demo()">Click here!</button>

        </div>

    </body>

    <script>
        function demo(){

            var paragraph =  document.createElement("p");

            var text = document.createTextNode("This is a paragaph");

            paragraph.appendChild(text);

            var div = document.getElementById("main");

            div.appendChild(paragraph)

        }
    </script>
</html>
```

Add a Button using innerHTML property

- Let's discuss another example, this time using the innerHTML property.
- In the example, a button is created using the document.createElement() method and then, innerHTML property is used on it to assign a value.

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <div id="main">
            <button onclick="demo()">Click here!</button>

        </div>

    </body>

    <script>
        function demo(){

            var button =  document.createElement("button");

            button.innerHTML = "New button"

        }
    </script>
</html>
```

```html
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <div id="main">
            <button onclick="demo()">Click here!</button>

        </div>

    </body>

    <script>
        function demo(){

            var button =  document.createElement("button");

            button.innerHTML = "New button"

            var div = document.getElementById("main");

            div.appendChild(button)

        }
    </script>
</html>
```

insertBefore()

- The insertBefore() method is used to insert an element right before an existing element.
- Suppose, we have a <p> tag and we want to add a <h1> tag before that <p> tag.

```html
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <div id="main">
            <button onclick="demo()">Click here!</button>
            <p id="para"> This is a paragraph.</p>
        </div>

    </body>

    <script>
        function demo(){

            var heading =  document.createElement("h1");
            var text = document.createTextNode("Heading");

            heading.appendChild(text);

            var div = document.getElementById("main");
            var paragraph = document.getElementById("para");

            div.insertBefore(heading, paragraph)

        }
    </script>
</html>
```

document.createAttribute()

- We can also create a new attribute for an existing element as well as for a newly created element.

- Suppose, there is a paragraph and on the button click, a class attribute with CSS is applied to it.

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            .css {
                font-size: 50px;
                color : blue;
            }
        </style>
    </head>
    <body>

        <button onclick="demo()">Click here!</button>
        <p id="para"> This is a paragraph.</p>

    </body>
    <script>
        function demo(){

            var paragraph = document.getElementById("para")

            var attribute = document.createAttribute("class");
            attribute.value = "css";

        }
    </script>
</html>
```

setAttributeNode()

- As of now, we have created an attribute but it is not added the <p> tag.
- To do this, we need to use the setAttributeNode() method.

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            .css {
                font-size: 50px;
                color : blue;
            }
        </style>
    </head>
    <body>

        <button onclick="demo()">Click here!</button>
        <p id="para"> This is a paragraph.</p>

    </body>
    <script>
        function demo(){

            var paragraph = document.getElementById("para")

            var attribute = document.createAttribute("class");
            attribute.value = "css";

            paragraph.setAttributeNode(attribute);
        }
    </script>
</html>
```

remove()

- The remove() method is used to remove an element from an HTML document.

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

        <button onclick="demo()">Click here!</button>
        <p id="para"> This is a paragraph.</p>

    </body>

    <script>
        function demo(){

            var paragraph = document.getElementById("para");

            paragraph.remove();

        }
    </script>
</html>
```

Summary

- The **`document.createElement()`** method is used to create a new element.
- To create a text node, use **`document.createTextNode()`** method and to assign a value to an element,
- use the innerHTML property with **`document.createElement()`** method.
- The **`appendChild()`** method is used to append a new element as the last child node of an existing node.

Summary

- The **`insertBefore()`** method is used to insert a new element right before an existing element.
- A new attribute can be created using the **`document.createAttribute()`** method while its value can be assigned using the **`setAttributeNode()`** method.
- To remove an existing element, use the **`remove()`** method.

Баярлалаа!

Бүгдийг нэг бүрчилэн ажиллуулж туршиж үзээрэй!