|  |  |
| --- | --- |
| Roll No | A-75 |
| Student Name | Prasad Kashid |
| Subject | IP |
| Date of Experiment |  |
| Date of Submission |  |

**EXPERIMENT NO. 5**

**AIM:** Create interactive webpage using basic JavaScript.

**OBJECTIVE:**

To orient students to JavaScript for developing interactive webpage.

**LAB OUTCOMES:**

LO1 – Identify and apply the appropriate HTML tags to develop a webpage.

LO2 – Identify and apply the appropriate CSS tags to format data on webpage.

**THEORY:**

**JavaScript** is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform. JavaScript is the most popular programming language in the world and that makes it a programmer’s great choice.

**Applications of Javascript Programming**

**Client side validation** - This is really important to verify any user input before submitting it to the server and Javascript plays an important role in validting those inputs at front-end itself.

**Manipulating HTML Pages** - Javascript helps in manipulating HTML page on the fly. This helps in adding and deleting any HTML tag very easily using javascript and modify your HTML to change its look and feel based on different devices and requirements.

**User Notifications** - You can use Javascript to raise dynamic pop-ups on the webpages to give different types of notifications to your website visitors.

**Back-end Data Loading** - Javascript provides Ajax library which helps in loading back-end data while you are doing some other processing. This really gives an amazing experience to your website visitors.

**Server Applications** - Node JS is built on Chrome's Javascript runtime for building fast and scalable network applications. This is an event based library which helps in developing very sophisticated server applications including Web Servers.

**Syntax:**

<script ...>

JavaScript code

</script>

**Variables:**

Variables are containers for storing data (values). JavaScript uses the keywords var, let and const to declare variables. An equal sign is used to assign values to variables.

**Variable Scope:**

The scope of a variable is the region of your program in which it is defined.

JavaScript variables have only two scopes.

* **Global Variables** − A global variable has global scope which means it can be defined anywhere in your JavaScript code.
* **Local Variables** − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

**Datatypes:**

One of the most fundamental characteristics of a programming language is the set of data types it supports. These are the type of values that can be represented and manipulated in a programming language.

JavaScript allows you to work with three primitive data types −

**●**

**●**

**●**

**Numbers,** eg. 123, 120.50 etc.

**Strings** of text e.g. "This text string" etc.

**Boolean** e.g. true or false.

**Operators:**

JavaScript supports the following types of operators.

* Arithmetic Operators
* Comparison Operators
* Logical (or Relational) Operators
* Assignment Operators
* Conditional (or ternary) Operators

**Javascript Conditionals:**

Conditional statements control behavior in JavaScript and determine whether or not pieces of code can run.

There are multiple different types of conditionals in JavaScript including:

“If” statements: where if a condition is true it is used to specify execution for a block of code.

“Else” statements: where if the same condition is false it specifies the execution for a block of code.

“Else if” statements: this specifies a new test if the first condition is false.

**Javascript Loops:**

The **JavaScript loops** are used *to iterate the piece of code* using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

In JavaScript we have the following looping statements:

* while - loops through a block of code while a condition is true
* do...while - loops through a block of code once, and then repeats the loop while a condition is true
* for - run statements a specified number of times

**Javascript Functions:**

A JavaScript function is a block of code designed to perform a particular task.A JavaScript function is executed when "something" invokes it (calls it).

A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:

**(parameter1, parameter2, ...)**

The code to be executed, by the function, is placed inside curly brackets: {}

**Syntax of Function:**

function name(parameter1, parameter2, parameter3) { // code to be executed

}

**Function Invocation:**

The code inside the function will execute when "something" **invokes** (calls) the function:

* When an event occurs (when a user clicks a button)
* When it is invoked (called) from JavaScript code
* Automatically (self invoked)

**Function Return:**

When JavaScript reaches a return statement, the function will stop executing.If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.Functions often compute a return value.

**Use of Function:**

You can reuse code: Define the code once, and use it many times.

You can use the same code many times with different arguments, to produce different results.

**Javascript as Object Oriented Programming Language:**

JavaScript is an Object Oriented Programming (OOP) language. A programming language can be called object-oriented if it provides four basic capabilities to developers −

* **Encapsulation** − the capability to store related information, whether data or methods, together in an object.
* **Aggregation** − the capability to store one object inside another object.
* **Inheritance** − the capability of a class to rely upon another class (or number of classes) for some of its properties and methods.
* **Polymorphism** − the capability to write one function or method that works in a variety of different ways.

**Object Properties:**

Object properties can be any of the three primitive data types, or any of the abstract data types, such as another object. Object properties are usually variables that are used internally in the object's methods, but can also be globally visible variables that are used throughout the page.

SYNTAX: objectName.objectProperty = propertyValue;

**Object Methods:**

Methods are the functions that let the object do something or let something be done to it. There is a small difference between a function and a method – at a function is a standalone unit of statements and a method is attached to an object and can be referenced by **this** keyword.

**Javascript Events:**

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

Events are a part of the Document Object Model (DOM) Level 3 and every HTML element contains a set of events which can trigger JavaScript Code.

**1) onclick Event Type**

This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.

**2) onsubmit Event Type**

onsubmit is an event that occurs when you try to submit a form. You can put your form validation against this event type.

**3) onmouseover and onmouseout**

These two event types will help you create nice effects with images or even with text as well. The onmouseover event triggers when you bring your mouse over any element and the onmouseout triggers when you move your mouse out from that element.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Value** | **Description** |
|  |  |  |
| Offline | script | Triggers when the document goes offline |
|  |  |  |
| Onabort | script | Triggers on an abort event |
|  |  |  |
| onafterprint | script | Triggers after the document is printed |
|  |  |  |
| onbeforeonload | script | Triggers before the document loads |
|  |  |  |
| onbeforeprint | script | Triggers before the document is printed |
|  |  |  |
| onblur | script | Triggers when the window loses focus |
|  |  |  |
| oncanplay | script | Triggers when media can start play, but might has to |
| stop for buffering |
|  |  |
|  |  |  |
| oncanplaythrough | script | Triggers when media can be played to the end, without |
| stopping for buffering |
|  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| onchange | script | Triggers when an element changes |
|  |  |  |
| onclick | script | Triggers on a mouse click |
|  |  |  |
| oncontextmenu | script | Triggers when a context menu is triggered |
|  |  |  |
| ondblclick | script | Triggers on a mouse double-click |
|  |  |  |
| ondrag | script | Triggers when an element is dragged |
|  |  |  |
| ondragend | script | Triggers at the end of a drag operation |
|  |  |  |
| ondragenter | script | Triggers when an element has been dragged to a valid |
| drop target |
|  |  |
|  |  |  |
| ondragleave | script | Triggers when an element is being dragged over a valid |
| drop target |
|  |  |
|  |  |  |
| ondragover | script | Triggers at the start of a drag operation |
|  |  |  |
| ondragstart | script | Triggers at the start of a drag operation |
|  |  |  |
| ondrop | script | Triggers when dragged element is being dropped |
|  |  |  |
| ondurationchange | script | Triggers when the length of the media is changed |
|  |  |  |
| onemptied | script | Triggers when a media resource element suddenly |
| becomes empty. |
|  |  |
|  |  |  |
| onended | script | Triggers when media has reach the end |
|  |  |  |
| onerror | script | Triggers when an error occur |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| onfocus | script | Triggers when the window gets focus |
|  |  |  |
| onformchange | script | Triggers when a form changes |
|  |  |  |
| onforminput | script | Triggers when a form gets user input |
|  |  |  |
| onhaschange | script | Triggers when the document has change |
|  |  |  |
| oninput | script | Triggers when an element gets user input |
|  |  |  |
| oninvalid | script | Triggers when an element is invalid |
|  |  |  |
| onkeydown | script | Triggers when a key is pressed |
|  |  |  |
| onkeypress | script | Triggers when a key is pressed and released |
|  |  |  |
| onkeyup | script | Triggers when a key is released |
|  |  |  |

**OUTCOMES:**

**CODE:**

<!DOCTYPE *html*>

<html>

    <head>

        <script *src*="script.js"></script>>

    </head>

    <body>

        <button *onClick* = "revealMessage()">Click Me!!</button>

        <div *id*="hiddenMessage" *style*="display:none">

            <p>You are going on Cruise</p>

            <button *id*="countDownButton" *onclick*="countDown()">10</button>

        </div>

    </body>

*function* revealMessage() {

  document.getElementById("hiddenMessage").style.display = "block";

}

*function* countDown() {

*var* currentVal = document.getElementById("countDownButton").innerHTML;

*var* newVal = 0;

  if (currentVal > 0) {

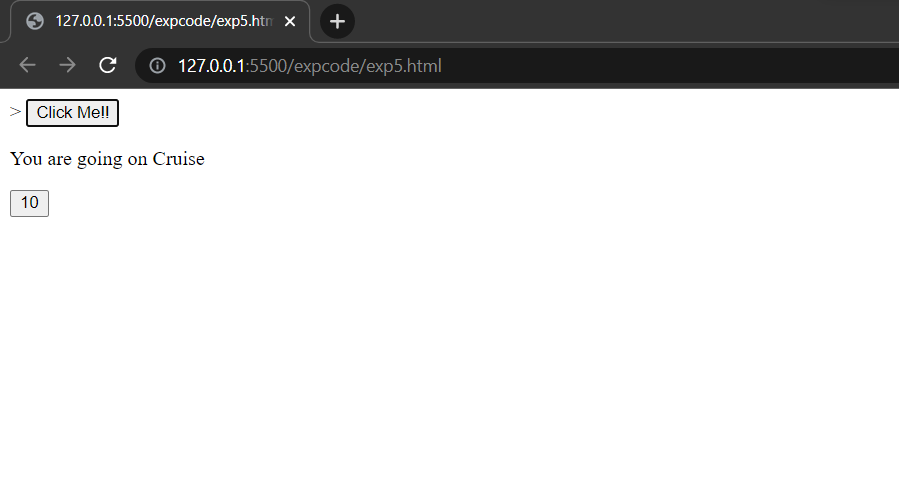
    newVal = currentVal - 1;

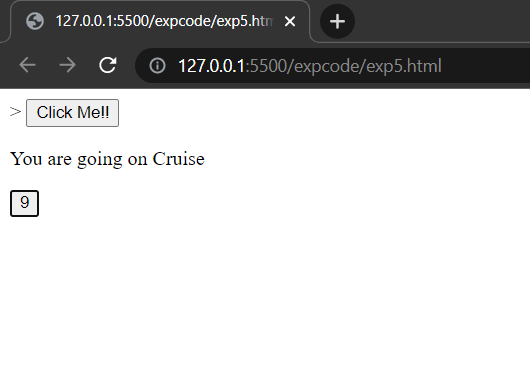
  }

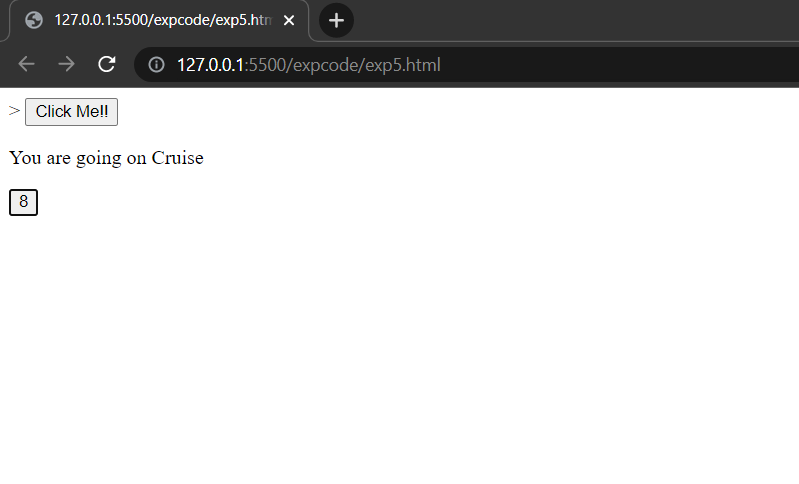
  document.getElementById("countDownButton").innerHTML = newVal;

}

**OUTPUT:**





****

**CONCLUSION:**

In the above experiment, we studied about basic Javascript concepts and have created an interactive Registration page with validations using Javascript.