JavaScript

ACADGILD

What is JavaScript

- JavaScript is a full-fledged programming language that can be applied to an HTML document and used to create **dynamic interactivity on websites.**
- It was invented by **Brendan Eich,** co-founder of the Mozilla project, the Mozilla Foundation and the Mozilla Corporation.

First JavaScript Program

- JavaScript is written in html file inside a script tag.
- alert is a function to display popup message to user.
- The plus (+) operator is used to concatenate strings.

```
<html>
    <head>
        <script type = text/javscript">
            Var name = "Smith";
            Var age = 29;
            alert("The name is "+name + "And age is "+age);
        </script>
    </head>
    <body> </body>
</html>
```

Variable

- **Variables** are containers that you can store values in.
- Declaring a variable with the var keyword, followed by any name you want to call it:

  Syntax : var variableName;

- **Rule :** They must begin with a letter or the underscore character.
- JavaScript is an **untyped programming language**.
- Semicolons in JavaScript terminates a statement.

Data Types

types of data values.

- **Primitive data types**
  - **Number:** integer & floating-point numbers
  - **Boolean:** logical values "true" or "false"
  - **String:** a sequence of alphanumeric characters
- **Composite data types (or Complex data types)**
  - **Object:** a named collection of data
  - **Array:** a sequence of values
- **Special data types**
  - **Null:** an initial value is assigned
  - **Undefined:** the variable has been created by not

Example of Data Types

Variable Data Types	Explanation	Example
String	A string of text. To signify that the variable is a string, you should enclose it in quote marks.	var myVariable = 'Bob';
Number	A number. Numbers don't have quotes around them.	var myVariable = 10;
Boolean	A True/False value. true/false are special keywords in JS, and don't need quotes.	var myVariable = true;
Array	A structure that allows you to store multiple values in one single reference.	var myVariable = [1,'Bob','Steve',10]; Call each member of the array like this: myVariable[0],myVariable[1], etc.
Object	Everything in JavaScript is an object, and can be stored in a variable.	var myVariable = document.querySelector('h1'); Note : All of the above examples too.

Operators

An operator is basically a mathematical symbol that can act on two values (or variables) and produce a result.

- Arithmetic operators + , - , / , * , %
- Logical operators && , || , !
- Comparison operators == , === , >= , <=
- String operators +
- Bit-wise operators & , ! , >> , <<
- Assignment operators += , -= , /= , *=

Conditional

Conditionals are code structures that allow you to test whether an expression returns true or not, and then run different code depending on the result.

- "if" statement
- "if ... else" statement
- "else if" statement
- "if/if ... else" statement
- "switch" statement

if Statement

- It is the main conditional statement in JavaScript.
- The keyword "if" always appears in lowercase.
- The condition yields a logical true or false value.
- The condition is true, statements are executed.

Syntax : if (condition) { statements; }

if...else Statement

You can include an "else" clause in an if statement when you want to execute some statements if the condition is false.

Syntax :

```
if (condition) { statements; }
else { statements; }
```

else if Statement

Allows you to test for multiple expression for one true value and executes a particular block of code.

Syntax :

```
if (condition) { statement; }
   else if (condition) { statement; }
   else { statement; }
```

if/if ... else statement

Syntax :

```
if (condition) {
    if (condition) { statements; }
    else { statements; }
 }
```


switch statement

Allows you to merge several evaluation tests of the same variable into a single block of statements.

Syntax :

```
switch (expression) {
    case label1:
        statements; break;
    default:
        statements;
  }
```

Looping Statement

Loops let you run a block of code a certain number of times.

- for" Loops
- "for/in" Loops
- "while" Loops
- "do … while" Loops

for loop

- One of the most used and familiar loops is the for loop.
- It iterates through a sequence of statements for a number of times controlled by a condition.
- The change_exp determines how much has been added or subtracted from the counter variable.

Syntax :

```
for (initial_expression; test_exp; change_exp)
   {  statements;  }
```

for –in loop

- When the for/in statement is used, the counter and termination are determined by the length of the object.
- The statement begins with 0 as the initial value of the counter variable, terminates with all the properties of the objects have been exhausted.
  - E.g. array → no more elements found

Syntax :

```
for (counter_variable in object)
  { statements; }
```

while loop

- The while loop begins with a termination condition and keeps looping until the termination condition is met.
- The counter variable is managed by the context of the statements inside the curly braces.

Syntax :

```
initial value declaration;
   while (condition) {
     statements;
     increment/decrement statement;
   }
```

do while loop

- The do/while loop always executes statements in the loop in the first iteration of the loop.
- The termination condition is placed at the bottom of the loop.

Syntax :

```
initial value declaration;
   while (condition) {
     statements;
     increment/decrement statement;
   }
```

function

- 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).

Syntax :

```
functionName(parameter1, parameter2, parameter3) {
code to be executed
}
```

Accessing Unnamed Arguments

How do we get more arguments than listed in parameters?

- There is a special pseudo-array inside each function called **arguments.**
- It contains all parameters by their number: arguments[0], arguments[1] etc.
- **Example :**

```
function sayHi() {
for(var i=0; i<arguments.length; i++) {
    alert("Hi, " + arguments[i]) }
}
sayHi("Ron", "Alice") // 'Hi, Ron', then 'Hi, Alice'
```

Scope

"Scope" refers to the variables that are available to a piece of code at a given time.

Functions have access to variables defined in the same scope

Example :

```
var foo = 'hello';

var sayHello = function() {
console.log(foo);
};

sayHello(); // logs 'hello'
console.log(foo); // also logs 'hello'
```

Scope

Code outside the scope in which a variable was defined does not have access to the variable

Example :

```
var sayHello = function() {
var foo = 'hello';
console.log(foo);
};

sayHello(); // logs 'hello'
console.log(foo); // doesn't log anything
```

Scope

Variables with the same name can exist in different scopes with different values

Example :

```
var foo = 'world';

var sayHello = function() {
var foo = 'hello';
console.log(foo);
};

sayHello(); // logs 'hello'
console.log(foo); // logs 'world'
```

Debugging in JavaScript

With the recent boom of JavaScript, all major browsers come with their own debug tools.

Examples:

- Chrome has **Chrome DevTools**. You can access it by shortcut key Ctrl+Shift+Alt.
- For Firefox you can use **firebug extension**.

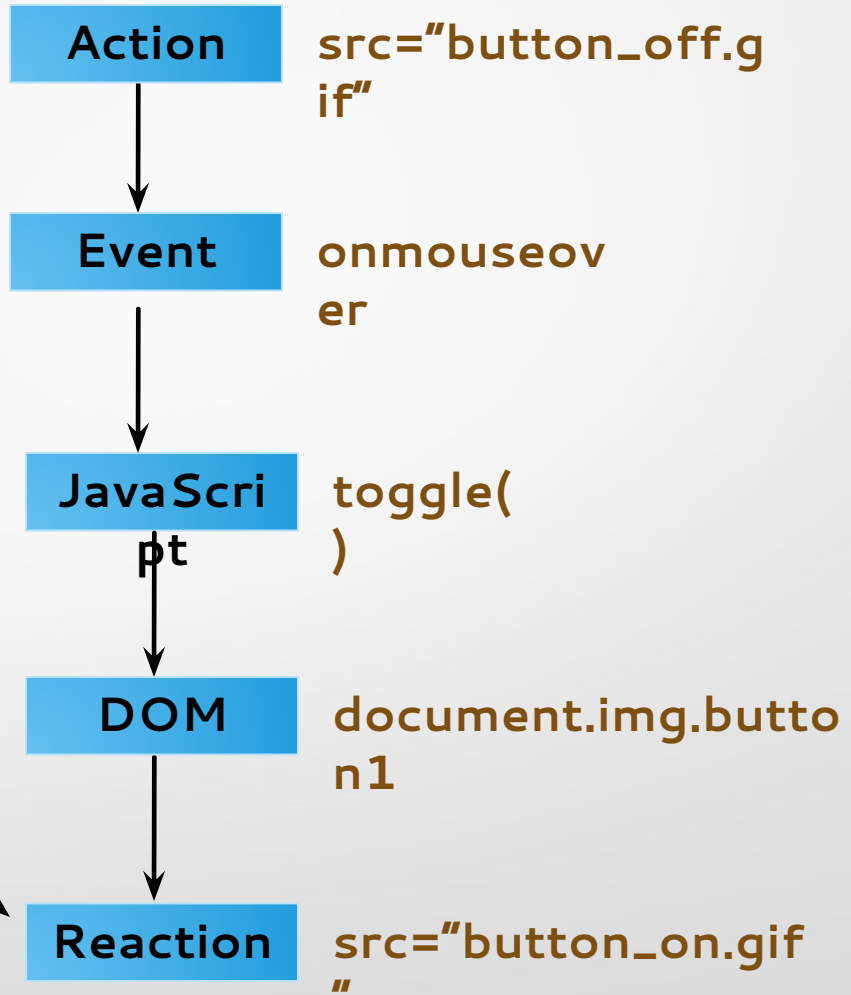
DOM (Document Object Model)

- A standard platform- and language-neutral programming interface **for building, accessing, and manipulating valid HTML and well-formed XML documents.**

- Ultimate goal is to make it possible for programmers to write applications that **work properly on all browsers and servers, and on all platforms.**

How DOM Works ?

```
<head>
    <script>
    function toggle()
    {
        document.img.button1.src
        =
    "button_on.gif"; }
    </script>
</head>
<body> <a href="test.html"
    onmouseover="toggle()">
    <img name="button1"
    src="button_off.gif">
</a> </body>
```



DOM Manipulation

- **document.getElementsByTagName(tagname)**

  This method returns a collection of all elements reference in the document with the specified tag name.

- **document.getElementsByClassName(classname)**

  This method returns a collection of all elements reference in the document with the specified class name.

Modifying HTML Using innerHTML

InnerHTML is the property of DOM object nodes. Using this property we can get/set the html inside a tag.

Example :

```
<head> <script type = "text/javascript">
    function addHeading(){
    var ref = document.getElementById("container");
    var htmlToInsert= '<h3> This is the Heading</h3>';
    ref.innerHTML = htmlToInsert;
} </script>
</head>
<body> <button onclick = "addHeading()">Add
  Heading</button>
        <div id="container"></div> </body>
```


Events

- JavaScript can also respond to events which can also be actions by the user.

- Example clicking on a element, hovering over an element are all actions by user and JavaScript uses events which can react to these actions.

- JavaScript attaches a function called an event listener or event handler to a specific event and the function invokes when the event occurs.

Events can be attached in the following ways

1) Inline HTML attributes
2) Adding to element properties with JavaScript
3) Using DOM Event Listeners

Inline HTML Elements

- Events can be attached as attributes to the elements like this

```
<div onclick = "showMsg()">Click<div>
```

Adding to Element Properties

We can also assign a function to the onclick property of a DOM node element. Have a look at the code snippet below

```
<div id = "container">click here</div>
<script type = "text/javascript">var ref =
    document.getElementById('container');
ref.onclick = function () {
alert('The div area is clicked');
};
</script>
```

Using DOM Event Listener

- The best way to handle events is to use the event listener approach.We can assign listeners to the click event using the addEventListener() method.

```
ref.addEventListener(event,function)
```

Event Types

- **Mouse Events** – mouseup , mousedown
- **Keyboard events** – keydown , keyup
- **window events** –load, unload
- **Form events** – focus ,change

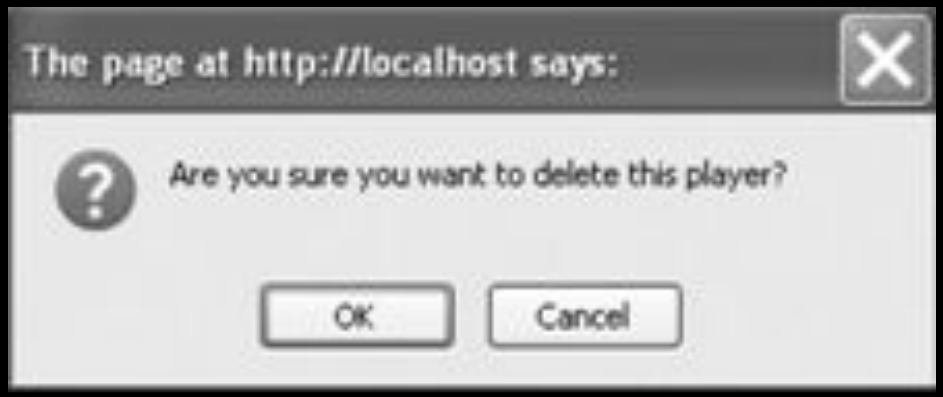
Action Dialog

```
<script type="text/javascript">

        function confirmDelete() {
          var answer = confirm("Are you sure you want"
            + "to delete this player?");
          return answer
        }

</script>

<form method="post" action="/delete">
          <p>
          <input type="submit" value="Delete"  onclick="return
          confirmDelete()" />
          </p>
</form>
```



Form Validation

```
<script>
 function validate() {
   if (document.getElementById("name").value.length ==
0) {
     alert("Please complete the required fields\n" +
       "and resubmit.");
     return false;
   }
   return true;
 }
</script>
```


```
<h3>Add Player:</h3>
<form id="form1" action="addplayer" onsubmit="return
validate()" >
 <p>Name: <input type="text" id="name" /></p>
  ...
 <p><input type="submit" value="Register" /></p>
</form>
```