Java Script in HTML

Why Study JavaScript?

- JavaScript is one of the **3 languages** all web developers **must** learn:

1. **HTML** to define the content of web pages
. **CSS** to specify the layout of web pages
3. **JavaScript** to program the behavior of web pages

The <script> Tag

- JavaScript code must be inserted between <script> and </script> tags.

```
<script>
  //java script code
</script>
```

JavaScript Functions and Events

- A JavaScript **function** is a block of JavaScript code, that can be executed when "called" for.
- For example, a function can be called when an **event** occurs, like when the user clicks a button.

JavaScript in <head> or <body>

- You can place any number of scripts in an HTML document.
- Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

JavaScript in <head>

- In this example, a JavaScript function is placed in the <head> section of an HTML page.
- Example of Javascript in head

JavaScript in <body>

- In this example, a JavaScript function is placed in the <body> section of an HTML page.
- Example

External JavaScript

- Scripts can also be placed in external files: myScript.js
- To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:
- <script src="myScript.js"></script>

External JavaScript Advantages

- It separates HTML and code
- It makes HTML and JavaScript easier to read and maintain
- Cached JavaScript files can speed up page loads

External References

- External scripts can be referenced with a full URL or with a path relative to the current web page.
- This example uses a full URL to link to a script:

```
<script src="https://www.w3schools.com/js/myScript1.js"></script>
```

JavaScript Display Possibilities

- Writing into the HTML output using **document.write()**.
- Writing into an HTML element, using **innerHTML**.
- Writing into an alert box, using **window.alert()**.

JavaScript Functions

- A JavaScript function is a block of code designed to perform a particular task.

```
function myFunction()
 {
     //Statements
}
```

Declaring (Creating) JavaScript Variables

- Creating a variable in JavaScript is called "declaring" a variable.
- You declare a JavaScript variable with the **var** keyword:

```
var carName;
```

- After the declaration, the variable has no value. (Technically it has the value of **undefined**)

HTML Events

- An HTML event can be something the browser does, or something a user does.
- Here are some examples of HTML events:
  - An HTML web page has finished loading
  - An HTML input field was changed
  - An HTML button was clicked

Common HTML Events

- **onchange**
  - An HTML element has been changed
- **onclick**
  - The user clicks an HTML element
- **onmouseover**
  - The user moves the mouse over an HTML element
- **onmouseout**
  - The user moves the mouse away from an HTML element
- **onkeydown**
  - The user pushes a keyboard key
- **onload**
  - The browser has finished loading the page

String Properties in JS

- String Properties
- length - The **length** property returns the length of a string
- Methods:
  - **toLowerCase()**
  - **toUpperCase()**
  - **charAt(x)**
  - **indexOf(substr, [start])**
  - **lastIndexOf(substr, [start])**
  - **substr(start, [length])**
  - **concat(v1, v2,…)**

String Function in JS

- **toLowerCase()**
  - Returns the string with all of its characters converted to lowercase.
- **toUpperCase()**
  - Returns the string with all of its characters converted to uppercase.

- **charAt(x)**
  - Returns the character at the "x" position within the string.
- **indexOf(substr, [start])**
  - Searches and returns the index number of the searched character or substring within the string.
  - If not found, -1 is returned.
- **lastIndexOf(substr, [start])**
  - Searches and returns the index number of the searched character or substring within the string from end to beginning.
  - f not found, -1 is returned.

- **substr(start, [length])**
  - Returns the characters in a string beginning at "start" and through the specified number of characters, "length".
- **concat(v1, v2,…)**
  - Combines one or more strings (arguments v1, v2 etc) into the existing one and returns the combined string. Original string is not modified.

Operators of JS

- Arithmetic Operators : +, -, *, /, %, ++, --
- Logical Operators : &&, ||, !
- Comparison Operators : ==, !=, <, >, <=, >=

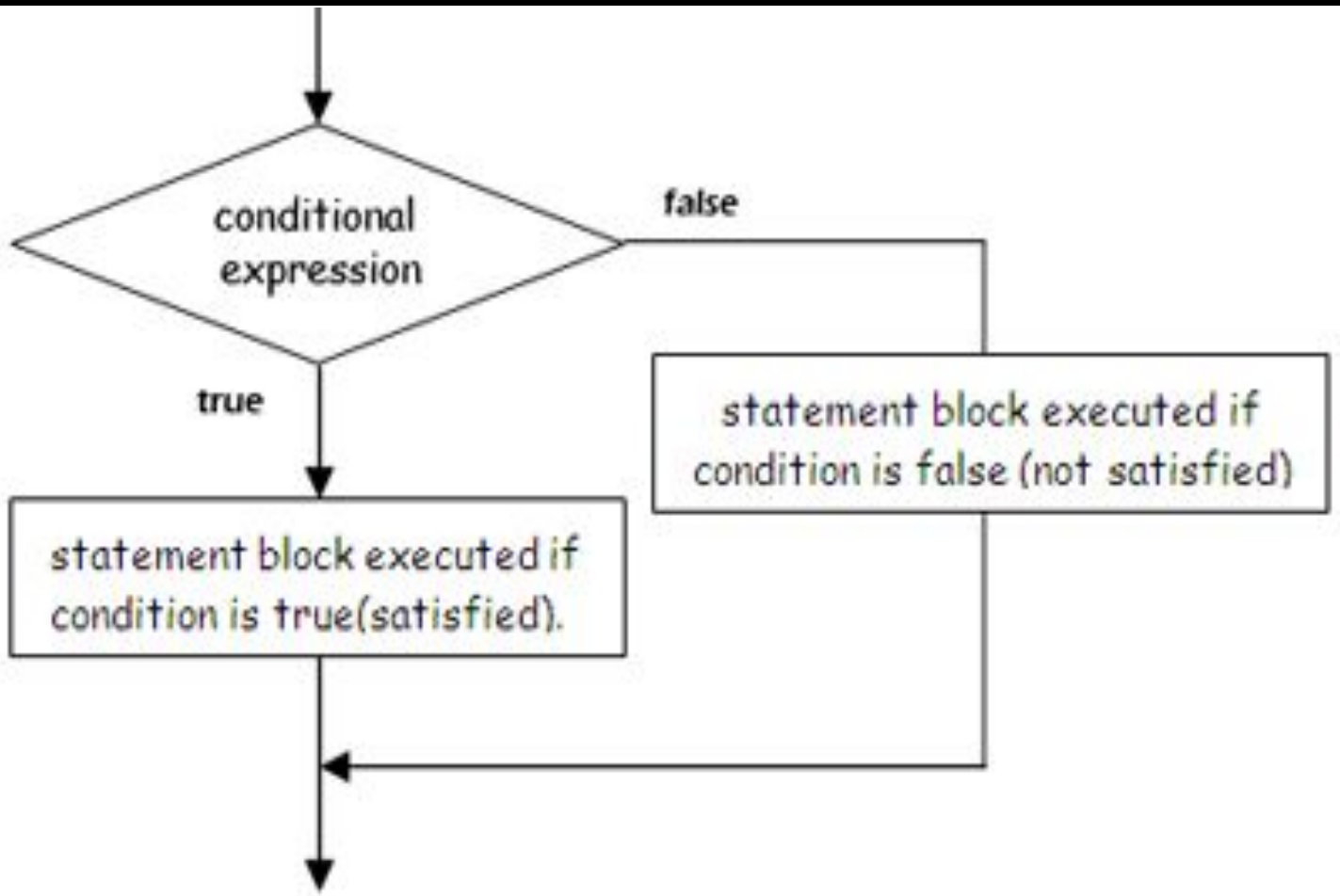
OPERATORS in JavaScript

Category	Operator	Name/Description	Example	Result
Arithmetic	+	Addition	3+2	5
	-	Subtraction	3-2	1
	*	Multiplication	3*2	6
	/	Division	10/5	2
	%	Modulus	10%5	0
	++	Increment and then return value	X=3; ++X	4
		Return value and then increment	X=3; X++	3
	--	Decrement and then return value	X=3; --X	2
		Return value and then decrement	X=3; X--	3
Logical	&&	Logical "and" evaluates to true when both operands are true	3>2 && 5>3	False
	||	Logical "or" evaluates to true when either operand is true	3>1 || 2>5	True
	!	Logical "not" evaluates to true if the operand is false	3!=2	True
Comparison	==	Equal	5==9	False
	!=	Not equal	6!=4	True
	<	Less than	3<2	False
	<=	Less than or equal	5<=2	False
	>	Greater than	4>3	True
	>=	Greater than or equal	4>=4	True
String	+	Concatenation(join two strings together)	"A"+"BC"	ABC

Conditional Statements

- Conditional statements are used to perform different actions based on different conditions.

- The conditional statement will either return TRUE or FALSE.


conditional expression
false
true
statement block executed if condition is false (not satisfied)
statement block executed if condition is true(satisfied).

- JavaScript supports two conditional statements:
  - If…Else Statement
  - Switch Statement.

If…Else Statements

- The **if statement** executes a statement if a specified condition is true.
- If the condition is false, else part can be executed.
- Syntax

```
If(condition){
    block of code to be executed if the condition is true
}
else{
    block of code to be executed if the condition is false
}
```

- Multiple if…else statements can be nested to create an else if clause

```
If(condition1){
        statement1
  }
  else if(condition2) {
        statement2
  }
  else if(condition n) {
        statement3
  }
  else
        statement4
  }
```

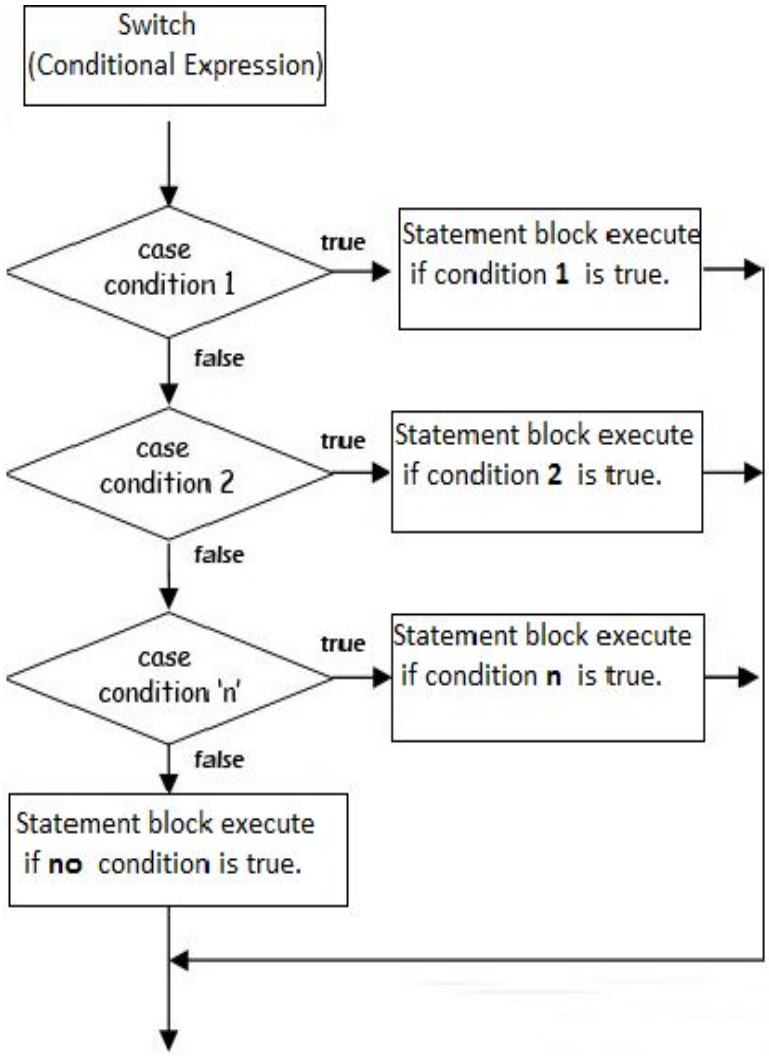
- Example

```
var a;
a=document.getElementById("fn").value;
if(a%2==0)
{
    alert("Even number");
}
else
{
    alert("odd number");
}
```

```
<SCRIPT LANGUAGE = "Javascript">

var first_name = "Jenny";

if (first_name == "Kenny") {
        document.write("First IF " + first_name);
}
else if (first_name == "Benny")  {
        document.write("Second IF " + first_name);
}
else if (first_name == "Lenny")  {
        document.write("Third IF " + first_name);
}
else {
        document.write("ELSE PART " + first_name);
}

</SCRIPT>
```

Switch Statement

- The value of the variable given into switch is compared to the value following each of the cases, and one value matches the value of the variable that statement is executed.
- The break is used to break out of the case statement.



- Example

```
var c;

c="Div";

switch(c)
{
     case "Add":
          alert(eval(a)+eval(b));
          break;
     case "Sub":
          alert(a-b);
          break;
     case "Div":
          alert(a/b);
          break;
     case "Mul":
          alert(a*b);
          break;
}
```

Loops in JS

- Looping structure enables to achieve repetitive tasks.
- A loop continues to operate until either a condition is true or your explicitly choose to exit the loop.
- Types of Loops
  - For Loop
  - While Loop
  - Do…While Loop
  - For…in loop

For Loop

- For loop, consisting of an initialization, an evaluation, and an increment.
- It is an entry controlled loop.
- Syntax

```
for(initialization ; test-condition ; increment/decrement)
{
    body of the loop
}
```

- The following three parts-
  - *Initialization* where we initialize counter to a starting value.
  - *Test-condition* which test for the given condition is true or false. If condition is true, then the code given inside the loop will be executed. If condition is false, then the loop will be terminated.
  - *Increment/decrement* where you can increase or decrease the counter.
- Example

```
<script>
for (i=1; i<=5; i++)
{
document.write(i + "<br/>")
}
</script>
```

For... In Loop

- The for…in loops through the elements of an array or through the properties of an object.
- Syntax

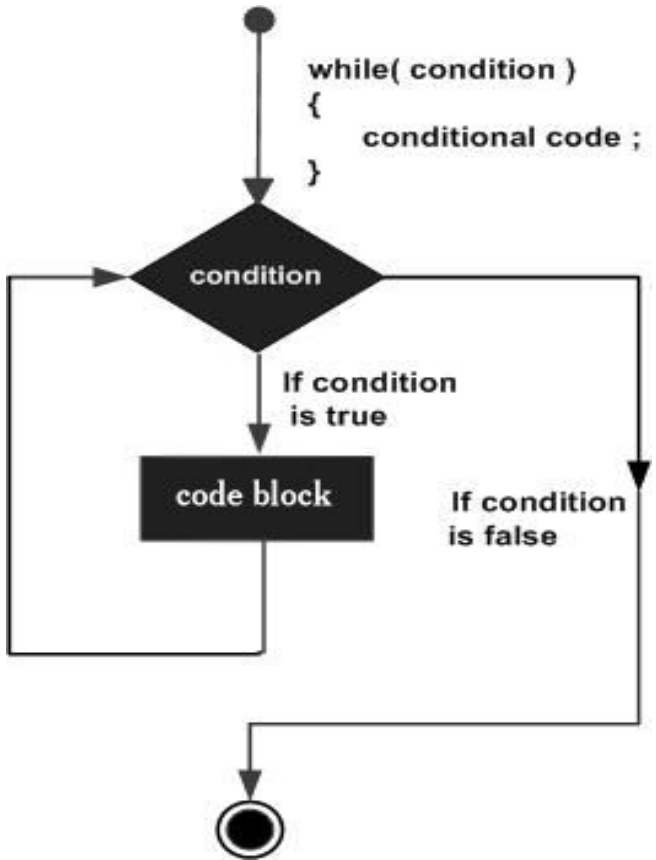
```
For(variable in object)
{
  Statements
}
```

- The variable argument can be a named variable, an array element, or a property of an object

While Loop

- The While loop will continue to execute until its test condition evaluates to false or the loop encounters a break statement.
- It is an entry controlled loop.
- Syntax

```
While(condition)
{
    Statement
}
```

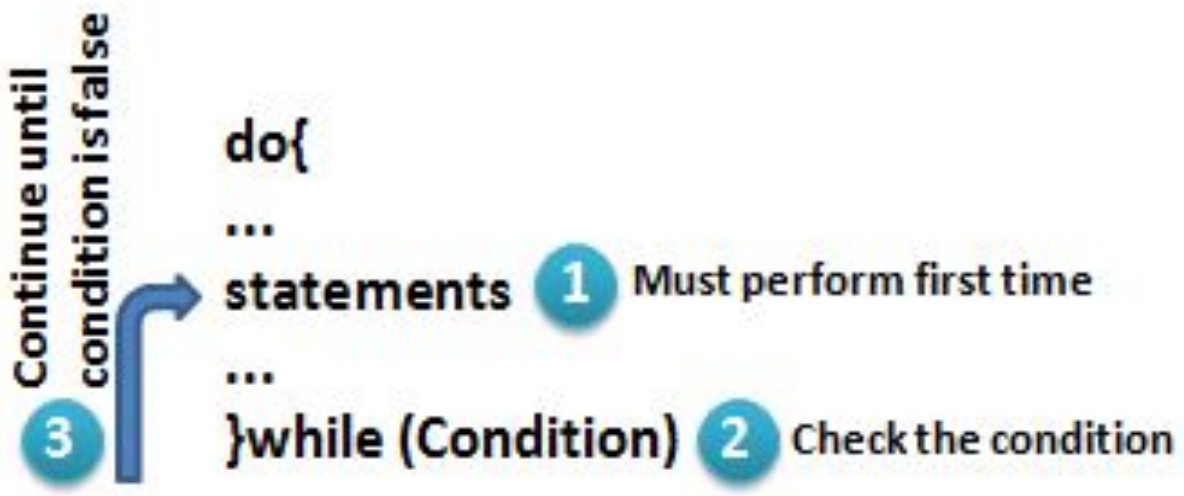


- Example

```
<script>
    var a=1;
    while(a<5)
    {
        document.write(a+"<br>");
        a++;
    }
</script>
```

Do...While Loop

- This loop will execute the block of code once, and then it will repeat the loop as long as the specified condition is true.
- Syntax



- Example

```
<script>
 var count=1;
 do
 {
     document.write(count+"<br>");
     count=count+1;
 }while(count<5);
 document.write("Loop Stopped");
</script>
```

- onclick
- onload
- onchange
- onfocus
- onblur
- onmouseenter
- onmouseout

Dialog Box

- JavaScript provides the ability to pickup user input or display small amounts of text to the user by using dialog boxes.
- These dialog boxes appear as separate windows and their content depends on the information provided by the user.
- There are three types of dialog boxes provided by JavaScript:
    - Alert Dialog Box
    - Prompt Dialog Box
    - Confirm Dialog Box

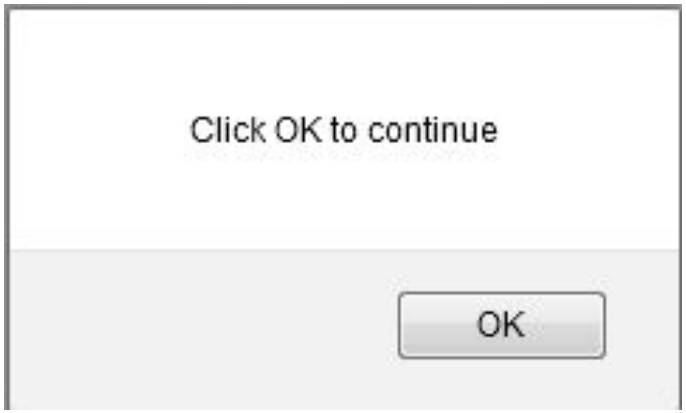
Alert Dialog Box

- An alert box is often used if you want to make sure information comes through to the user
- When an alert box pops up, the user will have to click "OK" to proceed.
- Syntax

```
alert("message");
```

- Example

```
alert("Click OK to continue");
```



Prompt Dialog Box

- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up the user will have to click either "OK" or "CANCEL" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value.
- If the user clicks "CANCEL" the box returns null.

- Syntax

  prompt("message", "default value");
- Example

  prompt("Enter your favorite color:","Violet");

Confirm Dialog Box

- A confirm box is often used if you want the user to verify or accept something.
- The user will have to click either "OK" or "CANCEL" to proceed.
- If the user clicks "CANCEL", the box returns false.
- If the user clicks "OK", the box returns true.

- Syntax

  confirm("message")
- Example

  confirm("Are you sure you want to exit our of the system");

Date and Timer

JAVASCRIPT

JavaScript - The Date Object

The Date object is a datatype built into the JavaScript language. Date objects are created with the **new Date( )** as shown below.

Once a Date object is created, a number of methods allow you to operate on it.

Syntax

You can use any of the following syntaxes to create a Date object using Date() constructor.

Constructors:

Date( )

Date(datestring)

Date(year,month,date[,hour,minute,second, millisecond])

Date Methods

getDate() / setDate(date)

getDay() / setDay(day)

getMonth() / setMonth(month)

getYear() / setYear(year)

getHours() / setHours(hour)

getSecond() / setSecond(second)

getMinute() / setMinute(min)

Timer in JS

setInterval(function, milliseconds)

Same as setTimeout(), but repeats the execution of the function continuously.

setTimeout(function, milliseconds)

Executes a function, after waiting a specified number of milliseconds.

clearTimeout(timeoutVariable)

The clearTimeout() method stops the execution of the function specified in setTimeout().

window.clearTimeout(timeoutVariable)

JavaScript - The Arrays Object

- The **Array** object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type.
- An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.
- The **Array** parameter is a list of strings or integers.
- Syntax:
- var fruits = new Array("apple", "orange"

Array Properties

- **index**
  - The property represents the zero-based index of the match in the string
- **length**
  - Reflects the number of elements in an array.

Array Methods

- **concat()**
  - Returns a new array comprised of this array joined with other array(s) and/or value(s).
- **indexOf()**
  - Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.
- **lastIndexOf()**
  - Returns the last (greatest) index of an element within the array equal to the specified value, or -1 if none is found.

- push()
  - Append the element to the end of the array:
- Example:
  - var fruits = new Array("Apple", "Orange“);
  - fruits.push("Pear"); //It will add pear at the end of the array.

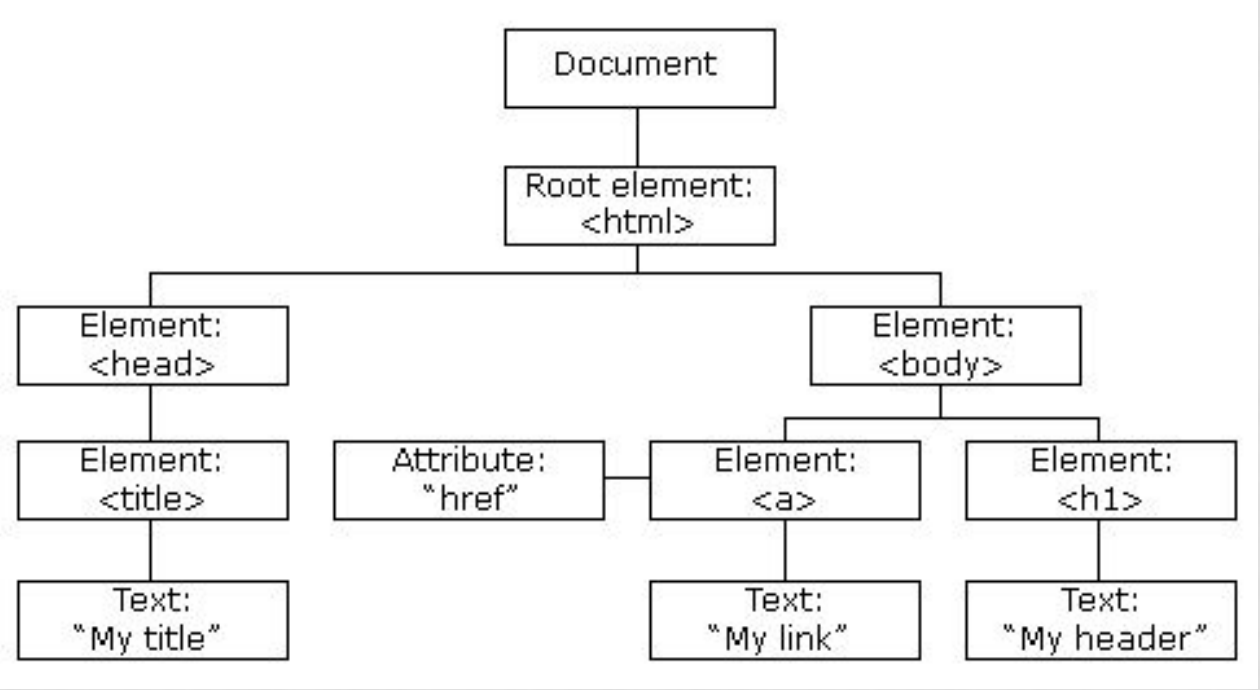
- pop()
  - Extracts the last element of the array and returns it:
- Example:
  - var fruits = new Array("Apple", "Orange", "Pear“);
  - alert(fruits.pop()); //remove "Pear"

THE HTML DOM (DOCUMENT OBJECT MODEL)

JALPA PORIYA

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:



WITH THE OBJECT MODEL, JAVASCRIPT GETS ALL THE POWER IT NEEDS TO CREATE DYNAMIC HTML:

JavaScript can change all the HTML **elements** in the page

JavaScript can change all the HTML **attributes** in the page

JavaScript can change all the **CSS styles** in the page

JavaScript can **remove** existing HTML **elements** and **attributes**

JavaScript can **add** new HTML **elements** and **attributes**

JavaScript can **react** to all existing HTML **events** in the page

JavaScript can **create** new HTML **events** in the page

HTML DOM forms Collection

- Definition and Usage
  - The forms collection returns a collection of all <form> elements in the document.
- Syntax
  - document.forms

Properties

- Length
  - Returns the number of <form> elements in the collection.

Method

- [*index*]
  - Returns the <form> element from the collection with the specified index (starts at 0).
    Note: Returns null if the index number is out of range
- item(*index*)
  - Returns the <form> element from the collection with the specified index (starts at 0).
    Note: Returns null if the index number is out of range
- namedItem(*id*)
  - Returns the <form> element from the collection with the specified id.
    Note: Returns null if the id does not exist