Web Programming

YJ – 2016

JavaScript

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.


```
<html>
   <body>

      <script language="javascript"
type="text/javascript">
         <!--
            document.write("Hello World!")
         //-->
      </script>

   </body>
</html>
```

Type

- Internal

```
<script>
    alert("Hello, world.");
    Console.log("Hello, world.");
</script>
```

- External

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

Variable

```
var surname;
var age;
var name = "Tom";
age = 43;
var apples = 5, pears = 10;

var piecesOfFruit = apples + pears;
```

(10 + 2) / 2 + 4 * 2

Operator

- + (Addition)

Adds two operands
Ex: A + B will give 30

- - (Subtraction)

Subtracts the second operand from the first
Ex: A - B will give -10

- * (Multiplication)

Multiply both operands
Ex: A * B will give 200

- / (Division)

Divide the numerator by the denominator
Ex: B / A will give 2

- % (Modulus)

Outputs the remainder of an integer division
Ex: B % A will give 0

- ++ (Increment)

Increases an integer value by one
Ex: A++ will give 11

- -- (Decrement)

Decreases an integer value by one
Ex: A-- will give 9

Logic

To find out when two values are equal, use the triple equals operator (“===”).

```
15.234 === 15.234
true
```

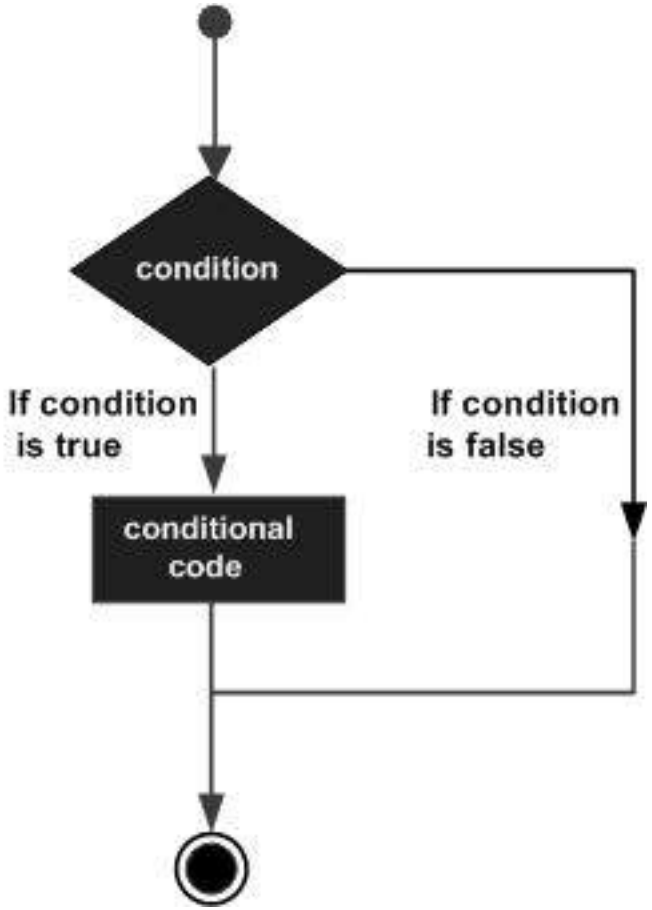
We can also determine if two values are not equal using the triple not equal operator (“!==”).

```
15.234 !== 18.4545
true
```

It’s important to know that strings containing a number and an actual number are not equal.

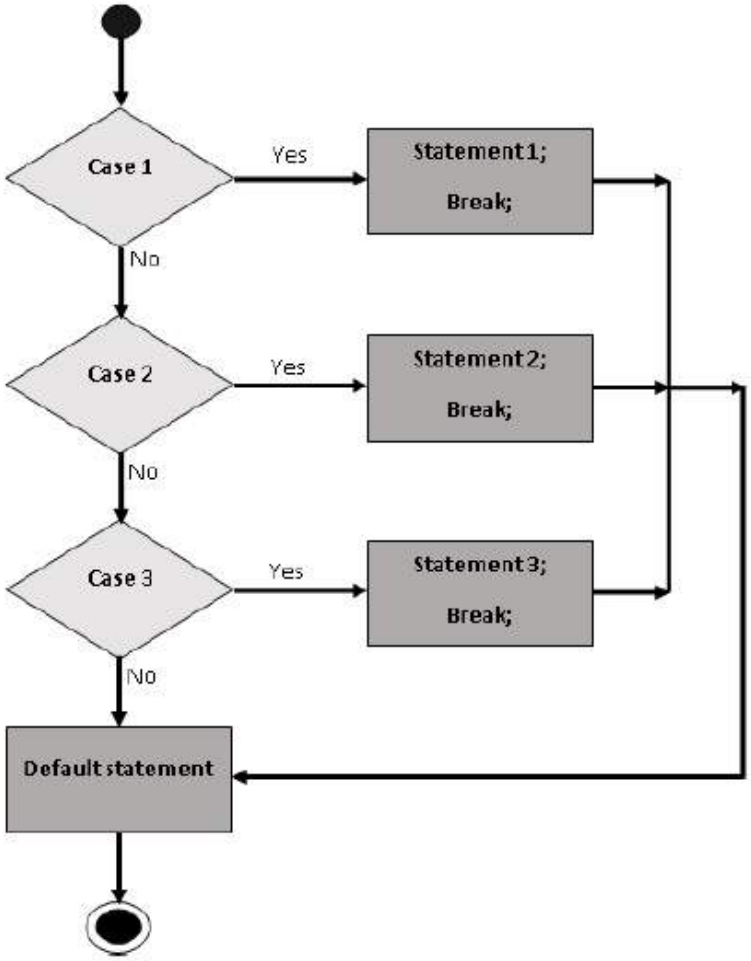
```
'10' === 10
False

10 > 5
True

10 >= 10
true
```

Condition



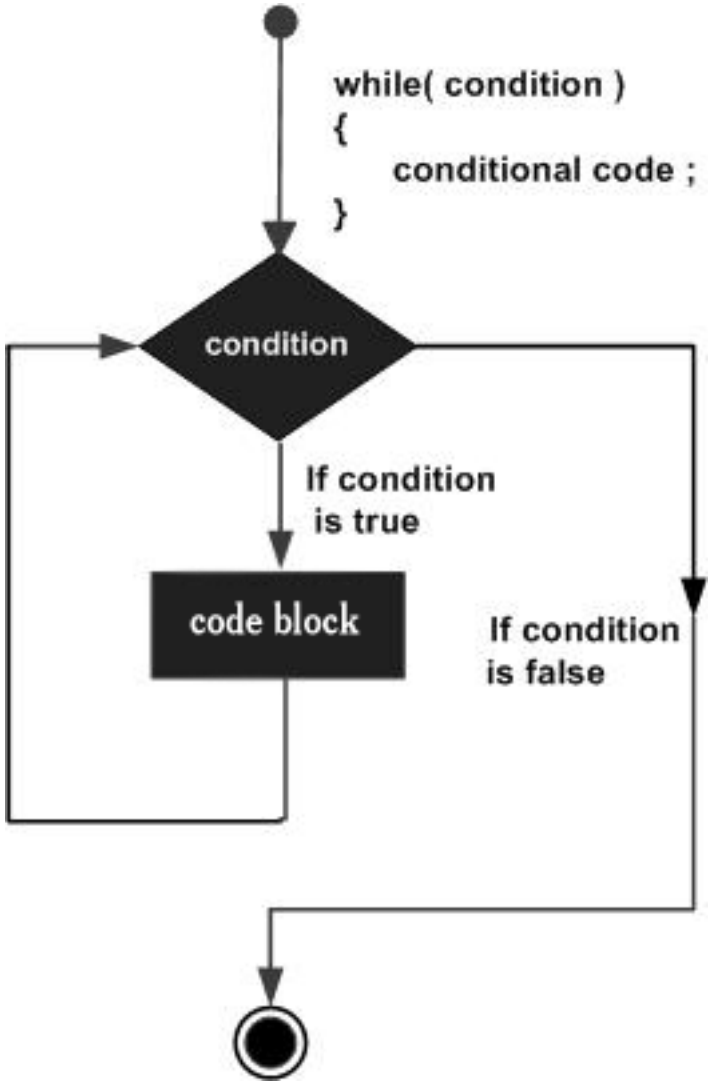

```
if (expression 1){
   Statement(s) to be executed if
expression 1 is true
}

else if (expression 2){
   Statement(s) to be executed if
expression 2 is true
}

else if (expression 3){
   Statement(s) to be executed if
expression 3 is true
}

else{
   Statement(s) to be executed if no
expression is true
}
```

Condition



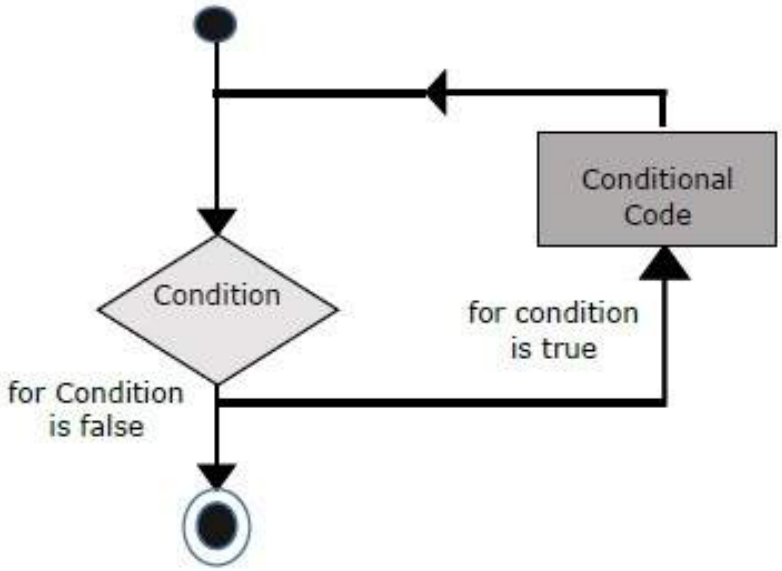

```
switch (expression)
{
   case condition 1: statement(s)
   break;

   case condition 2: statement(s)
   break;
   ...

   case condition n: statement(s)
   break;

   default: statement(s)
}
```

Loop




```
while (expression){
   Statement(s)  to be executed
if expression  is true
}
```

Loop




```
for (initialization; test condition;
iteration statement){
   Statement(s) to be executed if test
condition is true
}
```

```
for(count = 0; count < 10; count++){

document.write("Current Count : " + count
);

document.write("<br />");

}
```

Function

```
<script type="text/javascript">
   <!--
      function functionname(parameter-list)
      {
         statements
      }
   //-->
</script>
```

Event

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.

```
<html>
   <head>

      <script type="text/javascript">
         <!--
            function validation() {
               all validation goes here
               .........
               return either true or false
            }
         //-->
      </script>

   </head>
   <body>

      <form method="POST" action=""
onsubmit="return validate()">
         .......
         <input type="submit" value="Submit" />
      </form>

   </body>
</html>
```

Re-direction

```
function Redirect() {

window.location="http://www.google.com";

}

document.write("You will be redirected to main page
in 10 sec.");

setTimeout('Redirect()', 10000);
```

Dialog Boxes

- Alert()

```
alert ("This is a warning message!");
```

- Confirm()

```
var retVal = confirm("Do you want to continue ?");
              if( retVal == true ){
                 document.write ("User wants to continue!");
                 return true;
              }
```

- Promote()

```
var retVal = prompt("Enter your name : ", "your name here");
```

Object

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.

Objects

```
var val = new Number(number);
var val = new Boolean(value);
var val = new String(string);

var fruits = new Array( "apple", "orange", "mango" );
```

fruits[0] is the first element
fruits[1] is the second element
fruits[2] is the third element

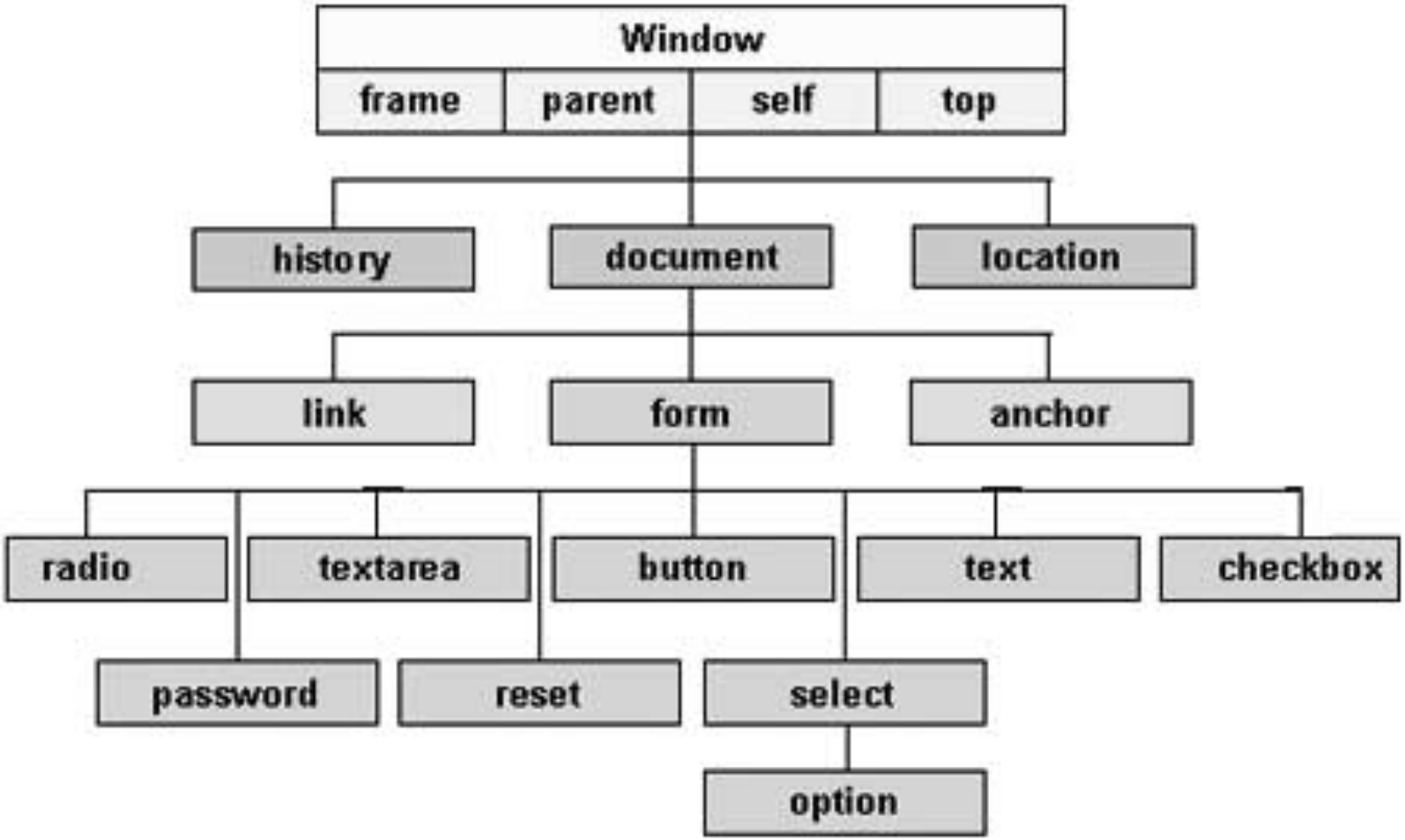
```
new Date( )
new Date(milliseconds)
new Date(datestring)
new Date(year,month,date[,hour,minute,second,millisecond ])

var pi_val = Math.PI;
var sine_val = Math.sin(30);
```

DOM

The way a document content is accessed and modified is called the **Document Object Model**, or **DOM**. The Objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a Web document.

- **Window object** – Top of the hierarchy. It is the outmost element of the object hierarchy.
- **Document object** – Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page.
- **Form object** – Everything enclosed in the <form>...</form> tags sets the form object.
- **Form control elements** – The form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.

DOM


Window
frame
parent
self
top
history
document
location
link
form
anchor
radio
textarea
button
text
checkbox
password
reset
select
option

Error Handlling

There are three types of errors in programming:
(a) Syntax Errors
(b) Runtime Errors
(c) Logical Errors

```
<script type="text/javascript">
   <!--
      try {
         // Code to run
         [break;]
      }

      catch ( e ) {
         // Code to run if an exception occurs
         [break;]
      }

      [ finally {
         // Code that is always executed
regardless of
         // an exception occurring
      }]
   //-->
</script>
```

Validation

Form validation normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button.

JavaScript provides a way to validate form's data on the client's computer before sending it to the web server. Form validation generally performs two functions.

1. **Basic Validation** − First of all, the form must be checked to make sure all the mandatory fields are filled in. It would require just a loop through each field in the form and check for data.

2. **Data Format Validation** − Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.