JAVASCRIPT

- JavaScript is the programming language of the Web.
- All modern HTML pages are using JavaScript.
- JavaScript is easy to learn.
- Using the JavaScript we can change and manipulate HTML Elements, Styles Etc...
- **JavaScript** to program the behavior of web pages
- Is a client-side scripting language.
- Can be directly embedded into a Web page by writing the code inside the <SCRIPT> tag.

The JavaScript code:

- Can be inserted in the following sections of the HTML document by using the
  <SCRIPT> tag:

- Can be embedded into a Web page by using the following syntax:
  - `<SCRIPT type="text/javascript"> JavaScript statements</SCRIPT>`

Variables

- JavaScript variables are containers for storing data values:
- All variables must be identified with unique names.
- Is declared by using the following syntax:

var test_name;

Rules for naming JavaScript variables

- Names must begin with a letter
- Names can also begin with $ and _ (but we will not use it)
- Names can contain letters, digits, underscores, and dollar signs.
- Names are case sensitive (y and Y are different variables)
- Reserved words (like JavaScript keywords) cannot be used as names

JavaScript Operators

- Consider an expression *4 + 5 is equal to 9*. Here 4 and 5 are called operands and + is called operator.
- JavaScript language supports following type of operators.
  - **Arithmetic Operators.**
  - **Comparison Operators**
  - **Logical Operators**
  - **Assignment Operators**

Arithmetic operator:

- Is used to perform arithmetic operations on variables and literals.
- Can be of the following types:

    +

    -

    *

    /

    %

Comparison operators:

- Are used to compare two values and perform an action on the basis of the comparison.
- Are of the following types:

```
<
>
<=
>=
==
!=
===
```

Logical operators:

- Are used to evaluate complex expressions.
- Return a Boolean value.
- Are of the following types:

  &&

  !

  ||

Arithmetic assignment operators:

- Are used to perform arithmetic operations and assign the value to the variable at the
- left side of the operator.
- Are of the following types:

  +=

  −=

  *=

  /=

  %=

Regular Expression

- A regular expression is a sequence of characters that forms a **search pattern**.
- A regular expression can be a single character, or a more complicated pattern.

  Example:

```
<script>
    function myFunction() {
        var str = "Visit W3Schools!";
       var n = str.search(/w3Schools/i);
        document.getElementById("demo").innerHTML = n;
    }
</script>
```

Conditional Constructs

Conditional constructs:

- Allow you to execute a block of statements based on the result of the expression being evaluated.
  - **if** to specify a block of code to be executed, if a specified condition is true
  - **else** to specify a block of code to be executed, if the same condition is false
  - **else if** to specify a new condition to test, if the first condition is false
  - **switch** to specify many alternative blocks of code to be executed

if Statement

```
<script>
if (new Date().getHours() < 18)
{
    document.getElementById("demo").innerHTML = "Good
day!";
}
</script>
```

The if...else construct:

Is used to evaluate the specified condition and perform actions on the basis of the result of evaluation.

Has the following syntax:

```
if (exp)
{
// Statements;
}
else
{
// Statements;
}
```

else if

```
if (condition1) {
    block of code to be executed if condition1 is true
} else if (condition2) {
    block of code to be executed if the condition1 is
false and condition2 is true
} else {
    block of code to be executed if the condition1 is
false and condition2 is false
}
```

Switch case

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.

```
switch(expression) {
    case n:
  code block
       break;
    case n:
       code block
      break;
   default:
       default code block
}
```

Loop Constructs

Loop structures:

- Are used to repeatedly execute one or more lines of code.
- Can be of the following types:

`while` `do...while` `for`

- The while loop:
  - Is used to repeatedly execute a block of statements till a condition evaluates to true.
  - Always checks the condition before executing the statements in the loop.
  - Has the following syntax:

```
while (expression)
        {
   statements;
        }
```

- The do…while loop:
  - Is executed at least once, even if the condition evaluates to false.
  - Has the following syntax:

```
          do
          {
      Statements;
           }
    while(condition)
```

- The for loop:
  - Allows the execution of a block of code depending on the result of the evaluation of the test condition.
  - Has the following syntax:

```
for (initialize variable; test condition; step value)
{
// code block
}
```

Functions

- Functions are used to write the code that needs to reused.
- They optimize the performance of the code.
- Are a self-contained block of statements that have a name.
- Are of the following types:

Built-in

User-defined

- Built-in functions:
  - Are ready to use as they are already coded.
- User-defined functions:
  - Are defined according to the need of the user

- Functions:
  - Are created by using the keyword, function, followed by the function name and the parentheses.
  - Are normally defined in the head section of a Web page.
  - Can optionally accept a list of parameters.
  - Are created using the following syntax:

```
function [functionName] (Variable1, Variable2)
{
//function statements
}
```

- A function is called by using the following syntax:

  functionName ();

  or

  functionName (parameter1, parameter 2...);
- A function returns a value by using the return statement as displayed in the following example:

```
function functionName()
{
var variable=10;
return variable;
}
```

JavaScript Output

- Writing into an alert box, using **window.alert()**.
- Writing into the HTML output using **document.write()**.
- Writing into an HTML element, using **innerHTML.** (document.getElementById("demo").innerHTML = 5 +6)
- Writing into the browser console, using **console.log()**.

JavaScript Popup Boxes

- JavaScript has three kind of popup boxes:

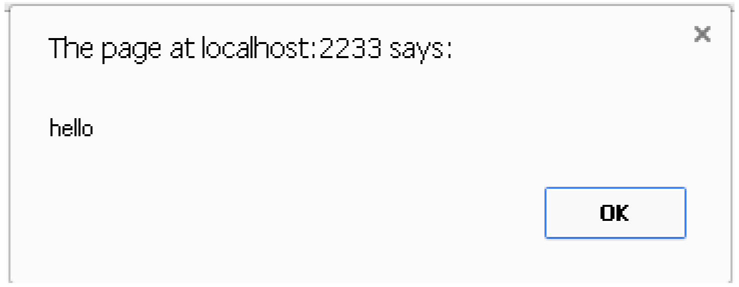
Alert box

Confirm box

Prompt box.

- **Alert 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.
  - Example

```
alert(”hello");
```

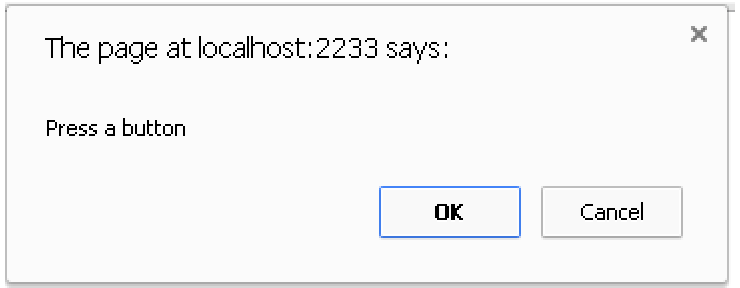

The page at localhost:2233 says:

hello

- **Confirm Box**
  - When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
  - Example

```
        var r = confirm("Press a button");
               if (r == true) {
            x = "You pressed OK!“;}

      else {    x = "You pressed Cancel!“;}
```

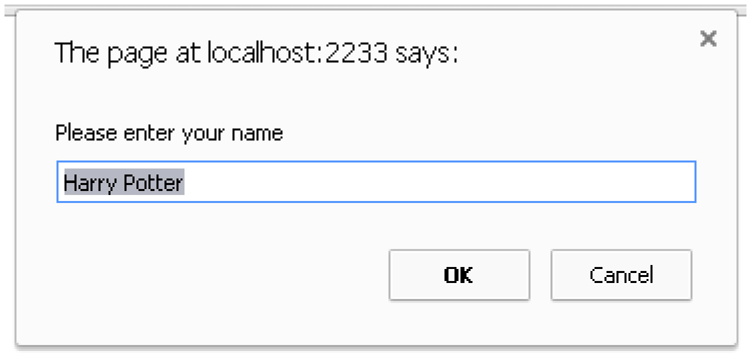


The page at localhost:2233 says:

Press a button

- **Prompt Box**
  - When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
  - Example

```
var person = prompt("Please enter your name", "Harry
Potter");
```


The page at localhost:2233 says:

Please enter your name

Harry Potter

Exception Handling

- There are three types of errors in programming:

  (a) Syntax Errors and

  (b) Runtime Errors

  (c) Logical Errors:

Syntax Errors

- Syntax errors, also called **parsing errors,** occur at compile time in traditional programming languages and at interpret time in JavaScript.
- EG:

```
<script type="text/javascript">
<!--
window.print(;
//-->
</script>
```

Runtime Errors

- Runtime errors, also called **exceptions,** occur during execution (after compilation/interpretation).

EG:

```
<script type="text/javascript">
<!–
window.printme();
//-->
</script>
```

Logical Errors

- Logic errors can be the most difficult type of errors to track down. These errors are not the result of a syntax or runtime error.

try...catch...finally Statement:

JavaScript implements three blocks to handle the exceptions

- The **try** statement lets you test a block of code for errors.
- The **catch** statement lets you handle the error.
- The **throw** statement lets you create custom errors.
- The **finally** statement lets you execute code, after try and catch, regardless of the result.

```
 <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>
```

JS Strings

- JavaScript strings are used for storing and manipulating text.

Eg: var carname = “car"; or ‘car’

- String Length

Eg:

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
document.getElementById("demo").innerHTML = txt.length;
```

- Finding a String in a String

Eg:

```
<script>
function myFunction() {
var str = document.getElementById("p1").innerHTML;
var pos = str.indexOf("locate");
document.getElementById("demo").innerHTML = pos;
}
</script>
```

Date methods let you get and set date values

getDate()	Get the day as a number (1-31)
getDay()	Get the weekday as a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)

getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

Eg:

```
<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.getTime();
</script>
```

Date Set Methods

- **setDate()** Set the day as a number (1-31)
- **setFull Year()** Set the year (optionally month and day)
- **setHours()** Set the hour (0-23)
- **setMilliseconds()**
- **Set the milliseconds (0-999)**
- **setMinutes()Set the minutes (0-59)**

- **setSeconds()** Set the seconds (0-59)
- **setTime()** Set the time (milliseconds since January 1, 1970)
- **setMonth()** Set the month (0-11)

Eg:

```
var d = new Date();
d.setDate(20);
document.getElementById("demo").innerHTML = d;
```

CSS with Java Script

```
<html>
<body>
<p id="p2">Hello World!</p>
<script>
document.getElementById("p2").style.color = "blue";
</script>
<p>The paragraph above was changed by a script.</p>
</body>
</html>
```