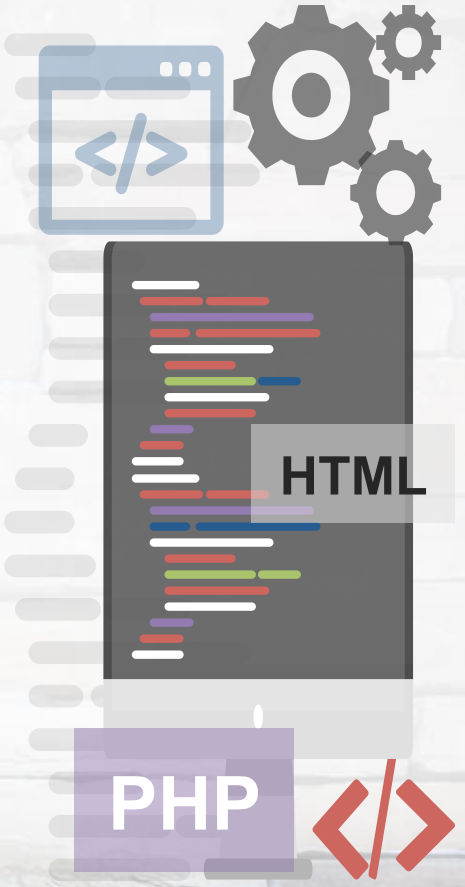
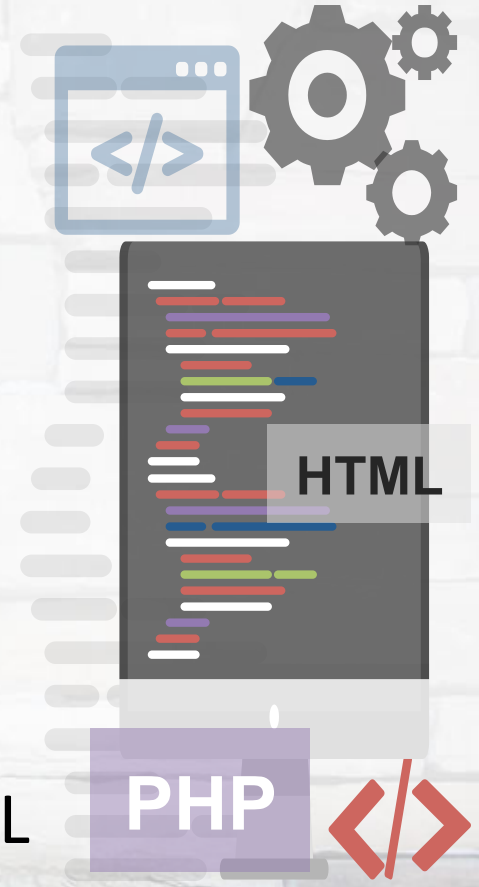
Chapter four

JavaScript



JavaScript

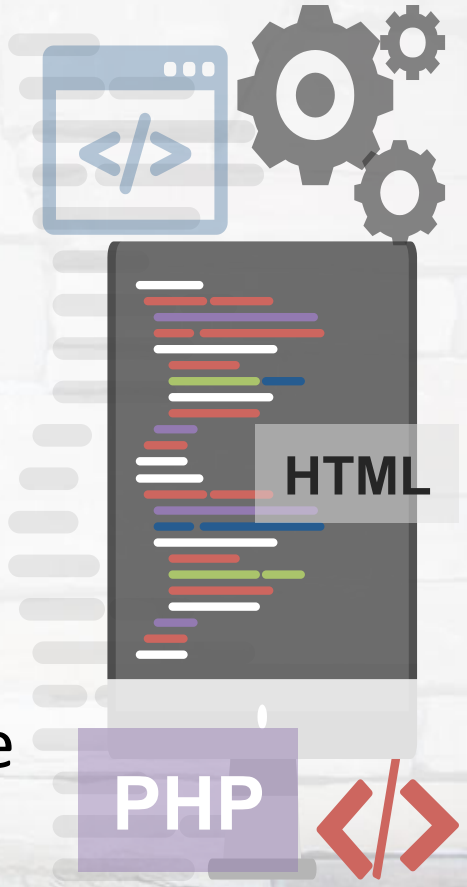
- İt was designed to add interactivity to HTML pages
- it is a scripting language (a scripting language is a lightweight programming language)
- it is an interpreted language (execute without preliminary compilation)
- usually embedded directly into HTML pages



Cont'd

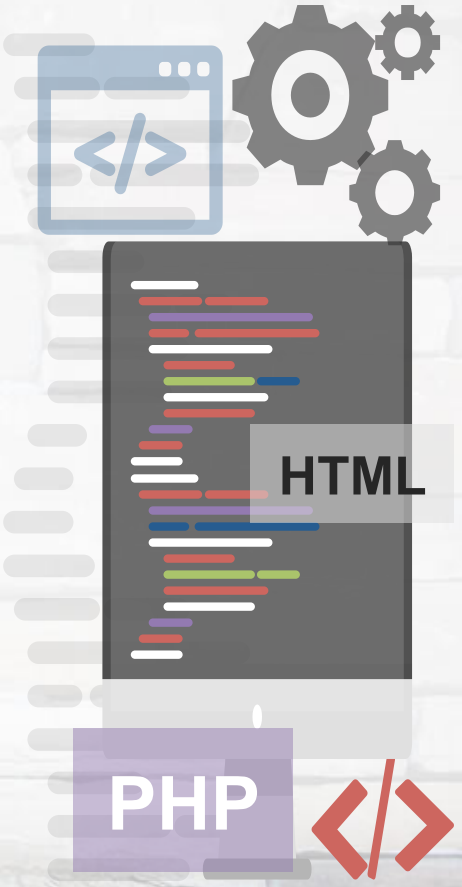
Where does JavaScript code run?

- **Browser:**
  - originally designed to run only on browser.
  - every browser has JavaScript engine
- **Node:** node is a C++ program that include googles v8 JavaScript engine.



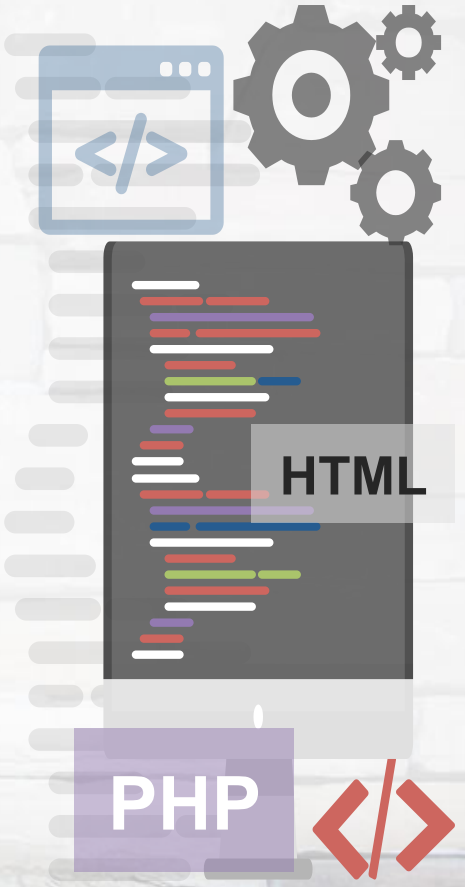
Are Java and JavaScript the Same?

- NO!
- They are two completely different languages in both concept and design!
- Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.



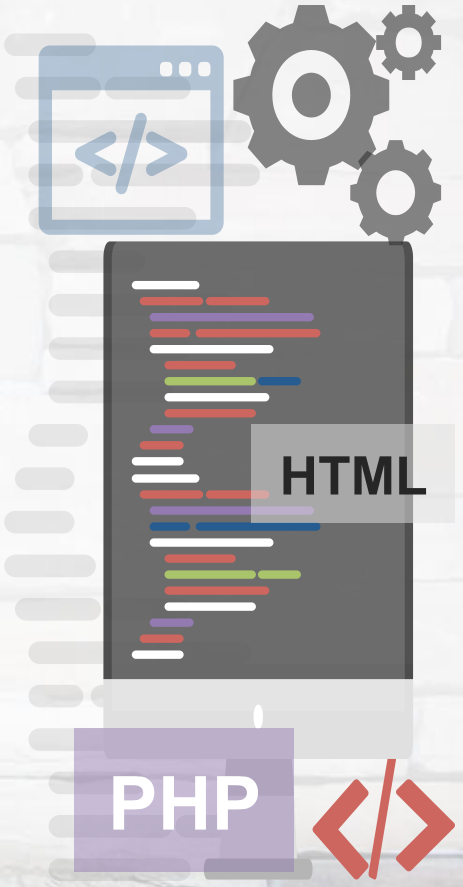
Advantages of JavaScript

- **Speed**
  - JavaScript tends to be very fast because it is often run immediately within the client's browser.
  - So long as it doesn't require outside resources, JavaScript isn't slowed down by calls to a backend server.
  - Also, major browsers all support JIT (just in time) compilation for JavaScript, meaning that there's no need to compile the code before running it.



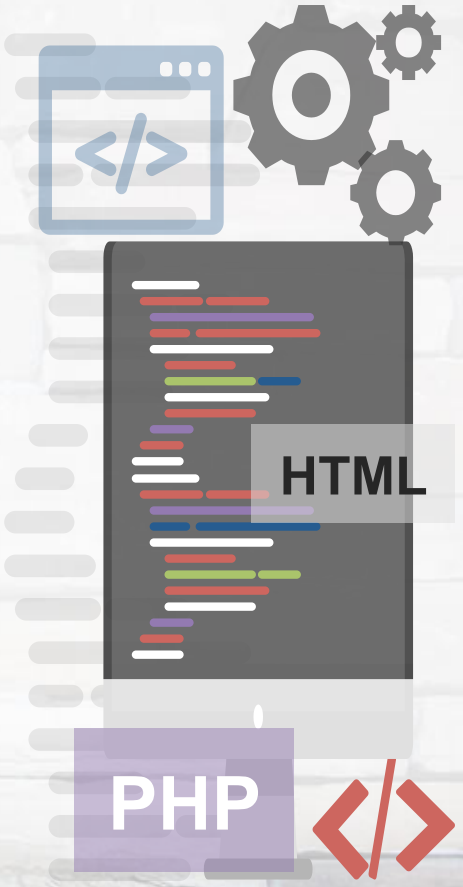
Cont'd

- **Simplicity**
  - JavaScript's syntax was inspired by Java's and is relatively easy to learn compared to other popular languages like C++.



Cont'd

- **Interoperability**
  - Unlike PHP or other scripting languages, JavaScript can be inserted into any web page.
  - JavaScript can be used in many different kinds of applications because of support in other languages like Pearl and PHP.

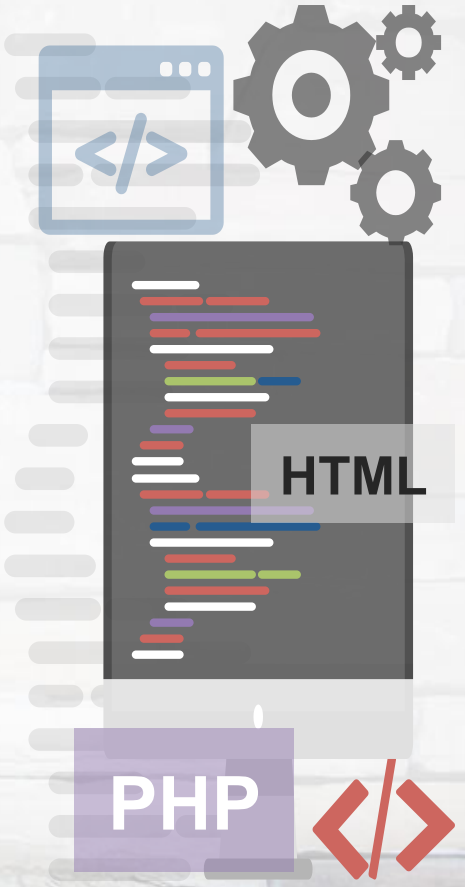


Cont'd

- **Server Load**
  - JavaScript is client-side, so it reduces the demand on servers overall, and simple applications may not need a server at all.

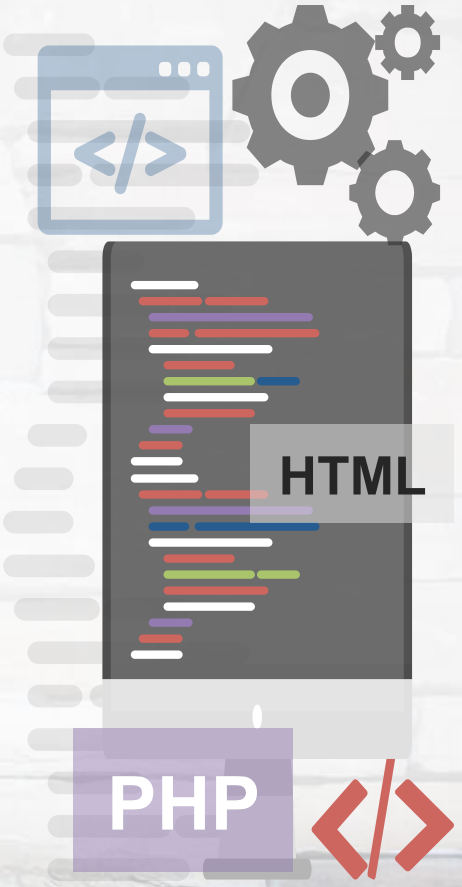
Cont'd

- **Versatility**
  - There are many ways to use JavaScript through Node.js servers.
  - If you were to bootstrap Node.js with Express, use a document database like MongoDB, and use JavaScript on the frontend for clients, it is possible to develop an entire JavaScript app from front to back using only JavaScript.



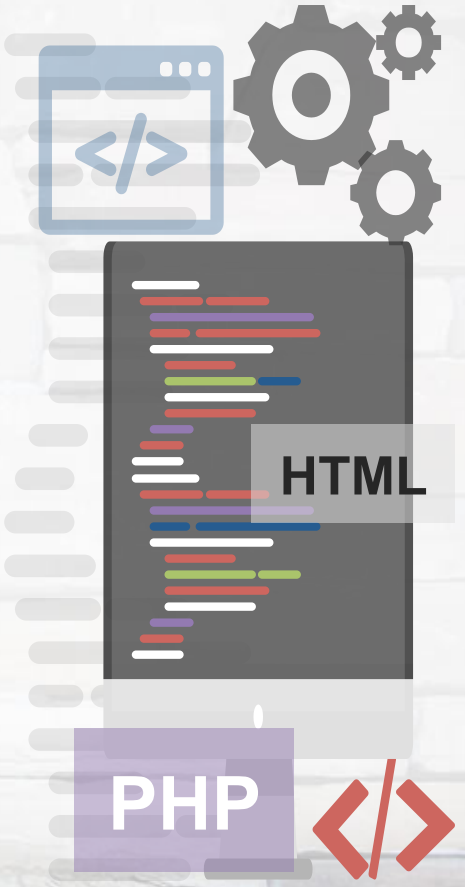
Cont'd

- **Increased interactivity:**
  - You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.



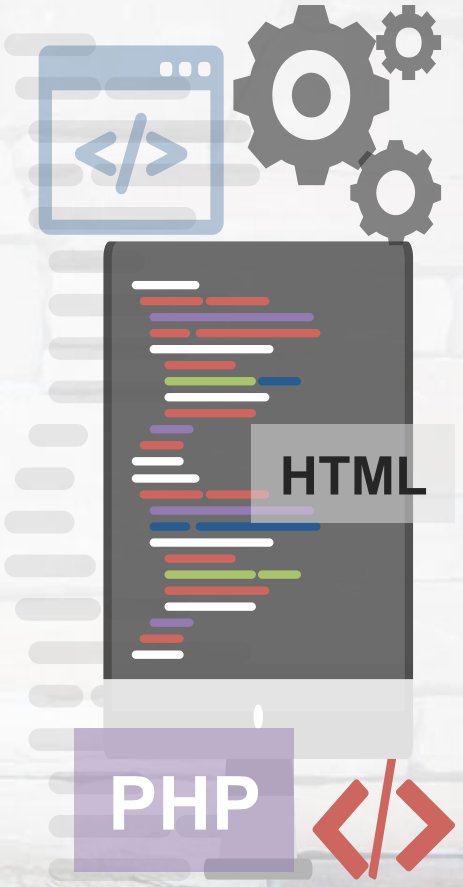
Cont'd

- **Richer interfaces:**
  - You can use JavaScript to include such items as drag-and drop components and sliders to give a Rich Interface to your site visitors.



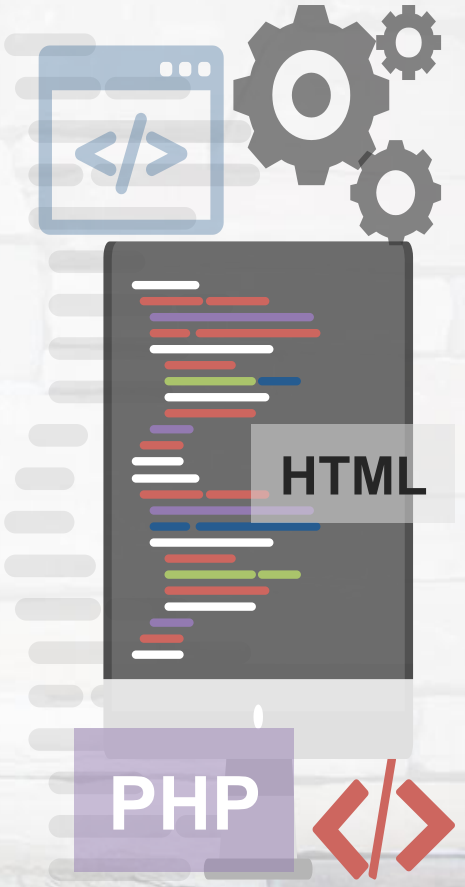
Disadvantage

- **Client-Side Security**
  - Since JavaScript code is executed on the client-side, bugs and oversights can sometimes be exploited for malicious purposes.
  - Because of this, some people choose to disable JavaScript entirely.



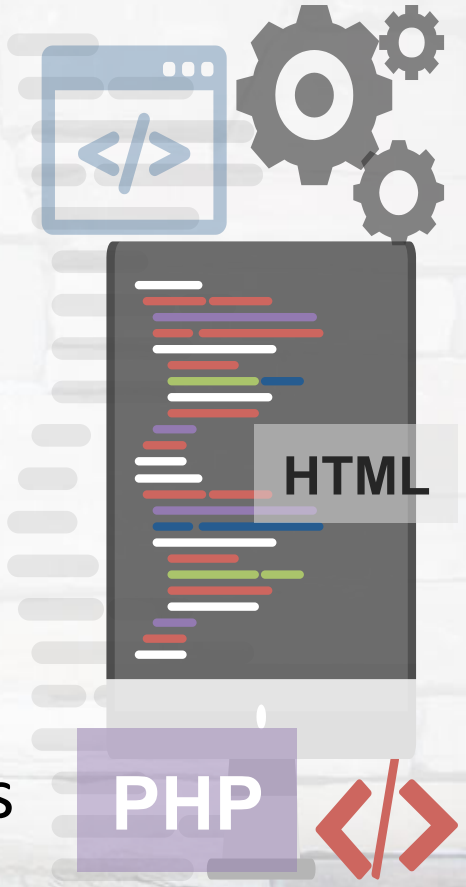
Cont'd

- **Browser Support**
  - While server-side scripts always produce the same output, different browsers sometimes interpret JavaScript code differently.
  - These days the differences are minimal, and you shouldn't have to worry about it as long as you test your script in all major browsers.



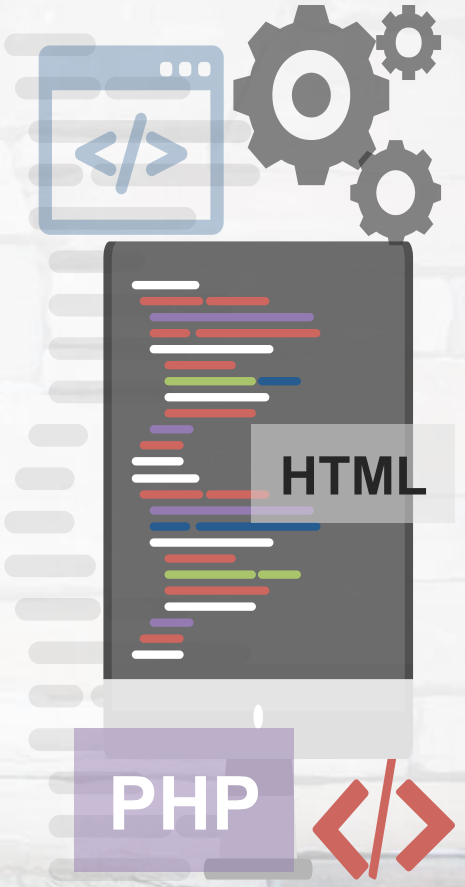
JavaScript syntax

- The JavaScript code can be written in the 'html' file or in the separate 'JavaScript file (.js)'
- JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.



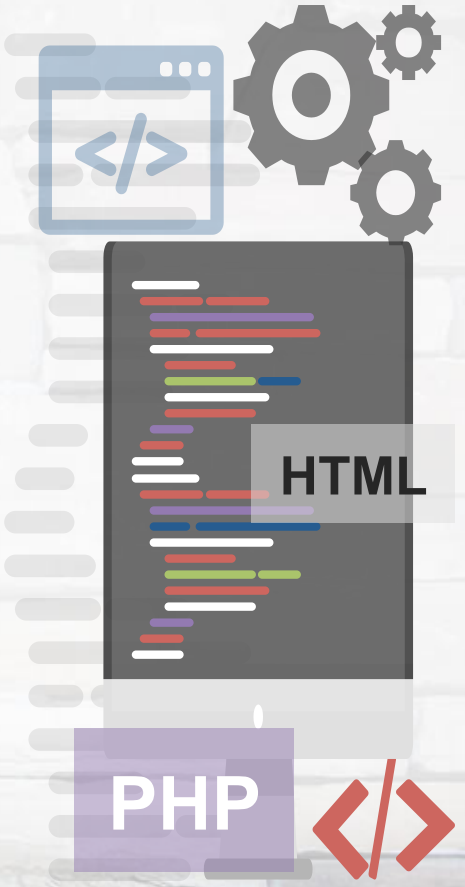
Cont'd

- The script tag takes two important attributes:
  - **Language:** specifies what scripting language you are using. Typically, its value will be JavaScript.
  - **Type:** is used to specify that the file type is text file and contains JavaScript code and its value should be set to "text/JavaScript".
  - **Src:** is used to specify the source JavaScript file



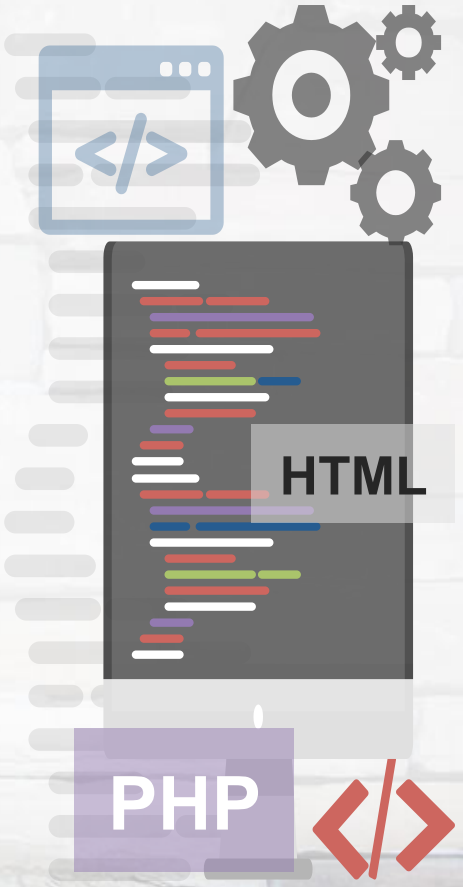
Cont'd

```
<script language="javascript"

type="text/javascript>

    JavaScript code

</script>
```



Cont'd

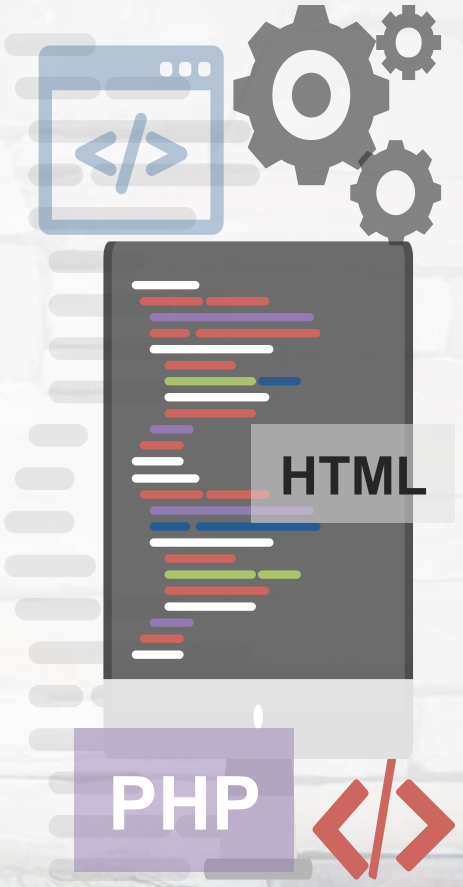
- Semicolons are required if you want to put more than one statement on a single line.
- It is **case sensitive language**.
- It is un typed language i.e. a variable can hold any type of value.



Cont'd

Scripts can be placed:

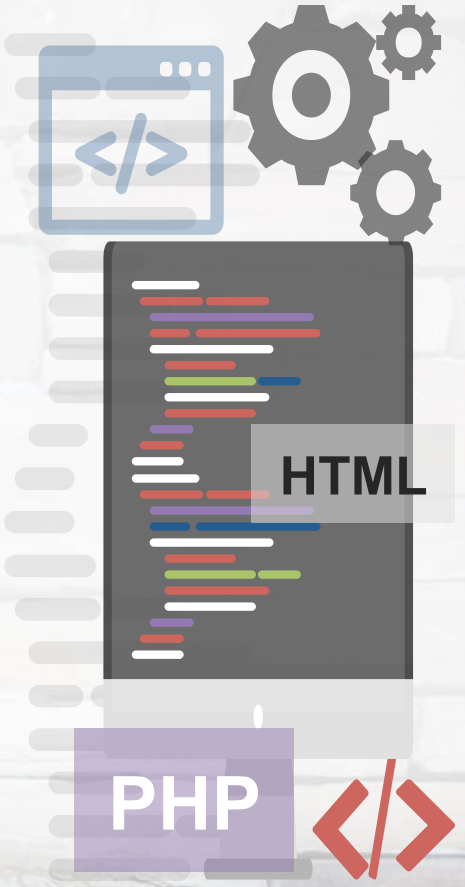
- In <head>...</head> section (If you want to have a script run on some event).
- In <body>...</body> section (If you need a script to run as the page loads).
- In <body>...</body> and <head>...</head> sections.
- In an external file with .js extension (To reuse identical JavaScript code on multiple pages of a site). Here src attribute of <script> tag is used.



Cont'd

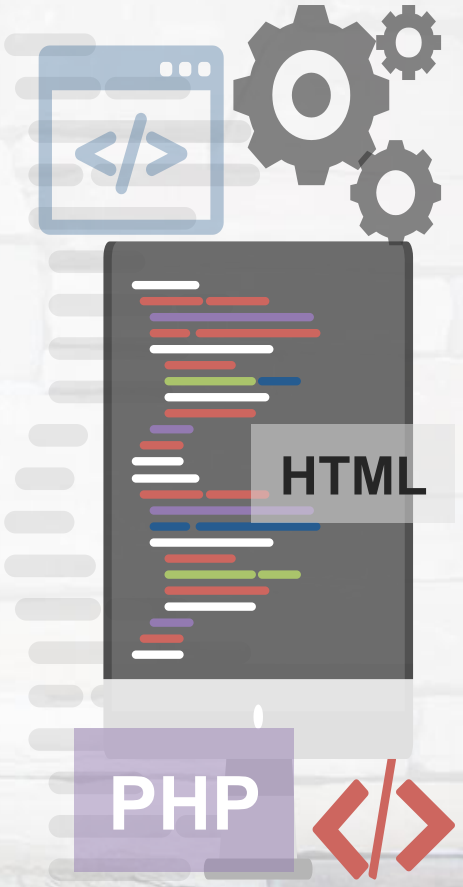
Syntax:

```
<html>
<head>
<script type="text/javascript">
...
</script>
</head>
<body>
<script type="text/javascript">
...
</script>
</body>
</html>
```



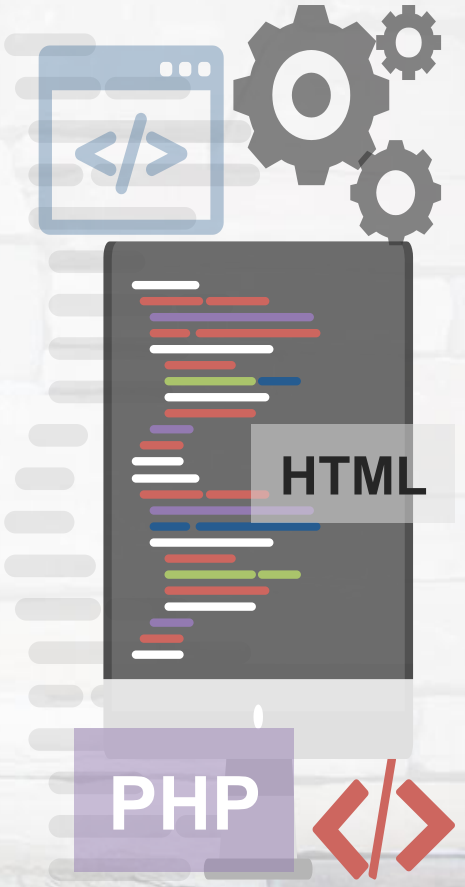
JS output methods

Method	Description
Document.write()	Display data in browser display area.
innerHTML	Display data in HTML element.



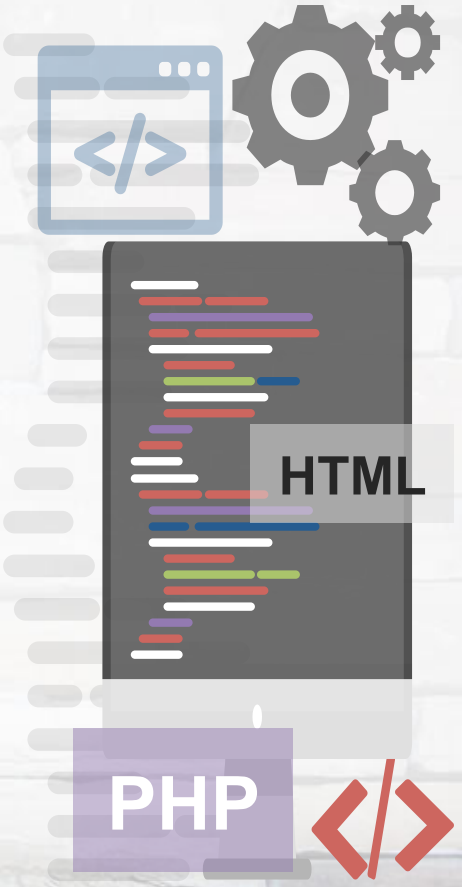
Cont'd

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body>
</html>
```



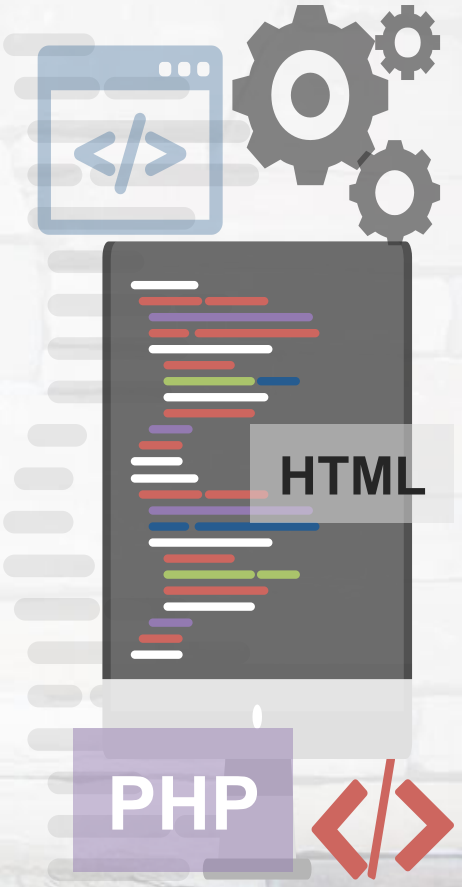
innerHTML

```
<!DOCTYPE html>
    <html>
    <body>
    <h1>my first JavaScript page </h1>
    <p>my first paragraph</p>
    <p id="demo"></p>
    <script type="text/javascript">
     document.getElementById('demo').innerHTML= 5+6;
    </script>
    </body>
    </html>
```



Js popup boxes

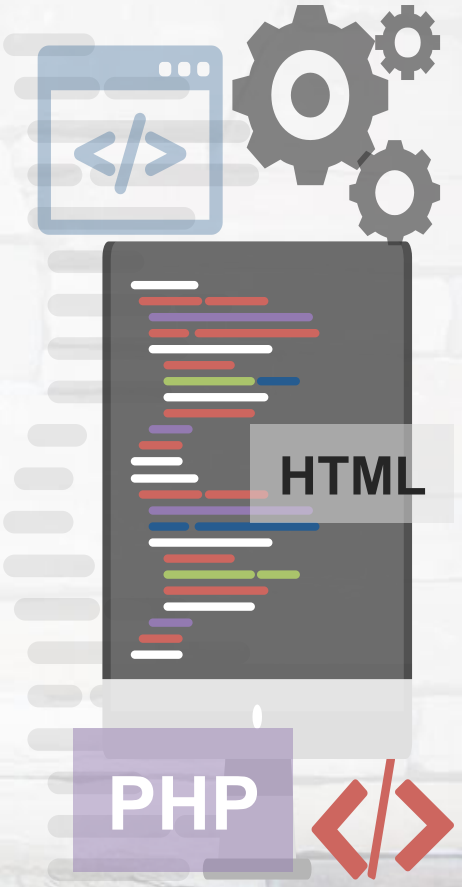
- **Alert box**
  - An alert dialog box is mostly used to give a warning message to the users.
  - When an alert box pops up, the user will have to click “ok” to proceed.
  - Window. Alert(“some text”) or alert(“some text”)
  - Eg: alert(“warning message”)



Cont'd

Confirm box

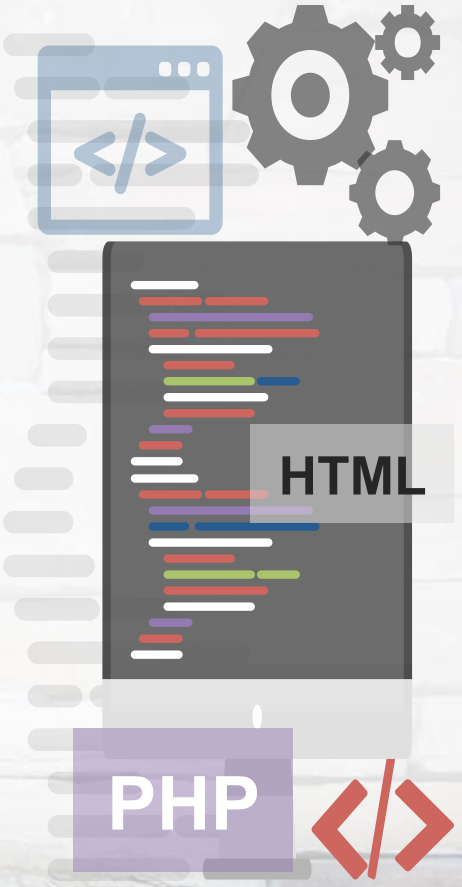
- Is often used if you want the user to verify or accept something
- When a confirm box pops up, the user will have to click either "ok" or "cancle" to proceed.
- Window. Confirm("some text") or confirm("some text")
- Eg: confirm("do you want to continue")



Cont'd

Prompt box

- Is often used if you want the user to input a value before entering a page.
- When a prompt box pop up, the user will have to click either "ok" or "cancle" to proceed after entering an input value.
  - Window.prompt("sometext"."defaulttext")

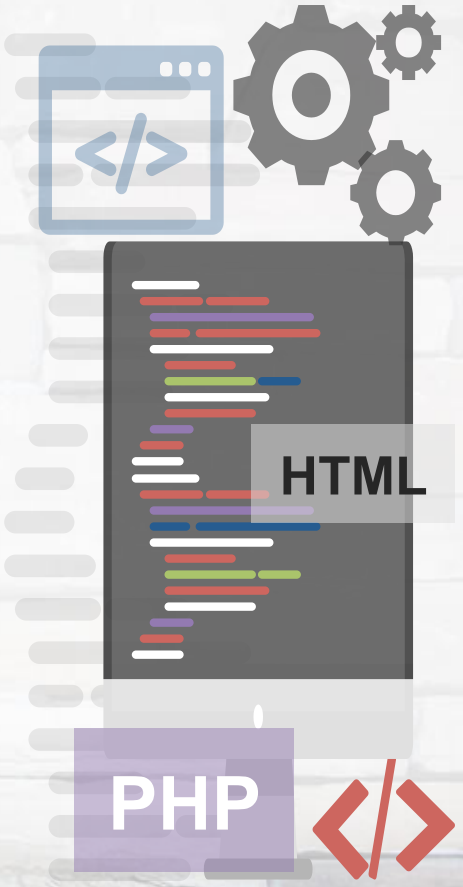


Cont'd

Prompt box

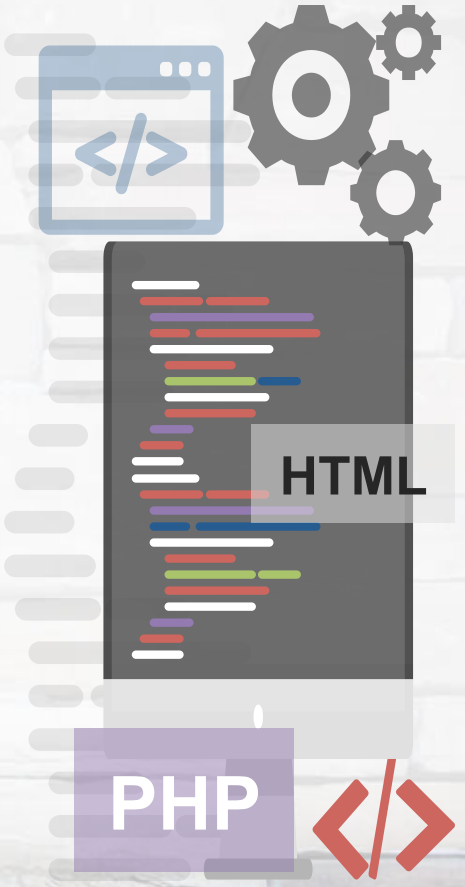
- This method can also be written without the window prefix.
- Eg:

```
var person=prompt("please enter ur
                name","name");
```



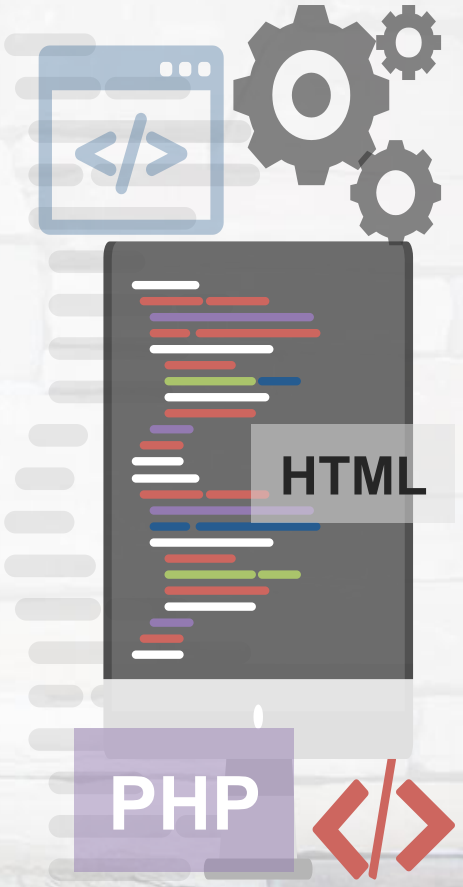
Cont'd

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script type="text/javascript">
document.write("Hello World from JavaScript!<br>");
</script>
</body>
</html>
```



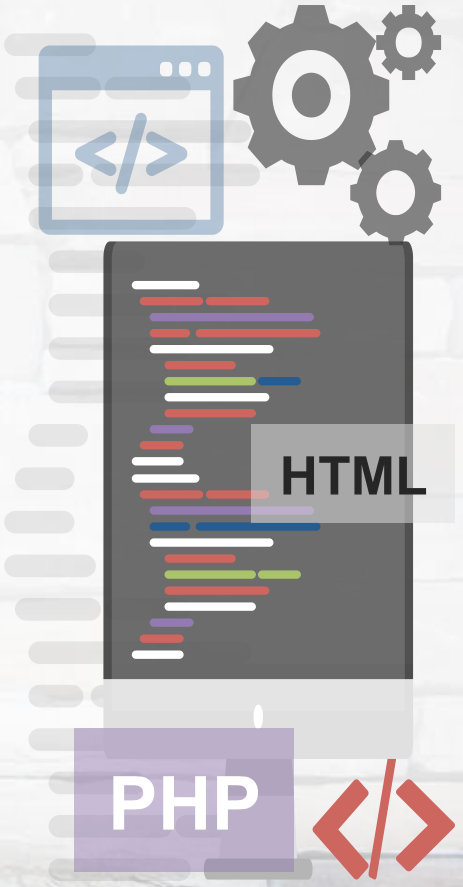
Cont'd

```
<html>
<head>
<script type="text/javascript">
function sayHello() {
alert("Hello World") }
</script>
</head>
<body>
Click here for the result
<input type="button" onclick="sayHello()" value="Say Hello“ />
</body> </html>
```



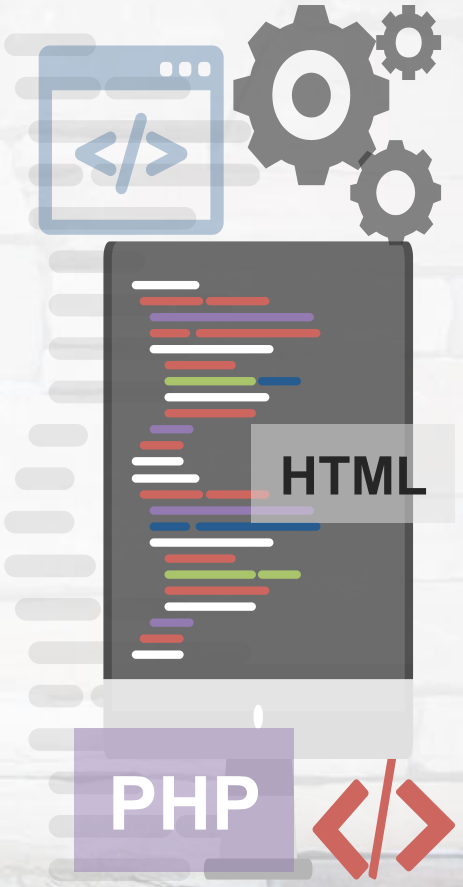
Manipulating HTML Elements

- To access an HTML element from JavaScript, you can use the **document.getElementById(*id*)** method.
- Use the **id attribute** to identify the HTML element, and **innerHTML** to refer to the element content (to display data in html element):



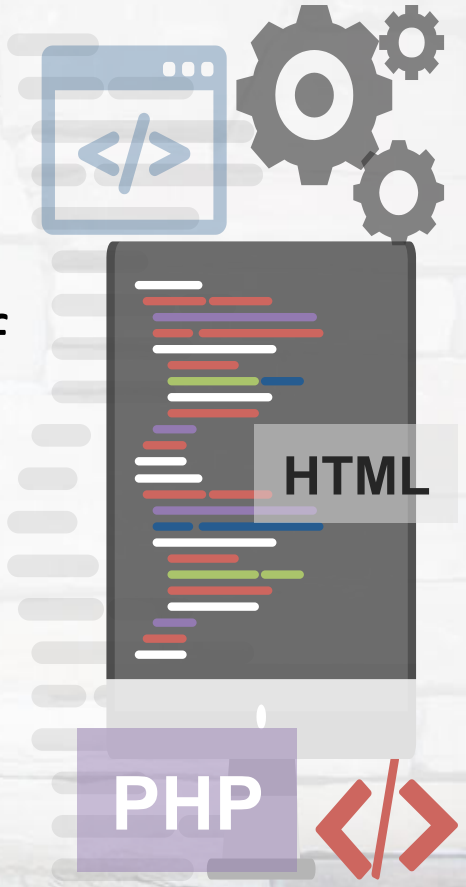
Manipulating HTML Elements

```
<html>
<body>
 <h1>My First Web Page</h1>
 <p id="demo">my first paragraph</p>
 <script>
document.getElementById("demo").inne
rHTML = "Paragraph changed.";
</script>
</body>
</html>
```



Comment in java script

- JavaScript supports both C-style and C++-style comments. Thus:
  - Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
  - Any text between the characters /* and */ is treated as a comment. This may span multiple lines.



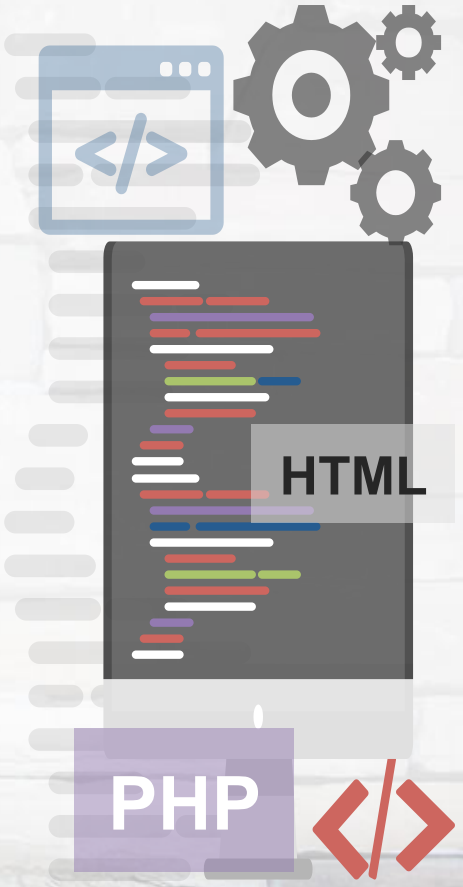
JavaScript Variables

- It is possible to create a variable **with or without** the var statement

  var strname = some value or **var** a= 5;

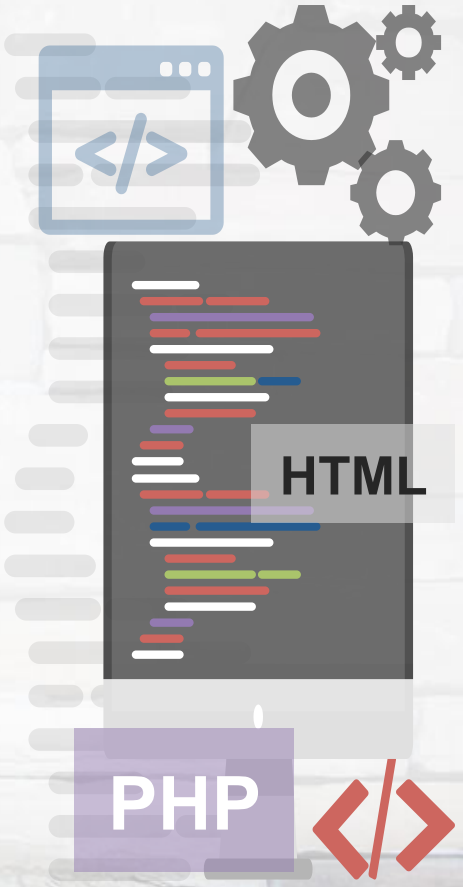
  strname = some value or a=5;

- JavaScript is ***untyped* language**. This means that a JavaScript variable can hold a value of any data type.



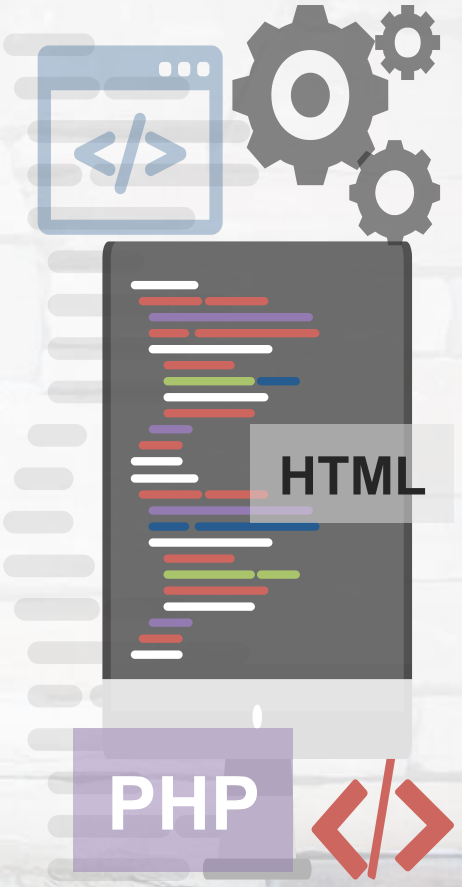
Cont'd

- You **should not use** any of the JavaScript **reserved keywords** as a variable name.
- JavaScript variable names **should not start** with **a numeral (0-9)**. They must **begin with a letter or an underscore** character.



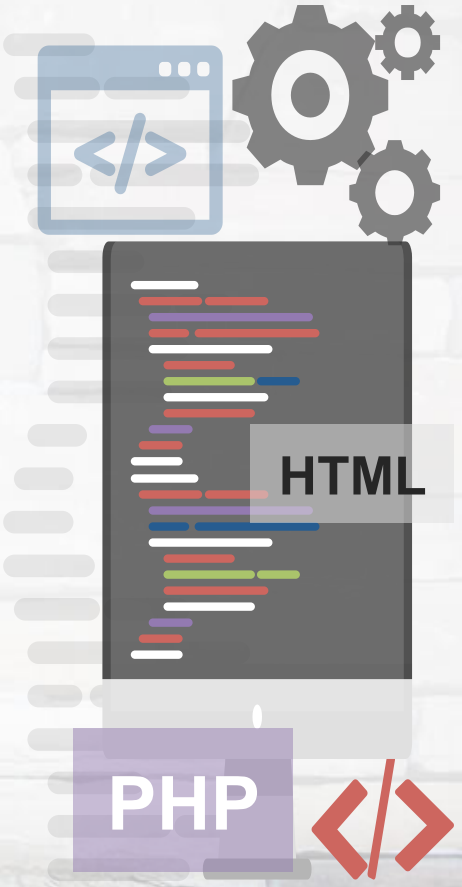
Cont'd

- **For example:**
  - **123test** is an invalid variable name but **_123test** is a valid
  - **Name** and **name** are two different variables. (case sensitive)



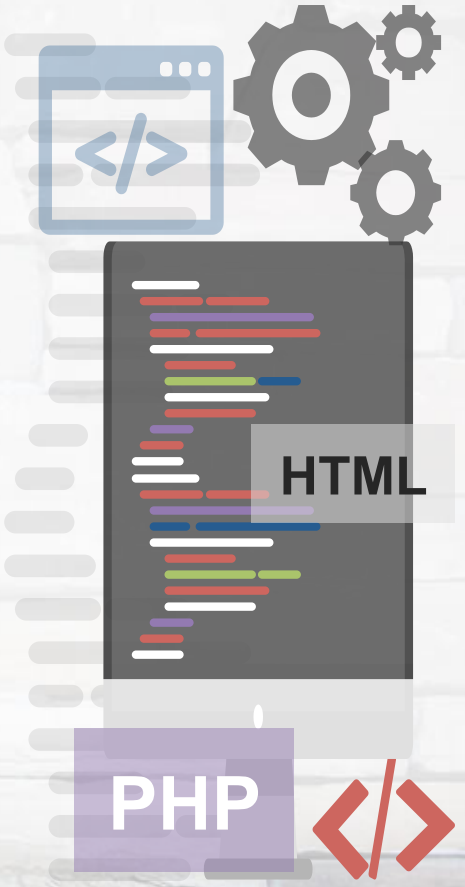
Java script variable scope

- 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.



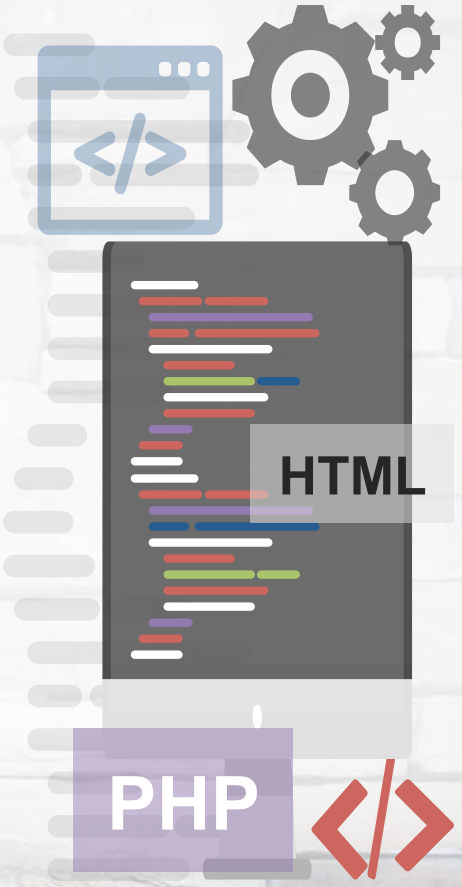
Cont'd

- **Local Variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.



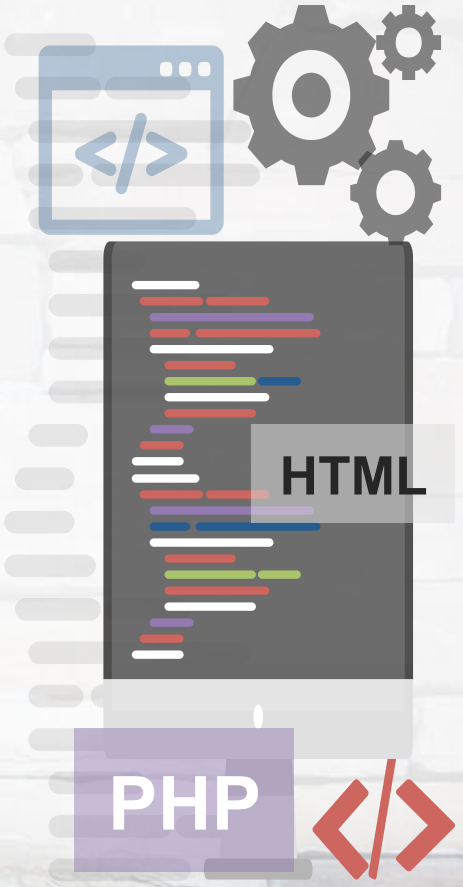
Cont'd

- The lifetime of variables starts when they are declared, and deleted when the page is closed or when the function is completed .



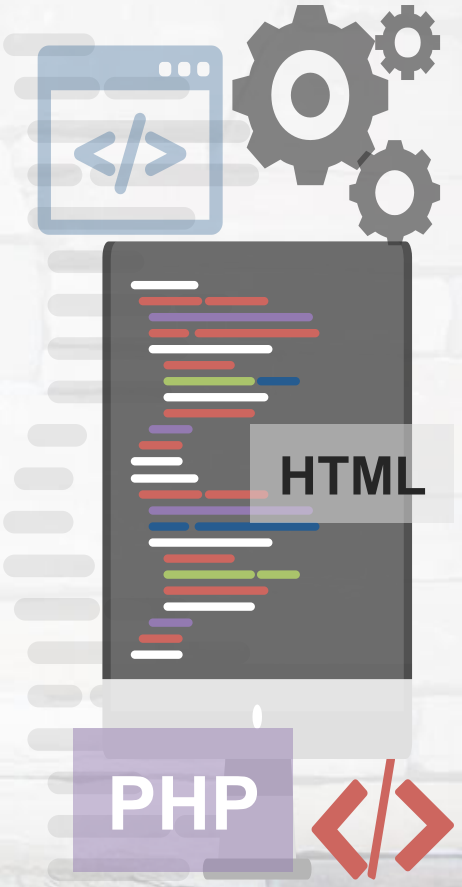
Example of variable scope

```
<script type="text/javascript">
var myVar = "global";
// Declare a global variable
function checkscope( )
 {
var myVar = "local"; // Declare a local variable
document.write(myVar);
}
</script>
```



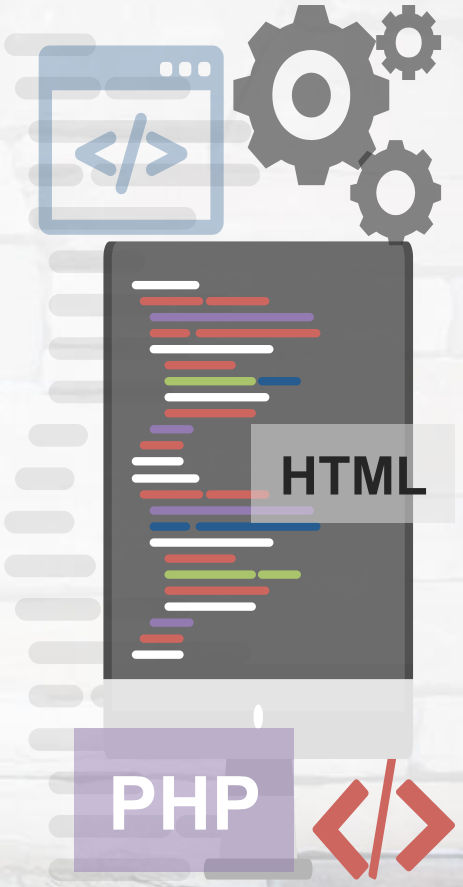
JavaScript operators

Type	Operators
Arithmetic operators	+, -, *, /, % , ++ and --
Assignment operators	=, +=, -=, *=, /= and %=
Comparison operators	== (compare only values not type), === (compare both values and type), !=, >, <, >= and <=
Conditional operator	? :
Logical operators	&&, || and !
Bitwise operators	& (and),| (or), ^ (xor), ~ (not), >>,

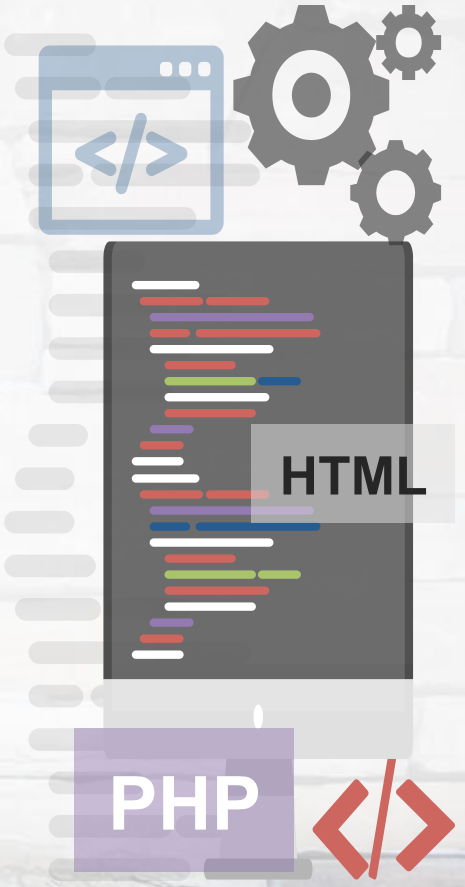


Example

```
<html>
<body>
<script type="text/javascript">
var a = 33;
var b = 10;
var c = "Test";
var linebreak = "<br />";
document.write("a + b = ");
result = a + b;
document.write(result);
document.write(linebreak);
document.write("a - b = ");
result = a - b;
document.write(result);
document.write(linebreak);
document.write("a / b = ");
result = a / b;
document.write(result);
document.write(linebreak);
document.write("a % b = ");
result = a % b;
document.write(result);
document.write(linebreak);
```



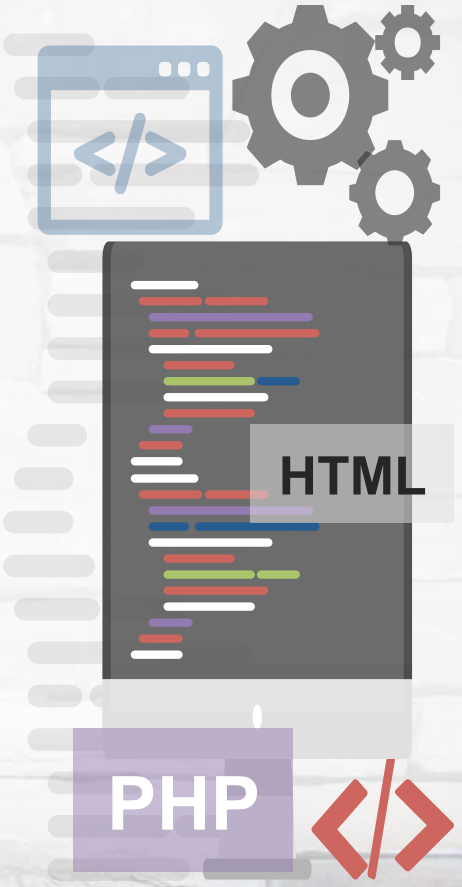
```
<html>
<body>
<script type="text/javascript">
var a = 33;
var b = 10;
var c = "Test";
var linebreak = "<br />";
document.write("a + b = ");
result = a + b;
document.write(result);
document.write(linebreak);
document.write("a - b = ");
result = a - b;
document.write(result);
document.write(linebreak);
document.write("a / b = ");
result = a / b;
document.write(result);
document.write(linebreak);
document.write("a % b = ");
result = a % b;
document.write(result);
document.write(linebreak);
document.write("a + b + c = ");
result = a + b + c;
document.write(result);
document.write(linebreak);
a = a++;
document.write("a++ = ");
result = a++;
document.write(result);
document.write(linebreak);
b = b--;
document.write("b-- = ");
result = b--;
document.write(result);
document.write(linebreak);
//-->
</script>
<p>Set the variables to different values and then
try...</p>
</body>
</html>
```



Conditional Operator (? :)

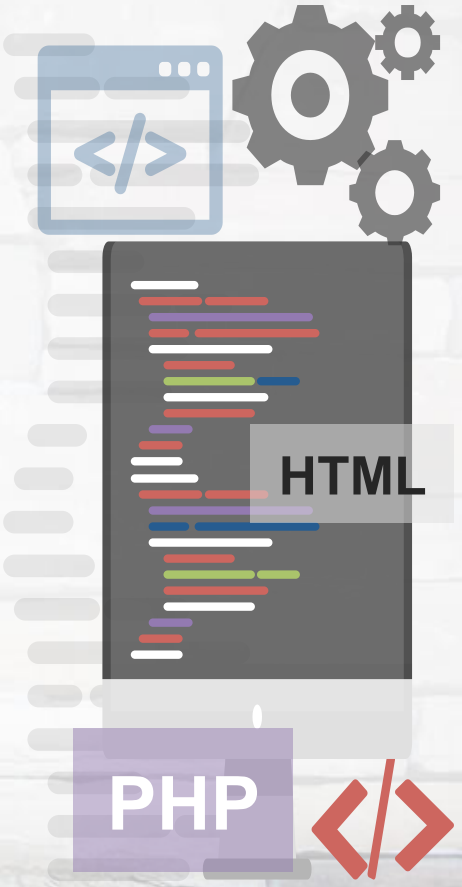
- The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

S.No	Operator and Description
1	**? : (Conditional)** If Condition is true? Then value X : Otherwise value Y

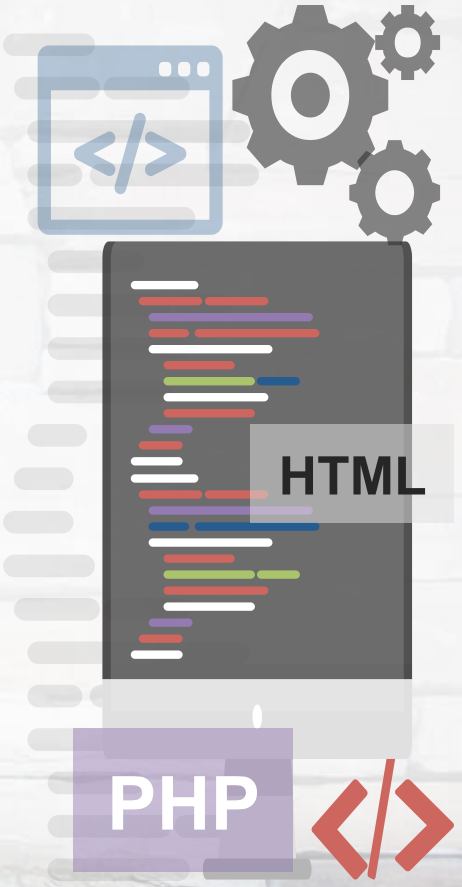


Cont'd

```
<html>
<body>
<script type="text/javascript">
var a = 10;
var b = 20;
var linebreak = "<br />";
document.write ("((a > b) ?
100 : 200) => ");
result = (a > b) ? 100 : 200;
document.write(result);
document.write(linebreak);
</script>
<p>Set the variables to
different values and
different operators and
then
try...</p>
</body>
</html>
```



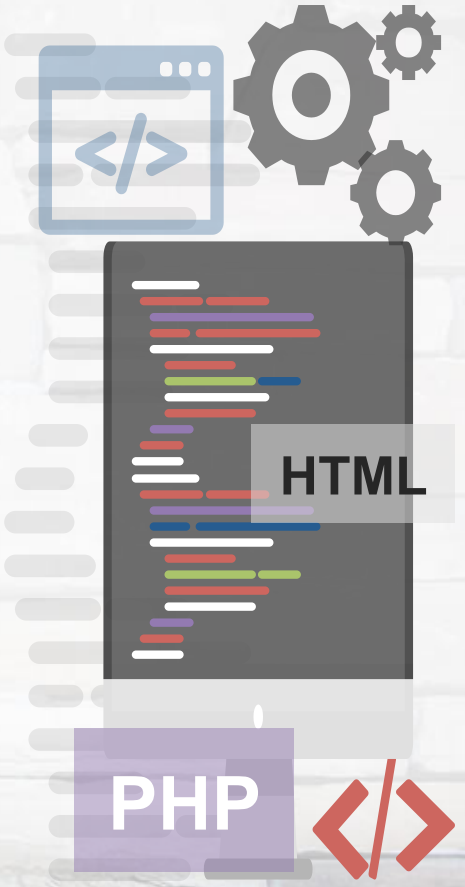
```
<html>
<head> <title></title> </head>
<body>
<script type="text/javascript">
var y = 0; Var x = 5;
x++;  y=++x+10;  y++;
document.write(x--)
document.write(",")
document.write(++y)
</script> </body>
</html>
```



Cont'd

- Output :

```
7,19
```



Cont'd

Eg:

Var x=8; var z="8" var k=8.0

x===z returns false.

x===k returns false.

x==z returns true

=== (Triple equals) is **a strict equality comparison operator in JavaScript, which returns false for the values which are not of a similar type**. This operator performs type casting for equality.

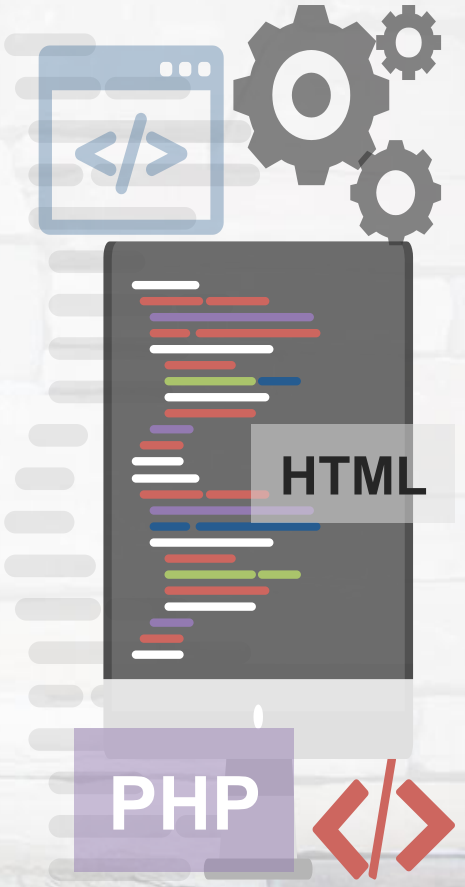
Cont'd

- Logical operators example:

  var x=60, var y=90

  (var x<=100 && var y>= 10)

Returns: true

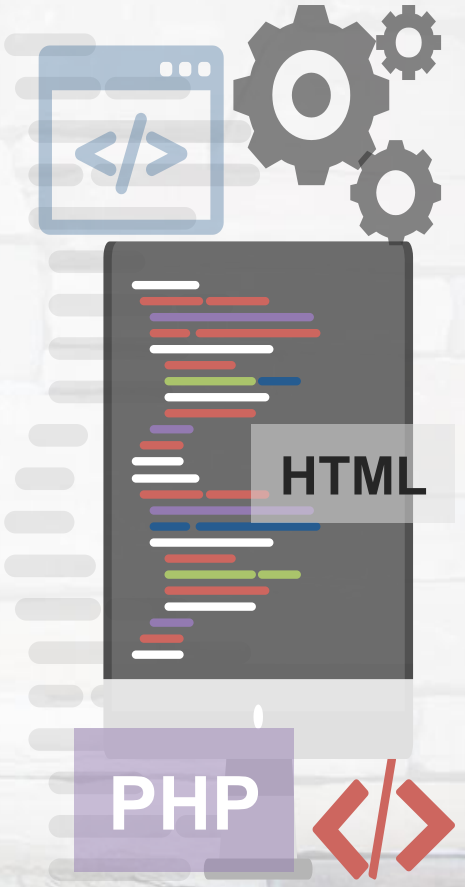


Bitwise Operators

Operator	Description	Example	Same as	Result	Decimal
&	AND	x = 5 & 1	0101 & 0001	0001	1
|	OR	x = 5 | 1	0101 | 0001	0101	5
~	NOT	x = ~ 5	~0101	1010	10
^	XOR	x = 5 ^ 1	0101 ^ 0001	0100	4
<<	Left shift	x = 5 << 1	0101 << 1	1010	10
>>	Right shift	x = 5 >> 1	0101 >> 1	0010	2

JavaScript if...else Statements

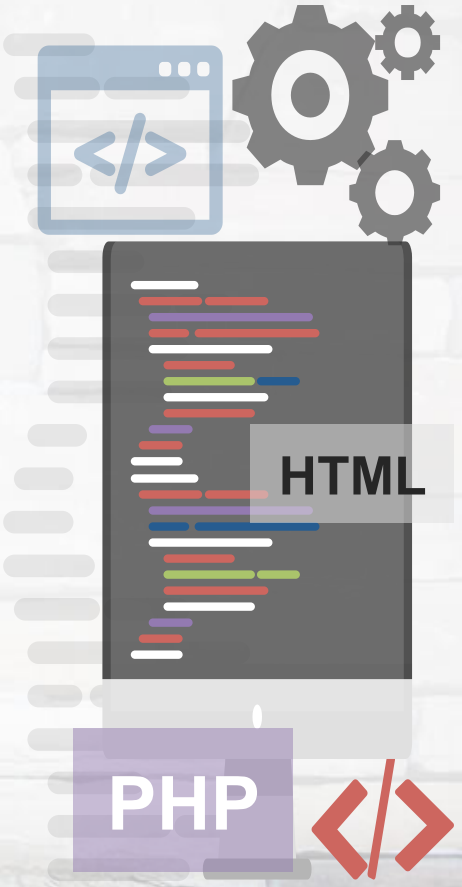
- JavaScript supports conditional statements which are used to perform different actions based on different conditions.



JavaScript if...else Statements

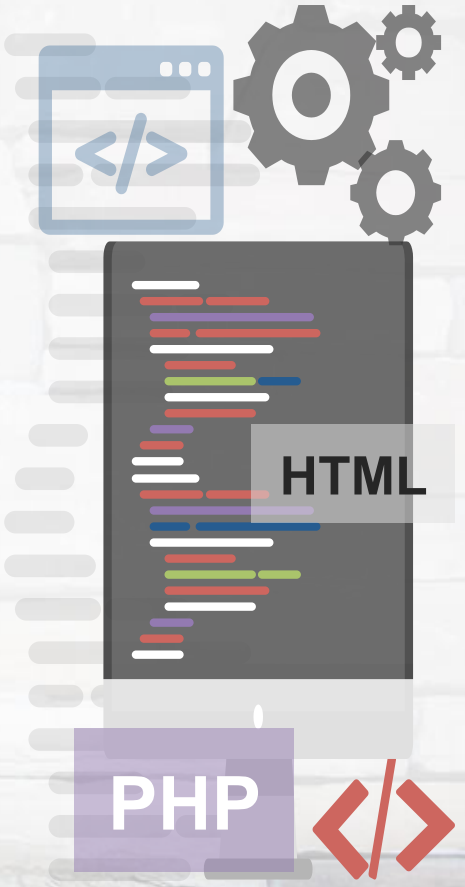
- **<u>if statement:</u>**
  - The **if** statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

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



Cont'd

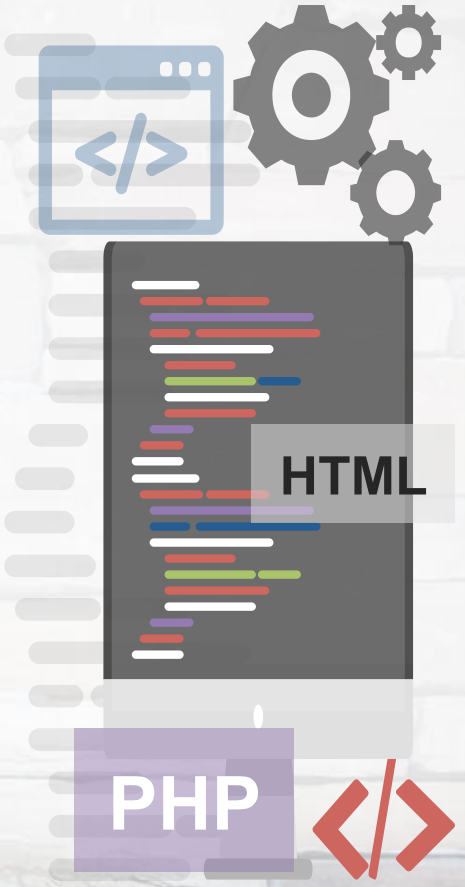
```
<script type="text/javascript">
  var age = 20;
if( age > 18 )
{
document.write("<b>Qualifies for
driving</b>");
 }
</script>
```



Cont'd

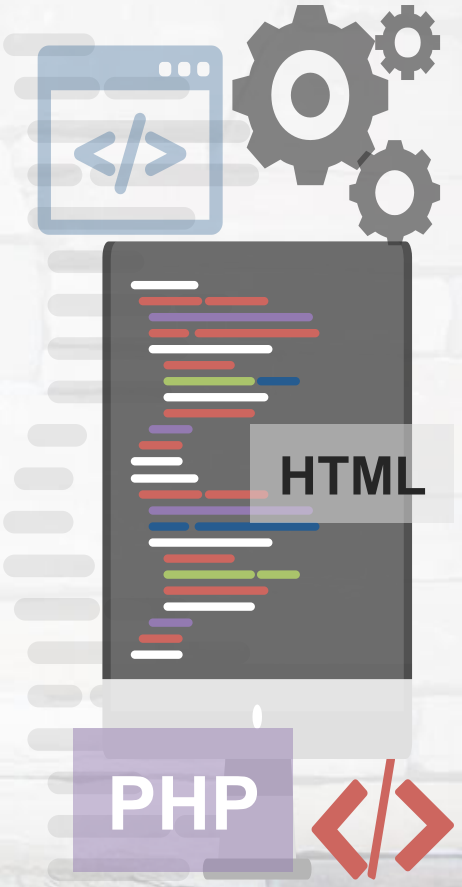
if...else statement:

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



Cont'd

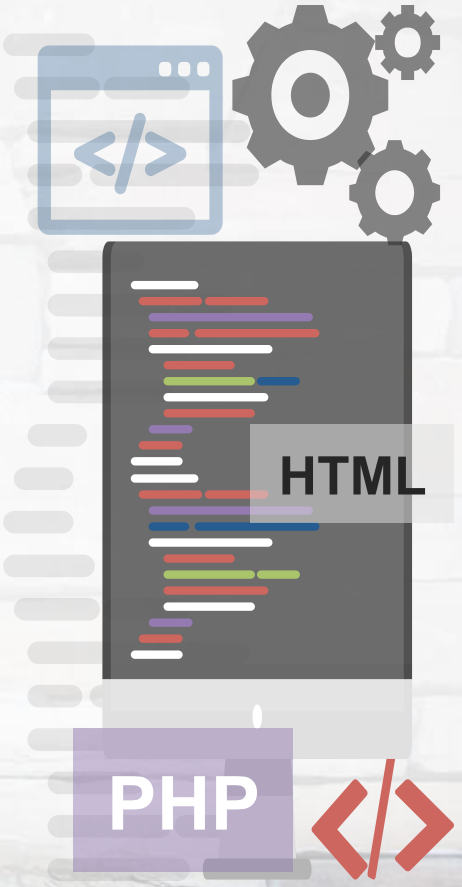
```
<script type="text/javascript">
 var age = 15;
if( age > 18 )
{  document.write("<b>Qualifies for driving</b>");  }
Else {
 document.write("<b>Does not qualify for driving</
b>");  }
</script>
```



Cont'd

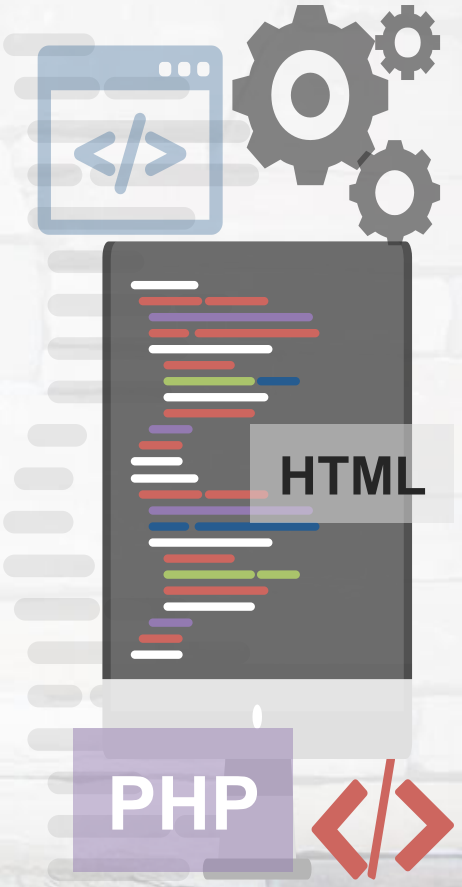
if...else if... statement:

```
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
 }
```



Cont'd

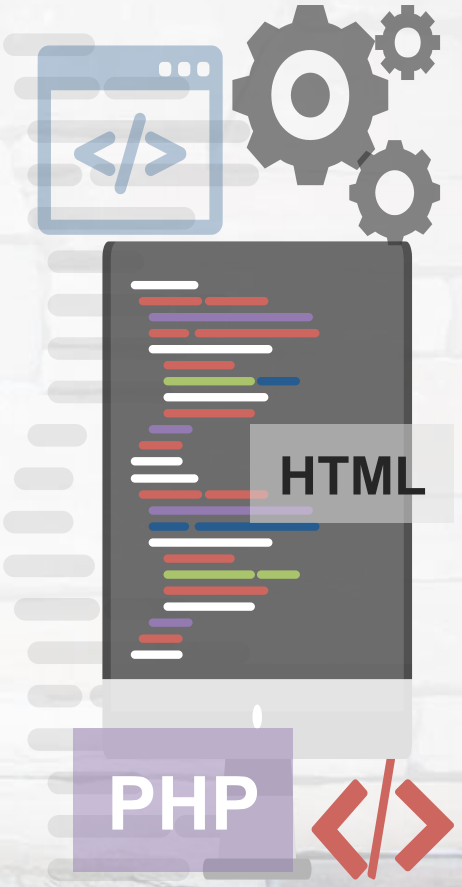
```
<script type="text/javascript">
var book = "maths";
if( book == "history" ) {
document.write("<b>History Book</b>");  }
else if( book == "maths" ) {
document.write("<b>Maths Book</b>");  }
else if( book == "economics" ) {
document.write("<b>Economics Book</b>");  }
Else {
document.write("<b>Unknown Book</b>");  }
</script>
```



JavaScript Switch Case

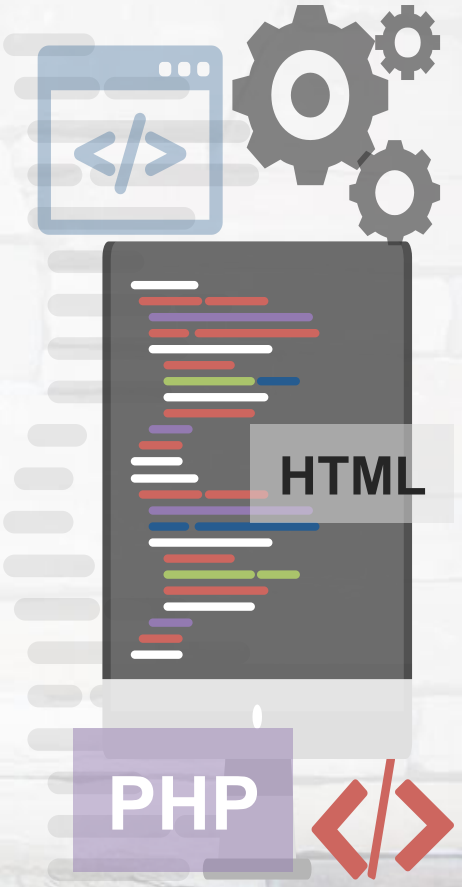
- The basic syntax of the **switch** statement is to give an expression to evaluate and several different statements to execute based on the value of the expression.

```
switch (expression) {
 case condition 1: statement(s)
                   break;
case condition 2: statement(s)
                   break;
            ...
case condition n: statement(s)
                   break;
default: statement(s)
}
```



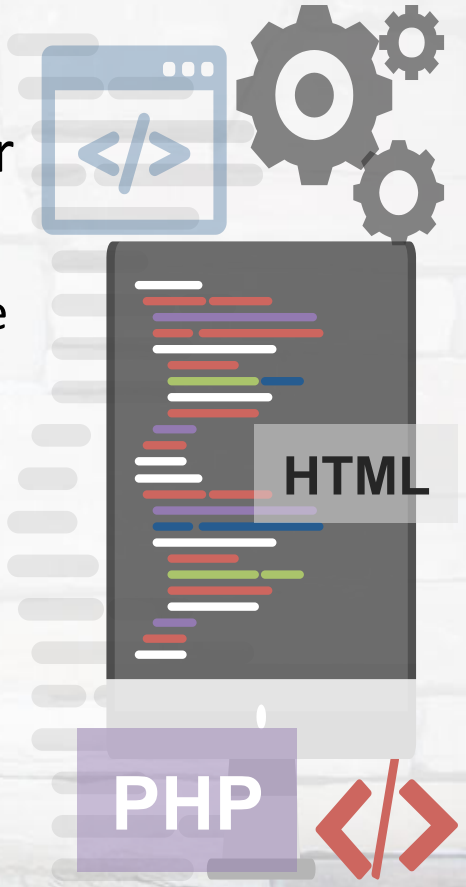
Cont'd

```
<script type="text/javascript">
 var grade='A';
document.write("Entering switch block<br />");
switch (grade) {
case 'A': document.write("Good job<br />"); break;
case 'B': document.write("Pretty good<br />"); break;
case 'C': document.write("Passed<br />"); break;
case 'D': document.write("Not so good<br />"); break;
case 'F': document.write("Failed<br />"); break;
default: document.write("Unknown grade<br />")  }
document.write("Exiting switch block");
 </script>
```



JavaScript *for* Loops

- includes the following three important parts:
  - The **loop initialization** where we initialize our counter to a starting value.
    - The initialization statement is executed before the loop begins.
  - The **test statement** which will test if the given condition is true or not. If condition is true then code given inside the loop will be executed otherwise loop will come out.
  - The **iteration statement** where you can increase or decrease your counter.

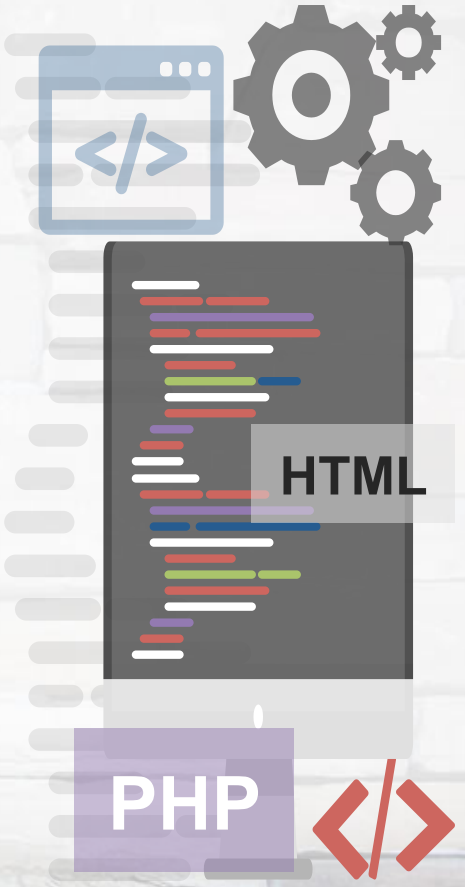


Cont'd

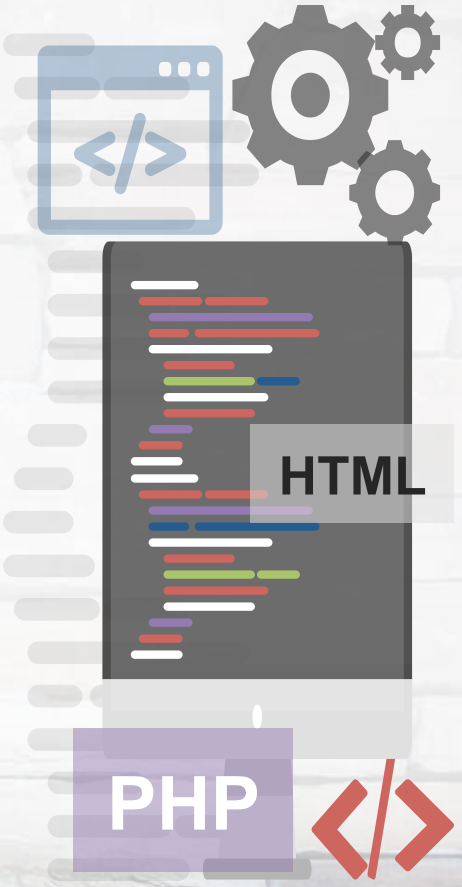
- You can put all the three parts in a single line separated by a semicolon.

for (initialization; test condition; iteration statement)

{ Statement(s) to be executed if test condition is true }

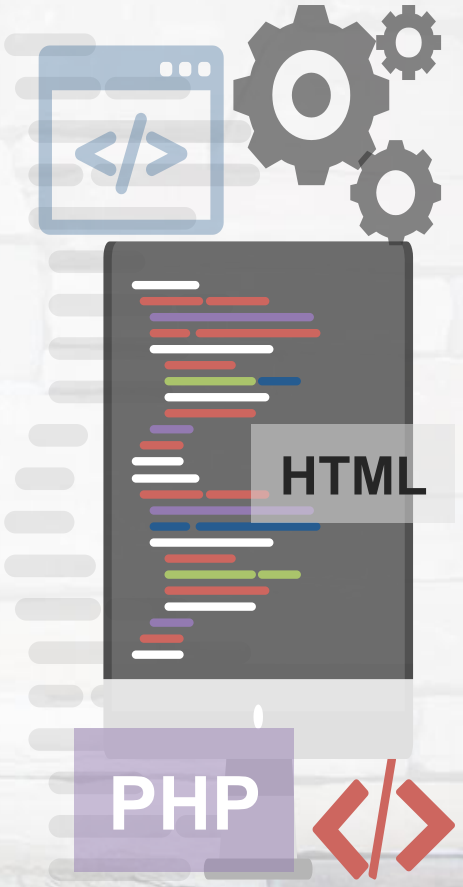


```
<html>
<head>
<script type="text/javascript”>
for(var i=5;i>=1;i--)
{   for(var j=1;j<=i;j++) {
document.write("*”) }
document.write('<br/>’) }
</script>
</head>
<body>
</body> </html>
```



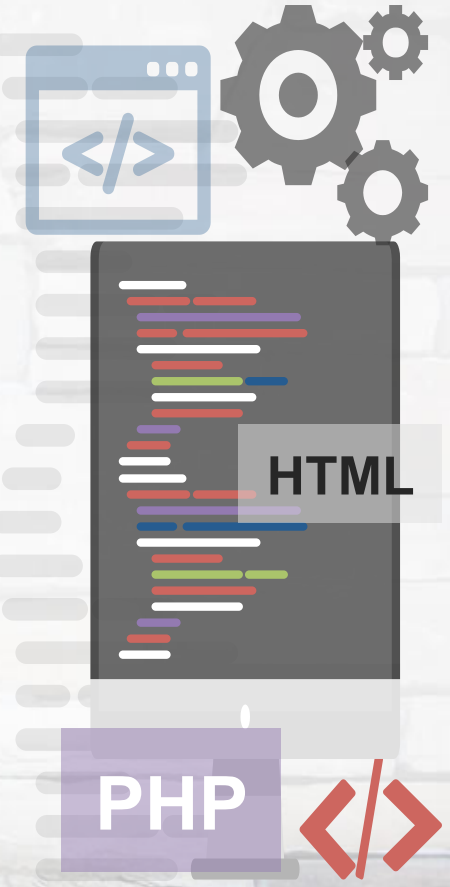
output

```
*****
****
***
**
*
```



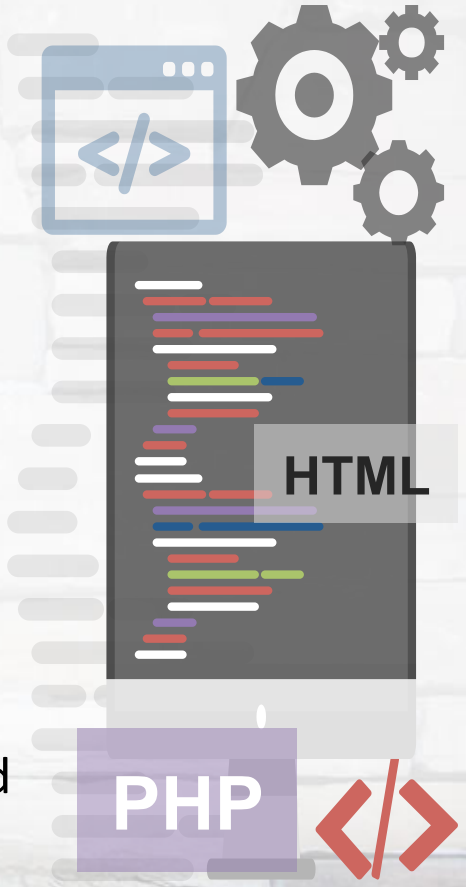
JavaScript Functions

- a group of reusable code which can be called anywhere in your program.
- executed only by an event or by a call to that function.
- You may call a function from anywhere within the page (or even from other pages if the function is embedded in an external .js file).
- Functions can be defined either <head> or <body> section
  - As a convention, they are typically defined in the <head> section



JavaScript Function Syntax

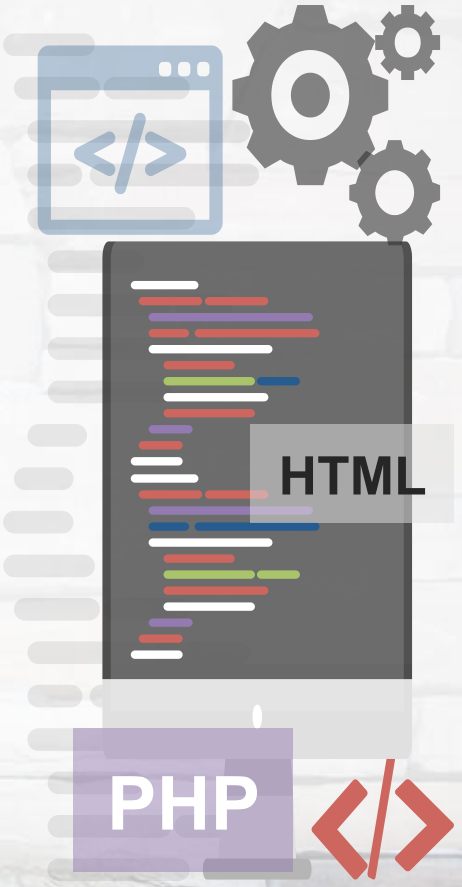
- A JavaScript function is defined with the **function** keyword, followed by a **unique function name**, followed by parentheses **()**, a list of parameters (that might be empty), and a statement block surrounded by curly braces.
- Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
- The parentheses may include parameter names separated by commas: **(*parameter1, parameter2, ...*)**



Cont'd

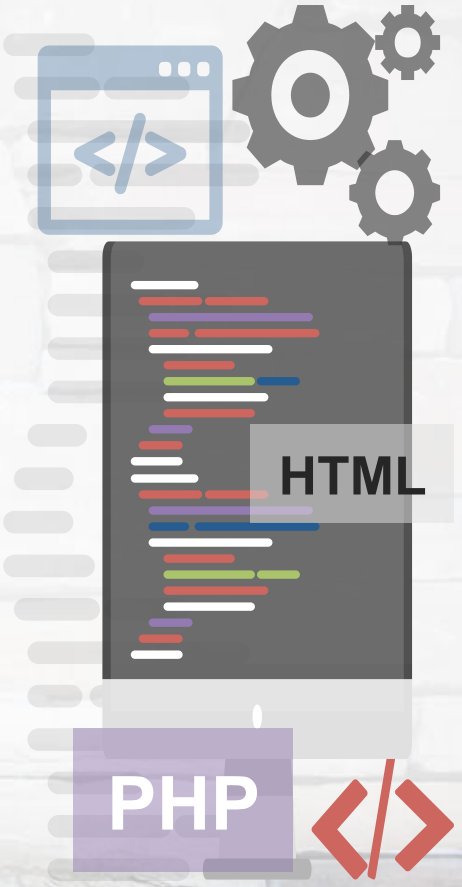
Syntax:

```
Function functionName(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)

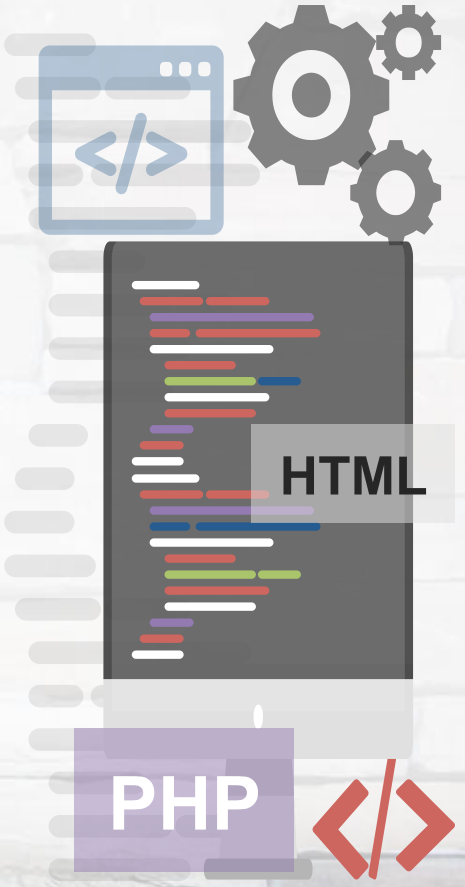


Cont'd

To invoke a function somewhere later in the script, you would simple need to write the name of that function as follows:

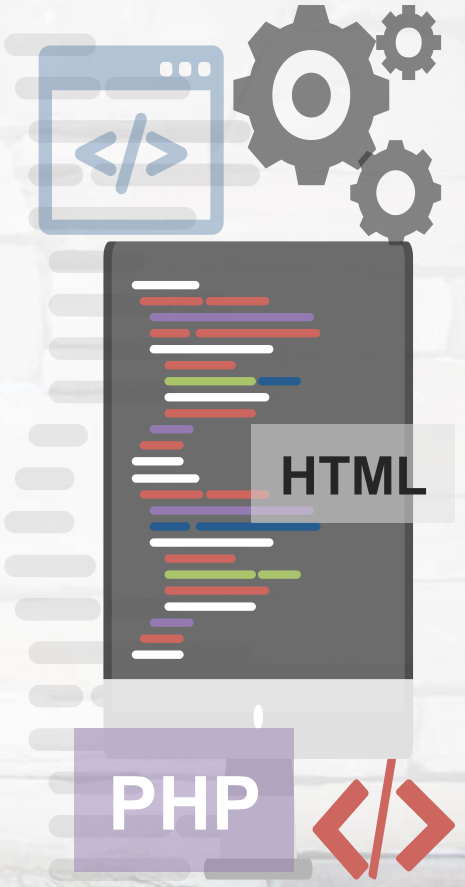
E.g.

```
<html> <head>
<script type="text/javascript">
function displaymessage() { alert("Hello World!”) }
</script> </head> <body> <form>
<input type="button" value="Click me!"
onclick="displaymessage()" >
</form></body>
</html>
```



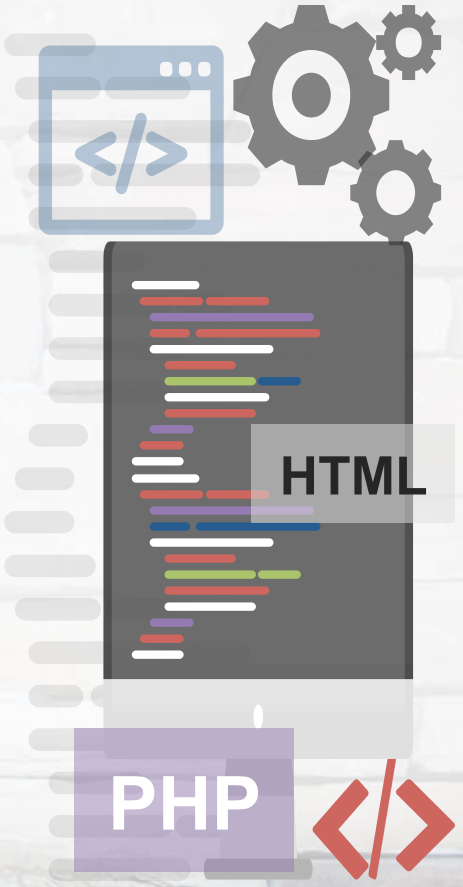
Cont'd

```
<script type="text/javascript">
        sayHello();  </script>
<script type="text/javascript">
        function sayHello(name, age){
    alert( name + " is " + age + " years old."); }
</script>
<script type="text/javascript">
            sayHello('Zara', 7 );
</script>
```



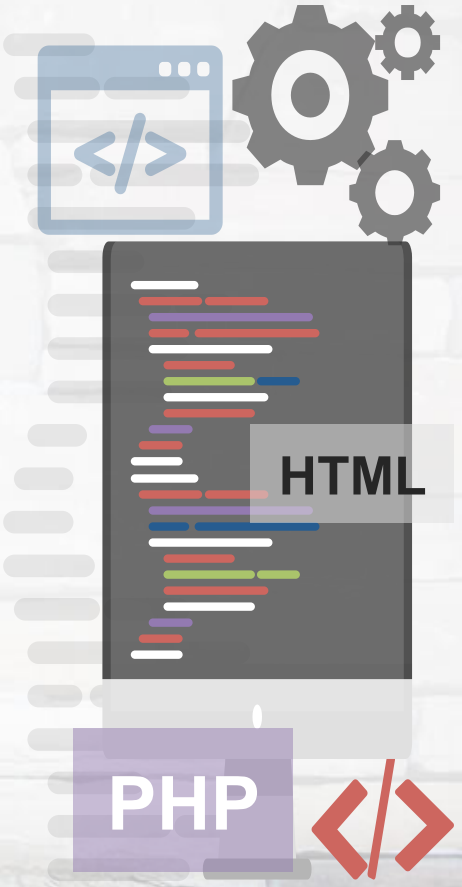
JavaScript event

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



Example of events are like

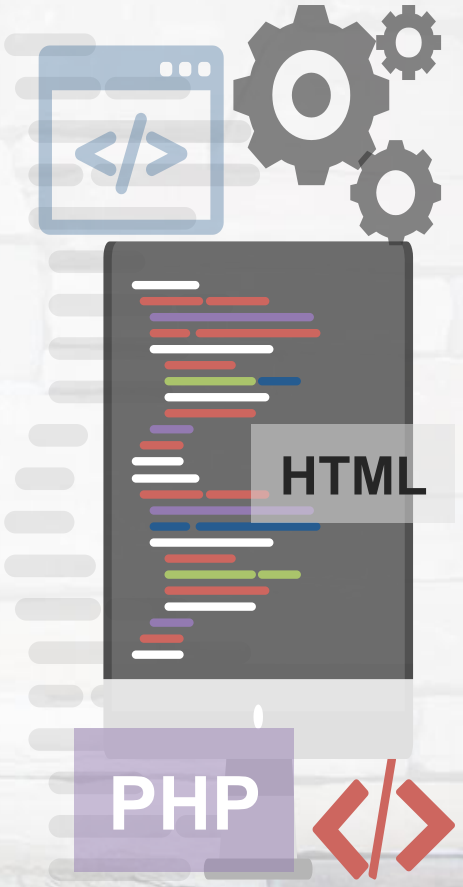
- Mouse click
- A web page or an image loading
- Selecting an input box in an HTML form
- When the user clicks a button (Submitting an HTML form)
- pressing any key,
- closing window,
- resizing window
- etc.



Cont'd

- Attributes are inserted into HTML tags to define events and event handlers
  - With single quotes:

  <some-HTML-element some-event='**some JavaScript'**>



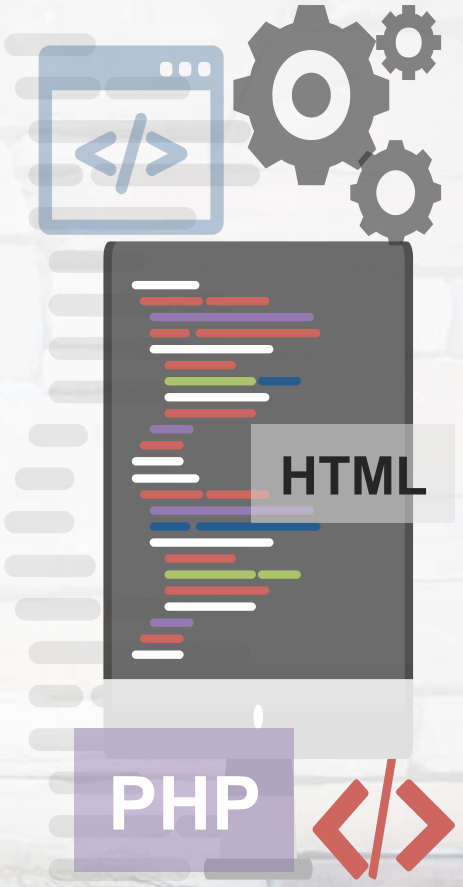
Cont'd

– With double quotes:

<some-HTML-element some-event="**some JavaScript**">

- **Example**

<button onclick='getElementById("demo").innerHTML=Date()'>The time is?</button>



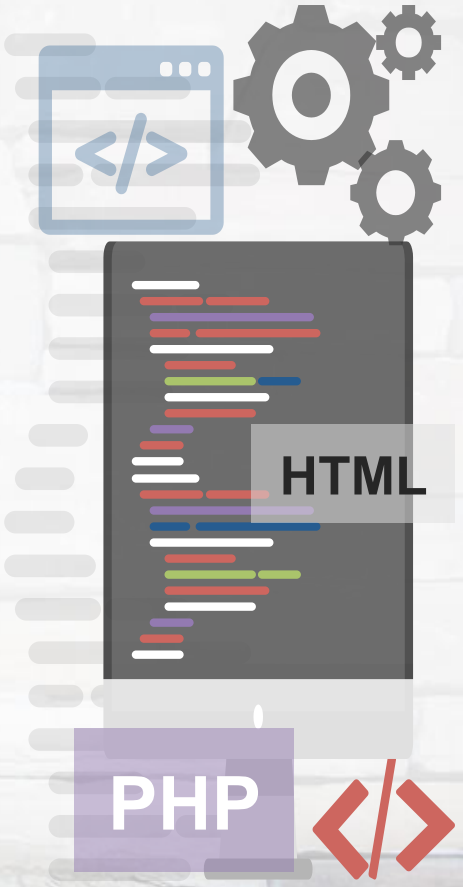
Cont'd

Event	Value	Description
onchange	script	Script runs when the element changes
onsubmit	script	Script runs when the form is submitted
onreset	script	Script runs when the form is reset
onselect	script	Script runs when the element is selected
onblur	script	Script runs when the element loses focus
onfocus	script	Script runs when the element gets focus
onkeydown	script	Script runs when key is pressed
onkeypress	script	Script runs when key is pressed and released

onkeyup	script	Script runs when key is released
onclick	script	Script runs when a mouse click
ondblclick	script	Script runs when a mouse double-click
onmousedown	script	Script runs when mouse button is pressed
onmousemove	script	Script runs when mouse pointer moves
onmouseout	script	Script runs when mouse pointer moves out of an element
onmouseover	script	Script runs when mouse pointer moves over an element
onmouseup	script	Script runs when mouse button is released
onkeyup	script	Script runs when key is released

Cont'd

```
<html>
<head>
<script type="text/javascript">
function upperCase() {
var x=document.getElementById("fname").value
document.getElementById("fname").value=x.toUpperCase() }
</script>
</head>
<body> Enter your name:
<input type="text" id="fname" onchange="upperCase()">
</body>
</html>
```



```
<html>
<head> <script type="text/javascript">
    function over() {
   alert("Mouse Over");  }
    function out()    {
  alert("Mouse Out");  }
  </script>  </head>
<body>
<div onmouseover="over()"
onmouseout="out()">
 <h2> This is inside the division </h2> </div>
 </body>
</html>
```