Lecture 09: JavaScript Functions
ICT1153

inputs
Function
output

Buddhika Gayashani Jayaneththi
Department of ICT
Faculty of Technology
University of Ruhuna

JavaScript Functions

- A function is a reusable code-block that will be executed by an event, or when the function is called.

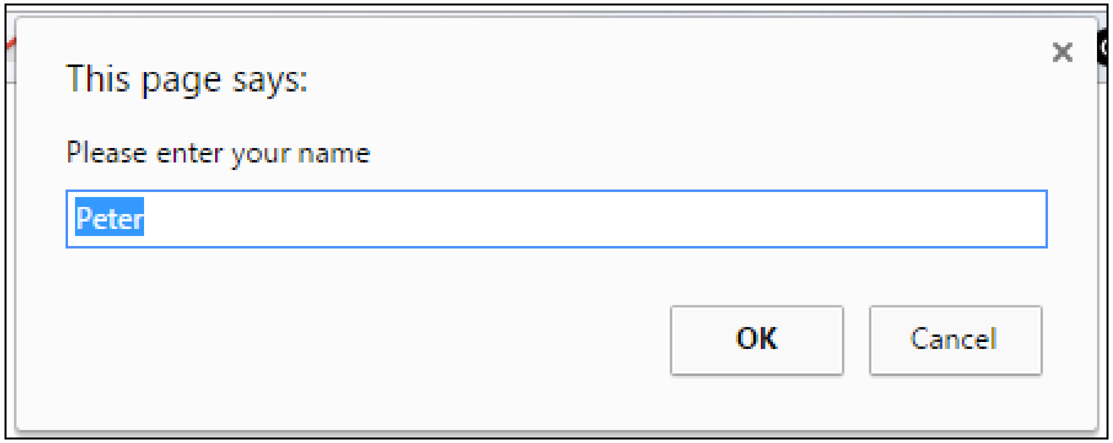
Why functions?

- You can reuse code: Define the code once and use it many times.
- You can use the same code many times with different arguments, to produce different results.

- JavaScript functions can be categorized into two types,
  - Build-in functions.
  - User defined functions.

Build-in Functions

- Build in functions can be used for several predefined operations.
- Example:
  - window.alert("Message");



This page says:

Please enter your name

Peter

User Defined Functions

function *function_name*(**parameter1,parameter2,parameter3)**

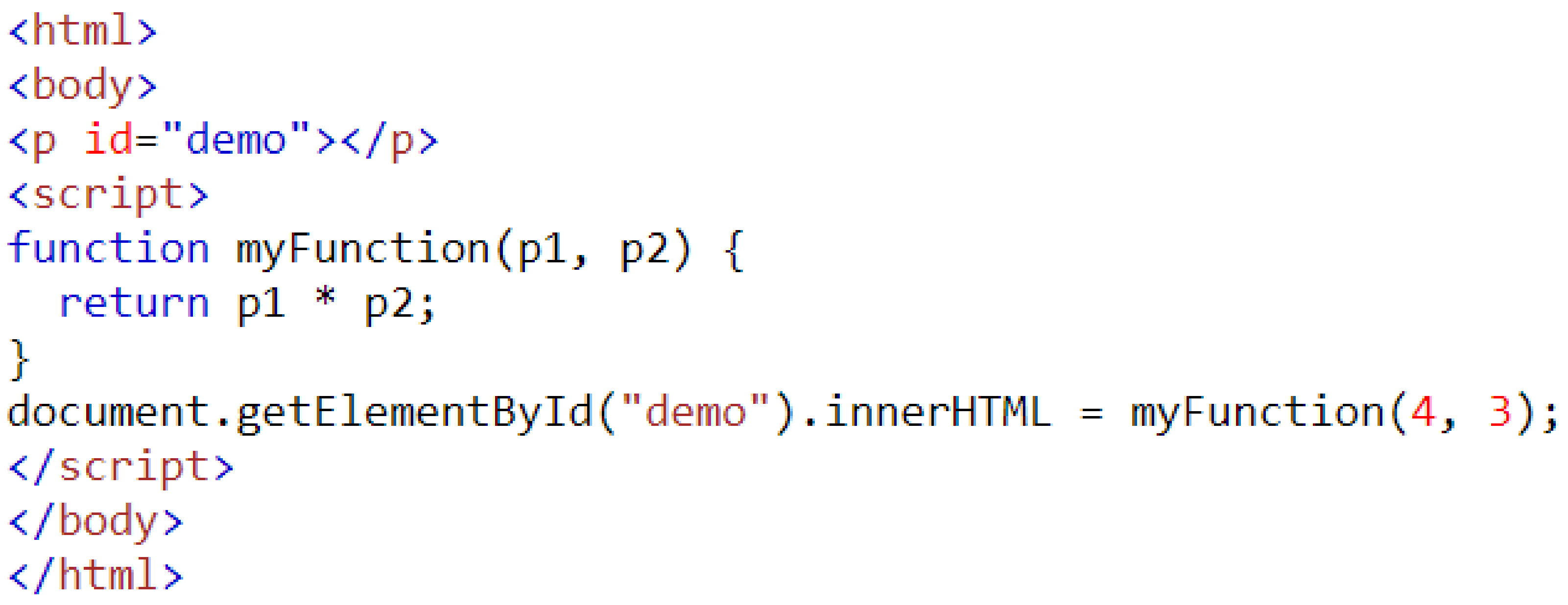
{

        code to be executed;

}

- JavaScript functions starts with **function** keyword.
- 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, ...)***
- The code to be executed, by the function, is placed inside curly brackets: {}

User Defined Functions



```
<html>
<body>
<p id="demo"></p>
<script>
function myFunction(p1, p2) {
  return p1 * p2;
}
document.getElementById("demo").innerHTML = myFunction(4, 3);
</script>
</body>
</html>
```

- Function **arguments** are the **values** received by the function when it is invoked.
- In here arguments are: 4 and 3.

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) by a JavaScript code.

Functions without Arguments

- Function **parameters** are the names listed in the function definition.
- Function **arguments** are the real values passed to (and received by) the function.

```
function function_name() {
        code to be executed;
}
```

Functions without Arguments

```
<html>
<body>
<button id="btn1" onclick=myFunction()>Click Me</button>
<script>
function myFunction() {
  window.alert("you clicked the button");
}
</script>
</body>
</html>
```

Functions with Arguments

```
function function_name(parameter1, parameter2) {
        code to be executed
}
```

Example

```
function sayhello(name){
        window.alert("Hello "+name);
}
sayhello("Peter");
```

Functions with Arguments

```
<html>
<body>
<button id="btn1" onclick=sayhello("Peter")>Display</button>
<script>
function sayhello(name) {
  window.alert("Hello"+" "+name);
}
</script>
</body>
</html>
```

Functions with a Return Value

- When JavaScript reaches a return statement, the function will stop executing.

- Functions often compute a return value. The return value is "returned" back to the "caller“.

Functions with a Return Value

```
<html>
<body>
<p id="demo"></p>
<script>
function myFunction(a, b) {
  return a * b;
}
var x = myFunction(4, 3);
document.getElementById("demo").innerHTML = x;
</script>
</body>
</html>
```

Exercise

- Write down the JavaScript function to return the speed of a vehicle which has a constant velocity ($s = ut$)('u' and 't' are parameters of the function). You can pass values for the parameters while calling the function. Display the answer on the browser.

Answer

```
<script>
        function findSpeed(u,t){
                var s=u*t;
                return s;
        }
        var speed=findSpeed(10,2);
        document.getElementById("demo").innerHTML=speed;
</script>
```

Decision Making

- Conditions are comparisons between data / variables.
  - Is A bigger than B?
  - Is X equal to Y?
  - Has the kettle boiled?
    - If so, then pour water into the cup;
    - otherwise continue to wait.

Conditions: If- else

- Use to perform different actions based on different conditions.

if	**if..else**	**if..else if..else**

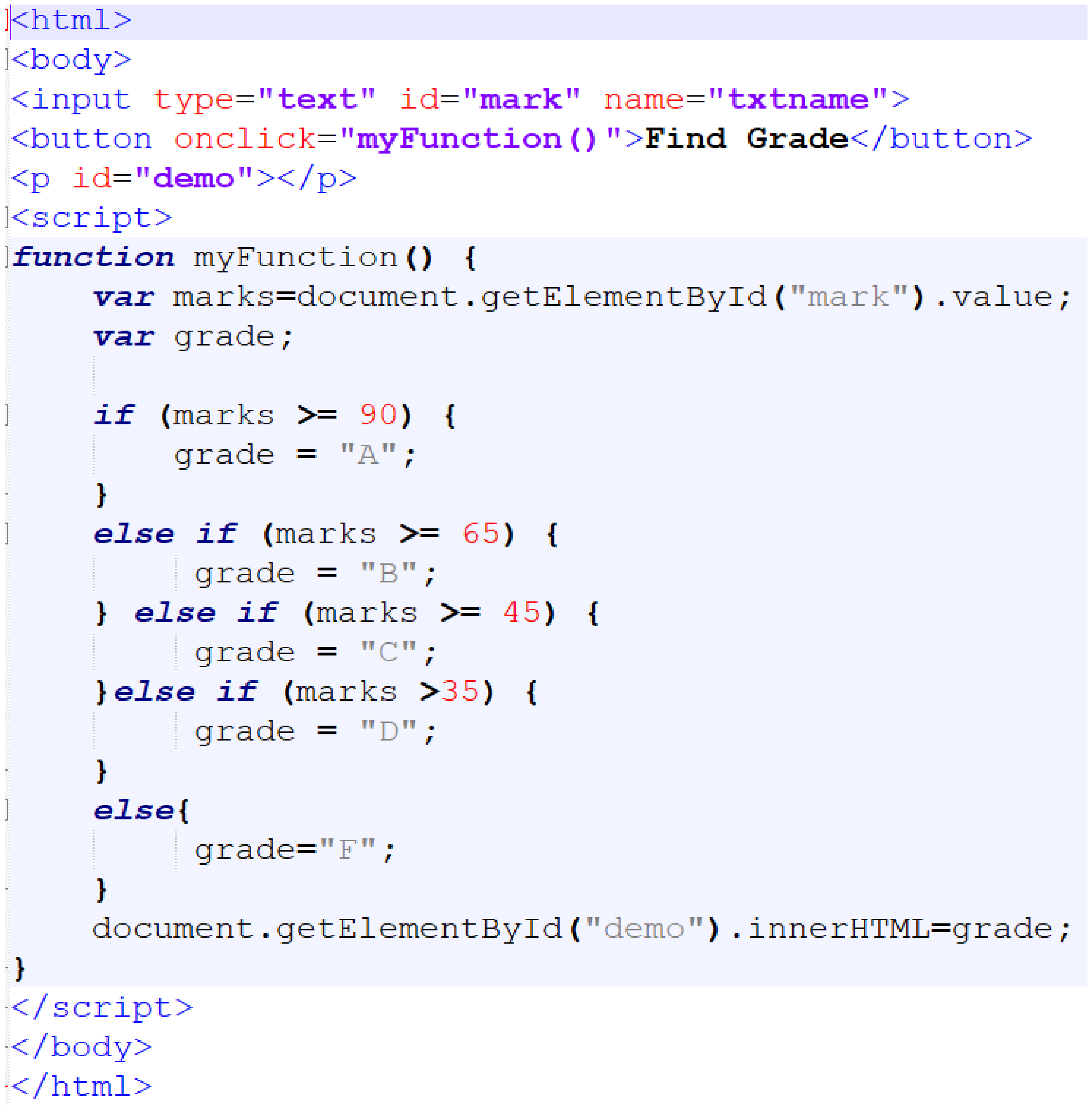
if (condition)
{
    Statement ;
}
	if (condition)
	{
	    Statement A ;
	}
	else
	{
	    Statement B;
	}
		if (condition 1)
		{
		    Statement A ;
		}
		else if (condition 2)
		{
		    Statement B ;
		}
		else
		{
		    statement C ;
		}

Exercise

- Write down a code to find out the grade by using the given data.
- There should be a text field to get the marks and when the user enters the marks and click the "Find Grade" button the grade should be displayed.(use **.value** to get the user entered value in the text field)

Marks	Grade
>= 90	A
>=65	B
>=45	C
>35	D
<=35	F

Answer



```
<html>
<body>
<input type="text" id="mark" name="txtname">
<button onclick="myFunction()">Find Grade</button>
<p id="demo"></p>
<script>
function myFunction() {
    var marks=document.getElementById("mark").value;
    var grade;

    if (marks >= 90) {
        grade = "A";
    }
    else if (marks >= 65) {
         grade = "B";
    } else if (marks >= 45) {
         grade = "C";
    }else if (marks >35) {
         grade = "D";
    }
    else{
         grade="F";
    }
    document.getElementById("demo").innerHTML=grade;
}
</script>
</body>
</html>
```

Switch Statement

- The switch statement is used to perform different actions based on different conditions.
- It tests the value of a given variable or expression against a list of case values.
  - When a match is found, a block of statements associated with that case is executed.
- A switch can have any number of cases.
- However, there should be only one **default** handler. If there is no match, the default code block is executed.

Note:

- **Logical operators cannot be used with switch statement**
  - Ex: case k>=20:

Switch Statement - Syntax

```
switch (expression)
{
 case value1:
      code segment1;
      break;
 case value2:
      code segment2;
      break;
default:
      default code segment;
}
```

How the Switch Statement Works ?

- 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.

The Break Keyword

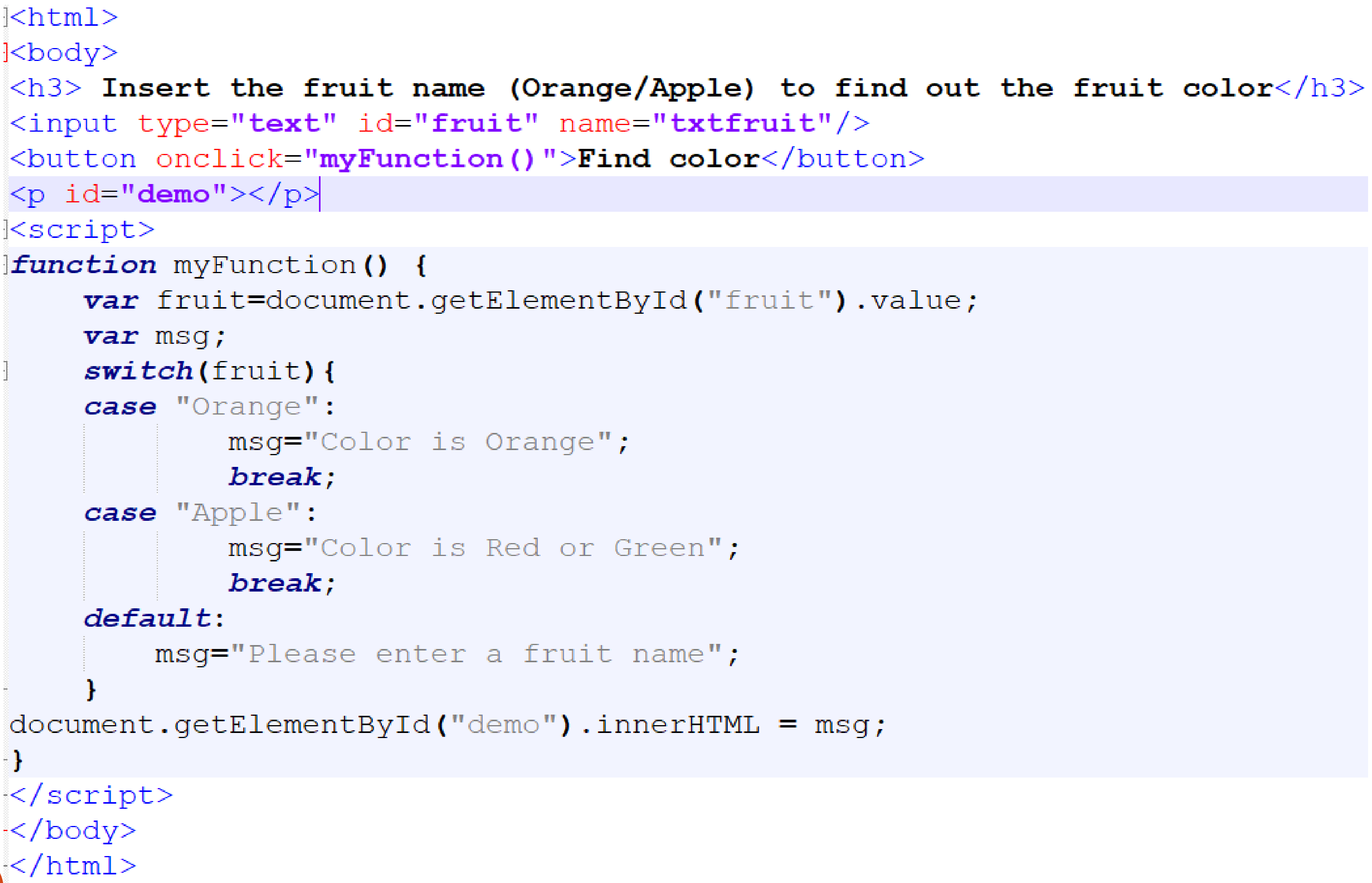
- When JavaScript reaches a break keyword, it breaks out of the switch block.
- This will stop the execution inside the switch block.
- It is not necessary to break the last case in a switch block. The block breaks (ends) there anyway.

Exercise

- Write down a code for the given scenario
  - If the fruit name is Orange, then print "color is orange"
  - If the fruit name is Apple, then print "color is red"
  - Otherwise, print "please enter a fruit name"

There should be a text field to get the name of the fruit and when the user enters the name of the fruit and click the "Find Color" button switch statement should be executed. You can write a function including the switch statement.

Answer



```
<html>
<body>
<h3> Insert the fruit name (Orange/Apple) to find out the fruit color</h3>
<input type="text" id="fruit" name="txtfruit"/>
<button onclick="myFunction()">Find color</button>
<p id="demo"></p>
<script>
function myFunction() {
    var fruit=document.getElementById("fruit").value;
    var msg;
    switch(fruit){
    case "Orange":
            msg="Color is Orange";
            break;
    case "Apple":
            msg="Color is Red or Green";
            break;
    default:
        msg="Please enter a fruit name";
    }
document.getElementById("demo").innerHTML = msg;
}
</script>
</body>
</html>
```

Loops

- Used to execute the same block of code a specified number of times.
- JavaScript supports different kinds of loops.
  - **for** - loops through a block of code a number of times.
  - **while** - loops through a block of code while a specified condition is true.
  - **do/while** - loops through a block of code while a specified condition is true.
    - Executes the content of the loop **once** before checking the condition of the while.

For Loop

For loop is used to execute the same code block a specified time.

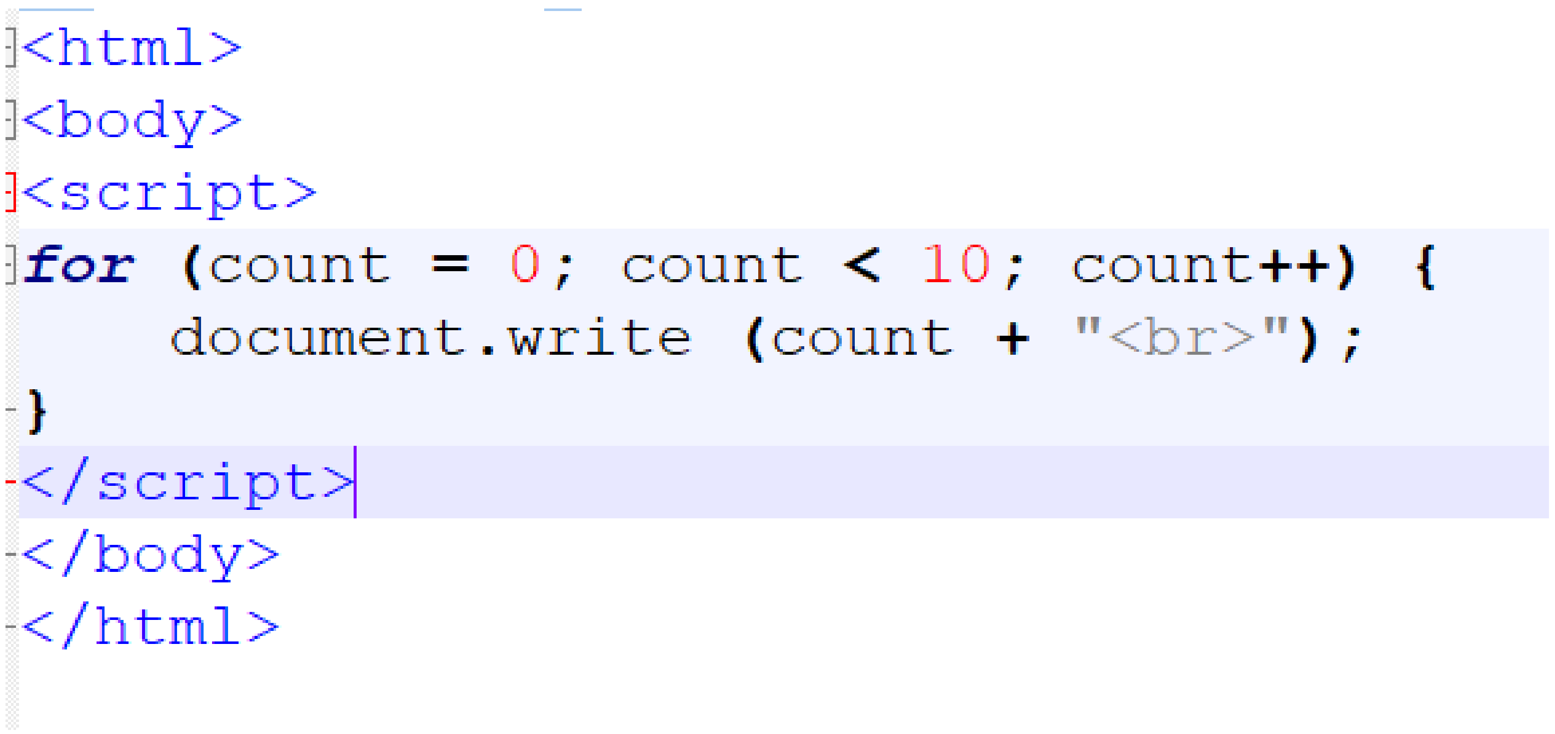
Syntax

```
for ( initialization; condition; increment counter) {
        statements ;
}
```

Example

```
for (count = 0; count < 10; count++) {
        document.write (count + "<br>");
}
```

For Loop



```
<html>
<body>
<script>
for (count = 0; count < 10; count++) {
        document.write (count + "<br>");
}
</script>
</body>
</html>
```

While Loop

Loops through a block of code as long as a specified condition is true.

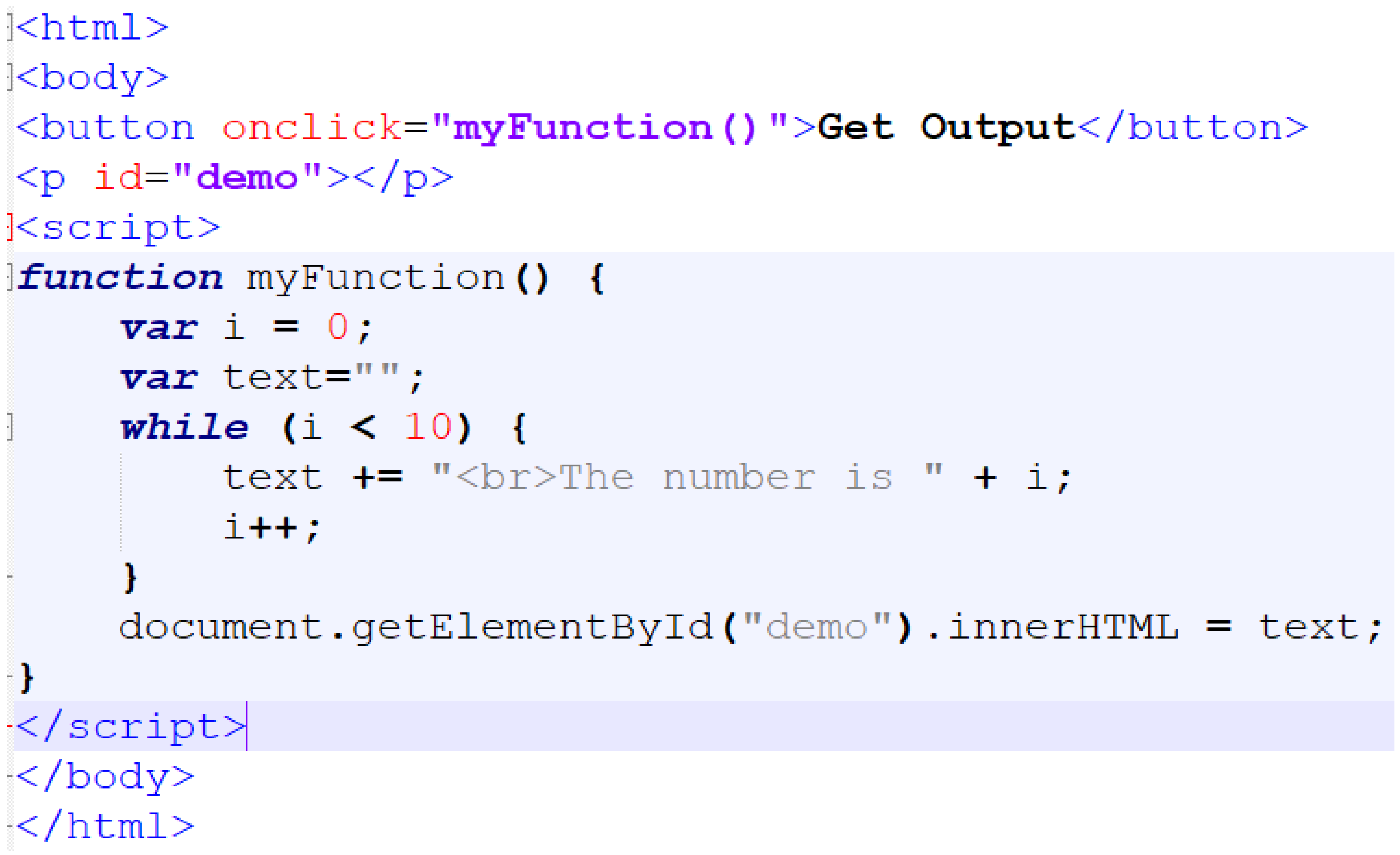
Syntax

```
while (condition) {
        code block to be executed;
        increment/decrement;
}
```

While Loop Example

```
var count = 0;
while (count < 10)
{
        document.write (count+“<br>”);
        count ++;
}
```

While Loop Example



```
<html>
<body>
<button onclick="myFunction()">Get Output</button>
<p id="demo"></p>
<script>
function myFunction() {
    var i = 0;
    var text="";
    while (i < 10) {
        text += "<br>The number is " + i;
        i++;
    }
    document.getElementById("demo").innerHTML = text;
}
</script>
</body>
</html>
```

Do-While Loop

- The do/while loop is a variant of the while loop.
- This loop will execute the code block **once**, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

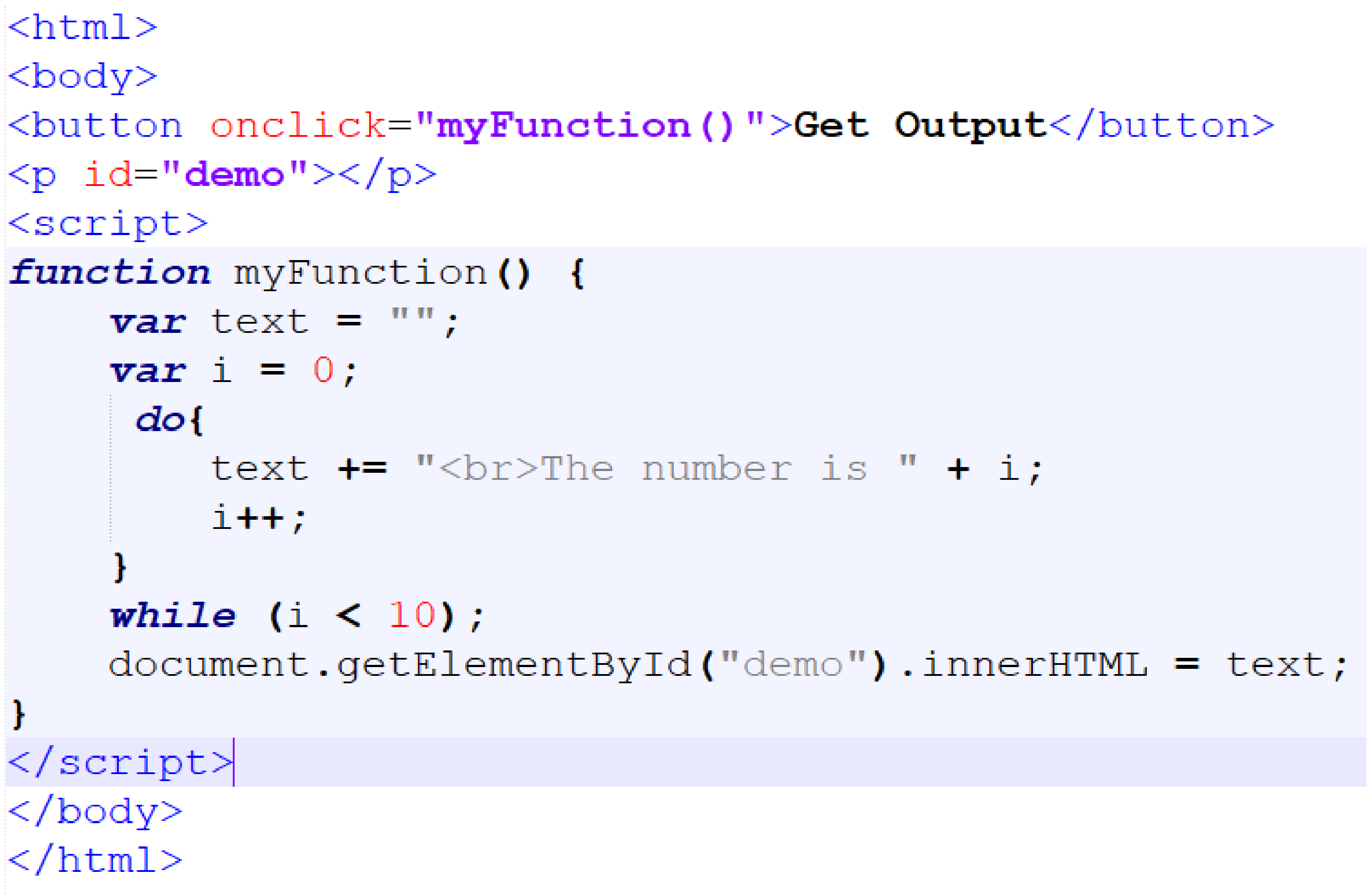
```
        do {
                statements ;
                increment/decrement;
        }
        while (condition);
```

Do-While Loop Example

Example

```
var count = 0;
do {
        document.write (count+“<br>”);
        count ++;
}
while (count < 10)
```

Do-While Loop Example



```
<html>
<body>
<button onclick="myFunction()">Get Output</button>
<p id="demo"></p>
<script>
function myFunction() {
    var text = "";
    var i = 0;
     do{
        text += "<br>The number is " + i;
        i++;
    }
    while (i < 10);
    document.getElementById("demo").innerHTML = text;
}
</script>
</body>
</html>
```

Break Statement

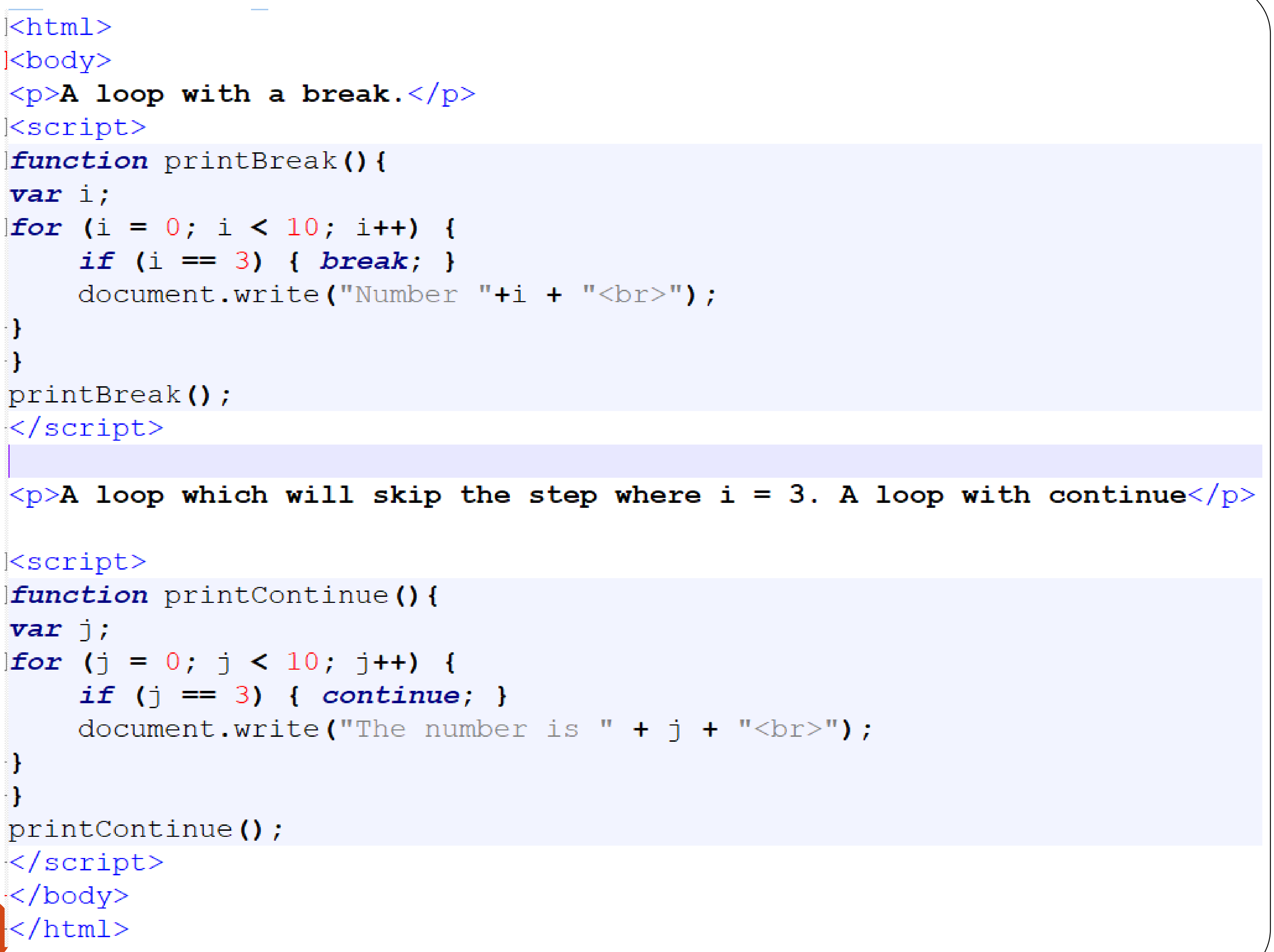
- The **break statement** is used to break the current loop and "jumps out" of a loop.

```
for (i=0;i<=10;i++)
{
            if (i==3){
                      break;
                      }
           document.write("The number is " + i);
           document.write("<br />");
 }
```

Continue Statement

- The **continue statement** is used to break one iteration (in the loop) and continues with the next iteration.

```
for (i=0;i<=10;i++)
{
          if (i==3){
                    continue;
                    }
          document.write("The number is " + i);
          document.write("<br />");
}
```



```
<html>
<body>
<p>A loop with a break.</p>
<script>
function printBreak(){
var i;
for (i = 0; i < 10; i++) {
    if (i == 3) { break; }
    document.write("Number "+i + "<br>");
}
}
printBreak();
</script>

<p>A loop which will skip the step where i = 3. A loop with continue</p>

<script>
function printContinue(){
var j;
for (j = 0; j < 10; j++) {
    if (j == 3) { continue; }
    document.write("The number is " + j + "<br>");
}
}
printContinue();
</script>
</body>
</html>
```

Questions…