JavaScript Break and Continue Statements

1 July 2024

The break statement "jumpsout" of a loop.
The continue statement "jumpsover" one iteration in the loop.

The Break Statement
It was used to "jump out" of a switch() statement.

```
for(leti= 0;i<10;i++)
    {
    if(i===3)
        {
            break;
        }
        console.log("Thenumberis:",i);
  }
```

Output:
PS C:\Users\dell\Documents\Javascript.js> node
"c:\Users\dell\Documents\Javascript.js\breakandcontinue.js"
Thenumberis:0 Thenumberis:1 Thenumberis: 2

The Continue Statement
The continue statement breaks one iteration (in the loop),if a specified condition occurs, and continues with the next iteration in the loop.

```
for(leti= 0;i<10;i++)
  {
  if(i===3)
      {
         continue;
      }
      console.log("Thenumberis:",i);
 }
```

Output:
PS C:\Users\dell\Documents\Javascript.js> node
"c:\Users\dell\Documents\Javascript.js\breakandcontinue.js"
Thenumberis:0 Thenumberis:1 Thenumberis:2 Thenumberis:4 Thenumberis:5
Thenumberis:6 Thenumberis:7 Thenumberis:8 Thenumberis: 9

Nested Loops JavaScript

A composition of loops is called a nested loop. The most common type of nested loops will be one loop inside another.The first loop is usually called the outer loop while the second loop is called the inner loop.

```
for (let i = 12; i <13; i++) {
   for(letj=1;j<=10;j++){
     console.log(`${i} x${j}=${i*j}`);
   }
 }
```

Output:
PS C:\Users\dell\Documents\Javascript.js> node
"c:\Users\dell\Documents\Javascript.js\nestedloop.js"
12x1= 12
12x2= 24
12x3= 36
12x4= 48
12x5= 60
12x6= 72
12x7= 84
12x8= 96
12x9= 108
12x10= 120

JavaScript Functions
A JavaScript function is a block of code designed to perform a particular task. A JavaScript function is executed when "something" invokes it (calls it).
JavaScript Function Syntax
A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**.
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:**{}**
Function **parameters** are listed inside the parentheses () in the function definition.

Function **arguments** are the **values** received by the function when it is invoked. Inside the function, the arguments (the parameters) behave as local variables.

Output:

```
functionadd()
{
   let   a=10;
   let   b=20;
   letc=a+b;
   console.log("Sumoftwonumbersis:",c);
}
add()
```

PSC:\Users\dell\Documents\Javascript.js>node
"c:\Users\dell\Documents\Javascript.js\add.js"
Sumoftwonumbersis:30 Arrow function:

```
constADD=()=>{ let
   a=20;
   letb=34;
   returnconsole.log("sum=",a+b);
}
ADD()
```

Output:
PSC:\Users\dell\Documents\Javascript.js>node
"c:\Users\dell\Documents\Javascript.js\add.js"
sum=54

Anonymous Function
It is a function that does not have any name associated with it. Normally we use the *function* keyword before the function name to define a function in JavaScript, however, in anonymous functions in JavaScript, we use only
The *function* keyword with out the function name.

An anonymous function is not accessible after its initial creation, it can only be accessed by a variable it is stored in as a *function as a value*. An anonymous function can also have multiple arguments, but only one expression.

Syntax:
The below-enlightened syntax illustrates the declaration of an anonymous function using the normal declaration:

```
function(){
   //Function Body
 }
```

We may also declare an anonymous function using the arrow function technique which is shown below:

```
(()=>{
   //FunctionBody...
} )();
```

```
(()=>{
   leta=20;
   letb=34;
   returnconsole.log("sub=",a-b);
}
)()
```

Output:
PS C:\Users\dell\Documents\Javascript.js> node
"c:\Users\dell\Documents\Javascript.js\tempCodeRunnerFile.js"
sub=-14

Parameters:

```
functionadd(a,b)
{
   letc=a+b;
```

```
    returnconsole.log("Sum of two numbers is:",c);
}
add(20,20)
```

output:
PSC:\Users\dell\Documents\Javascript.js>node
"c:\Users\dell\Documents\Javascript.js\add.js"
Sum of two numbers is: 40