JavaScript Session 2

JavaScript Controlling(Looping) Statements

Loops in JavaScript are used to execute the same block of code a specified number of times or while a specified condition is true.

JavaScript Loops

Very often when you write code, you want the same block of code to run over and over again in a row.

Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript there are two different kind of loops:

- for - loops through a block of code a specified number of times
- while - loops through a block of code while a specified condition is true

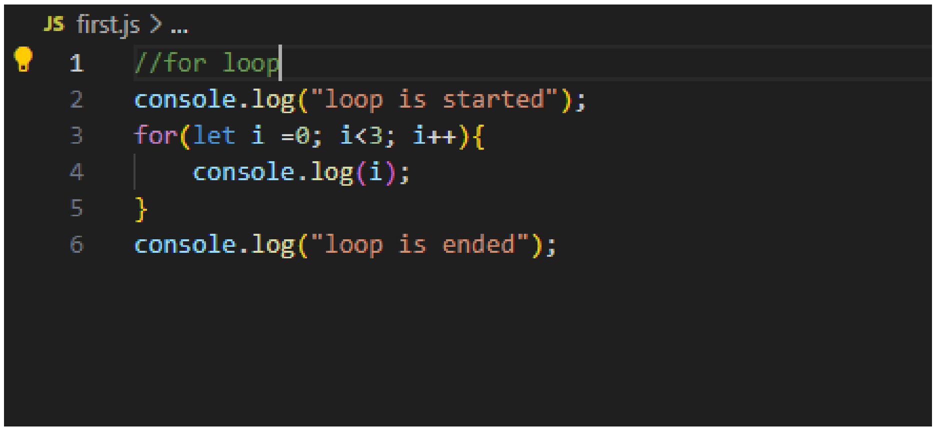
The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (var=startvalue;var<=endvalue;var=var+increment)

{

 code to be executed

}
```

Example:



```
//for loop
console.log("loop is started");
for(let i =0; i<3; i++){
    console.log(i);
}
console.log("loop is ended");
```

JavaScript While Loop

Loops in JavaScript are used to execute the same block of code a specified number of times or while

a specified condition is true.

The while loop

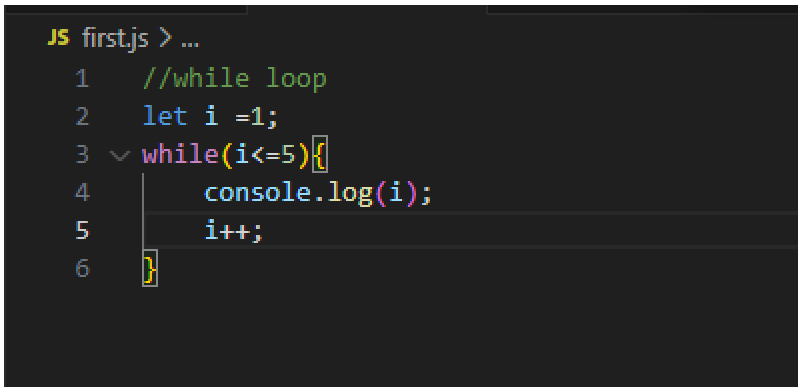
The while loop is used when you want the loop to execute and continue executing while the specified

condition is true.

```
while (var<=endvalue)

{

 code to be executed

}
```

Note: The <= could be any comparing statement.

Example



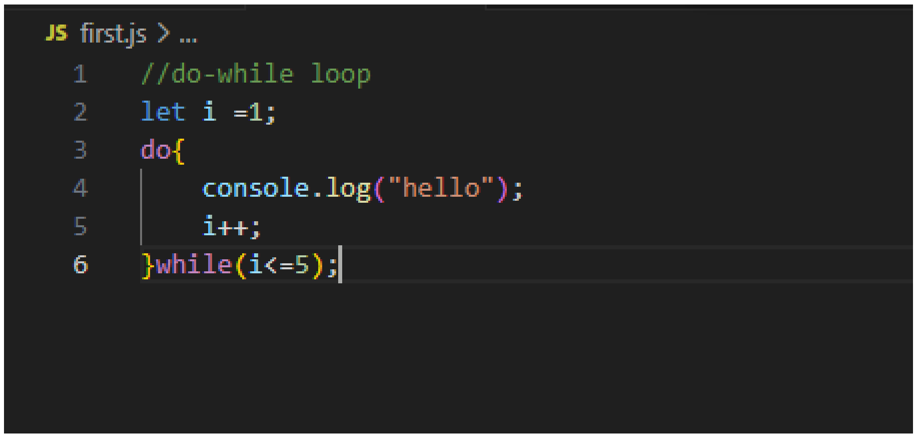
```
//while loop
let i =1;
while(i<=5){
    console.log(i);
    i++;
}
```

The do...while Loop

The do...while loop is a variant of the while loop. This loop will always execute a block of code ONCE, and then it will repeat the loop as long as the specified condition is true. This loop will always be executed at least once, even if the condition is false, because the code is executed before the condition is tested.

```
do

{

 code to be executed

}

while (var<=endvalue);
```

Example



JS first.js > ...

```
//do-while loop
let i =1;
do{
    console.log("hello");
    i++;
}while(i<=5);
```

JavaScript Functions

A function (also known as a method) is a self-contained piece of code that performs a particular "function". You can recognise a function by its format - it's a piece of descriptive text, followed by open and close brackets.A function is a reusable code-block that will be executed by an event, or when the function is called.

To keep the browser from executing a script when the page loads, you can put your script into a function.

A function contains code that will be executed 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).

How to Define a Function

No Parameters Function

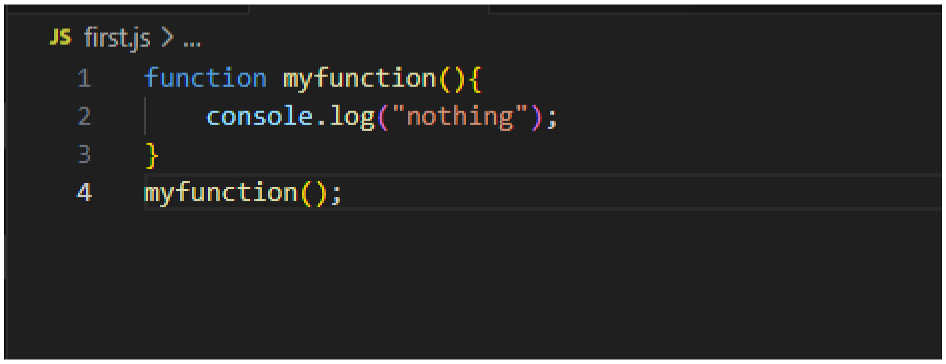
The syntax for creating a function is:

Note: A function with no parameters must include the parentheses () after the function name:

```
function functionname()
{
some code
}
```

Note: Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

Example:



```javascript
function myfunction(){
    console.log("nothing");
}
myfunction();
```

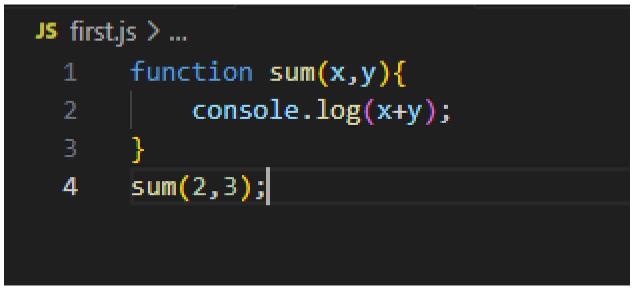
Parameters Functions

The syntax for creating a function is:

```
function functionname(var1,var2,...,varX)
{
some code
}
```

var1, var2, etc are variables or values passed into the function. The { and the } defines the start and end of the function.

Example:



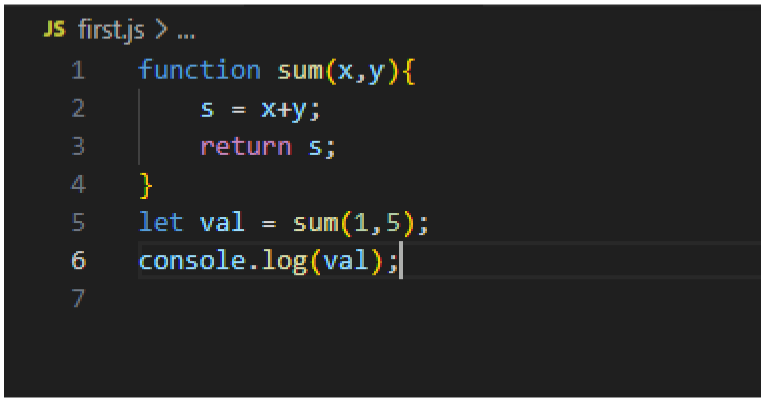
```
JS first.js > ...
function sum(x,y){
    console.log(x+y);
}
sum(2,3);
```

The return Statement

The return statement is used to specify the value that is returned from the function.

So, functions that are going to return a value must use the return statement.

Example:



```
JS first.js > ...
function sum(x,y){
    s = x+y;
    return s;
}
let val = sum(1,5);
console.log(val);
```

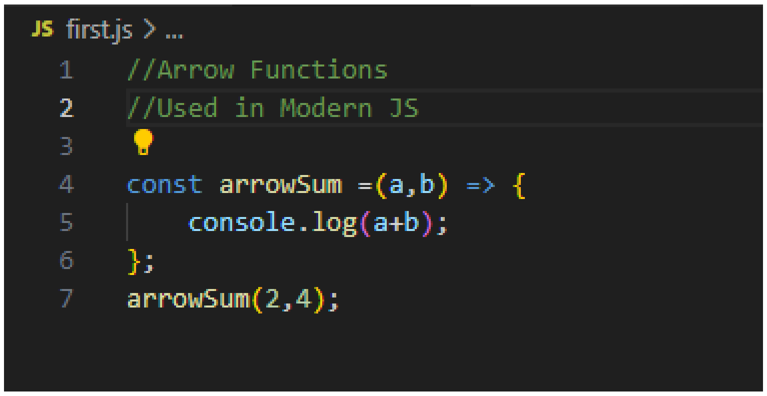
Arrow Functions

Compact way of writing a function

Syntax

```
const functionName = ( param1, param2 ...) => {

//do some work

}
```

Example:


JS first.js > ...

```
//Arrow Functions
//Used in Modern JS

const arrowSum =(a,b) => {
    console.log(a+b);
};
arrowSum(2,4);
```

Higher Order Function

forEach Loop in Arrays

arr.forEach(callBackFunction)

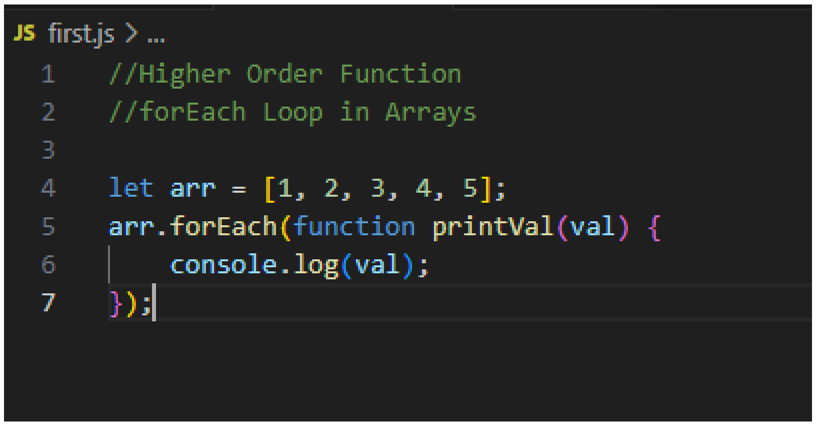
CallbackFunction : Here, it is a function to execute for each element in the array

*A callback is a function passed as an argument to another function.

Syntax

```
arr.forEach( ( val ) => {
console.log(val);
} )
```

Example:


first.js > ...

```javascript
//Higher Order Function
//forEach Loop in Arrays

let arr = [1, 2, 3, 4, 5];
arr.forEach(function printVal(val) {
    console.log(val);
});
```