BasicArrayLoop.html

Arrays & Loops

Javascript ‘Fundamentals’ {

[Basic Arrays & Loops]

< I don’t like studying, I love learning >

}

Programming Language

What is an 'Array?';

- An array is a data type that can contain one or more items called **elements**.
- Each element stores a value that you can refer to with an **index**.
- The length of the array indicates the number of elements that it contains.
- In short, an array is a structure that allows us to store multiple pieces of similar data.

Variable Declaration ‘Difference’

{

	Scope	Hoisting	Reassignment	Redeclaration
var	Function Scope	Allows	Allows	Allows
let	Block Scope	No	Allows	No
const	Block Scope	No	No	No

Two ways to create an array in JavaScript:

1. Using the new keyword with the Array object name
   ```
   var arrayName = new Array ();
   ```

2. Using an array literal
   ```
   var arrayName = [];
   ```

Array Methods

1. Array Length {

< The length property returns the length (size) of an array>

e.g.

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let size = fruits.length;
```

The output will be 4 since there are 4 elements.

}

Array Methods

2. Array toString() {

< The JavaScript method toString() converts an array to a string of (comma separated) array values.>

e.g.
```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.toString();
```

The output will be Banana,Orange,Apple,Mango.

Note: If you wanna specify the separator you can use the join() method, everything is the same but the separator. (e.g. fruits.join(" * "))

}

Array Methods

3. Array pop() {

< The pop() method removes the last element from an array:.>

e.g.

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.pop();
```

The output will be ['Banana', 'Orange', 'Apple'].

}

Array Methods

4. Array push() {

< The push() method adds a new element to an array (at the end):>

e.g.

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Kiwi");
```

The output will be ['Banana', 'Orange', 'Apple', 'Mango', 'Kiwi']

}

Array Methods

5. Array shift() {

< The shift() method removes the first array element and "shifts" all other elements to a lower index.>

e.g.

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.shift();
```

The output will be ['Orange', 'Apple', 'Mango']

}

Array Methods

6. Array unshift() {

< The unshift() method adds a new element to an array (at the beginning), and "unshifts" older elements:>

e.g.

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon");
```

The output will be ['Lemon', 'Banana', 'Orange', 'Apple', 'Mango']

}

Array Methods

7. Array delete() {

< Array elements can be deleted using the JavaScript operator delete().Using delete leaves undefined holes in the array.>

E.g.

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
delete fruits[0];
```

The output will be [<1 empty item>, 'Orange', 'Apple', 'Mango']

}

Array Methods

8. Array concat() {

< The concat() method creates a new array by merging (concatenating) existing arrays:>

E.g.

```
const myGirls = ["Cecilie", "Lone"];
const myBoys = ["Emil", "Tobias", "Linus"];

const myChildren = myGirls.concat(myBoys);
```

The output will be ['Cecilie', 'Lone', 'Emil', 'Tobias', 'Linus']

}

Array Methods

9. Array foreach() {

< The forEach() method is a built-in JavaScript function that allows you to iterate over elements in an array.The function takes two parameters: the current element value and its index. >

E.g.

```
var myFruits = ['apples', 'oranges', 'bananas'];
myFruits.forEach(function(value, index)
```

The output will be

```
apples 0
oranges 1
bananas 2
```

}

What is a ‘Loop?’;

- In programming, a "loop" is a series of commands (called "a block of code") that repeats for a specified number of iterations.
- It's a programmatic way of doing the same thing over and over again.

Syntax of a For Loop

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

Counter initialization – is executed (one time) before the execution of the code block.

Condition – defines the condition for executing the code block.

Increment expression – defines the condition for executing the code block.

Syntax of a For Loop

Counter initialization - is executed (one time) before the execution of the code block.
Condition - defines the condition for executing the code block.
Increment expression - defines the condition for executing the code block.

sets a variable before the loop starts (let i = 0).

defines the condition for the loop to run (i must be less than 5).

increases a value (i++) each time the code block in the loop has been executed.

```
for (let i = 0;          i < 5;                i++) {
  text += "The number is " + i + "<br>";
}
```

Note: You can initiate many values (separated by comma):

The While Loop

- The while loop loops through a block of code as long as a specified condition is true.

Syntax:

```
(initialize)
while (condition) {
  // code block to be executed
}
```

e.g.

The code in the loop will run, over and over again, as long as a variable (i) is less than 10:

```
var i = 0
while (i < 10) {
  text += "The number is " + i;
  i++;
}
```

The Do While Loop

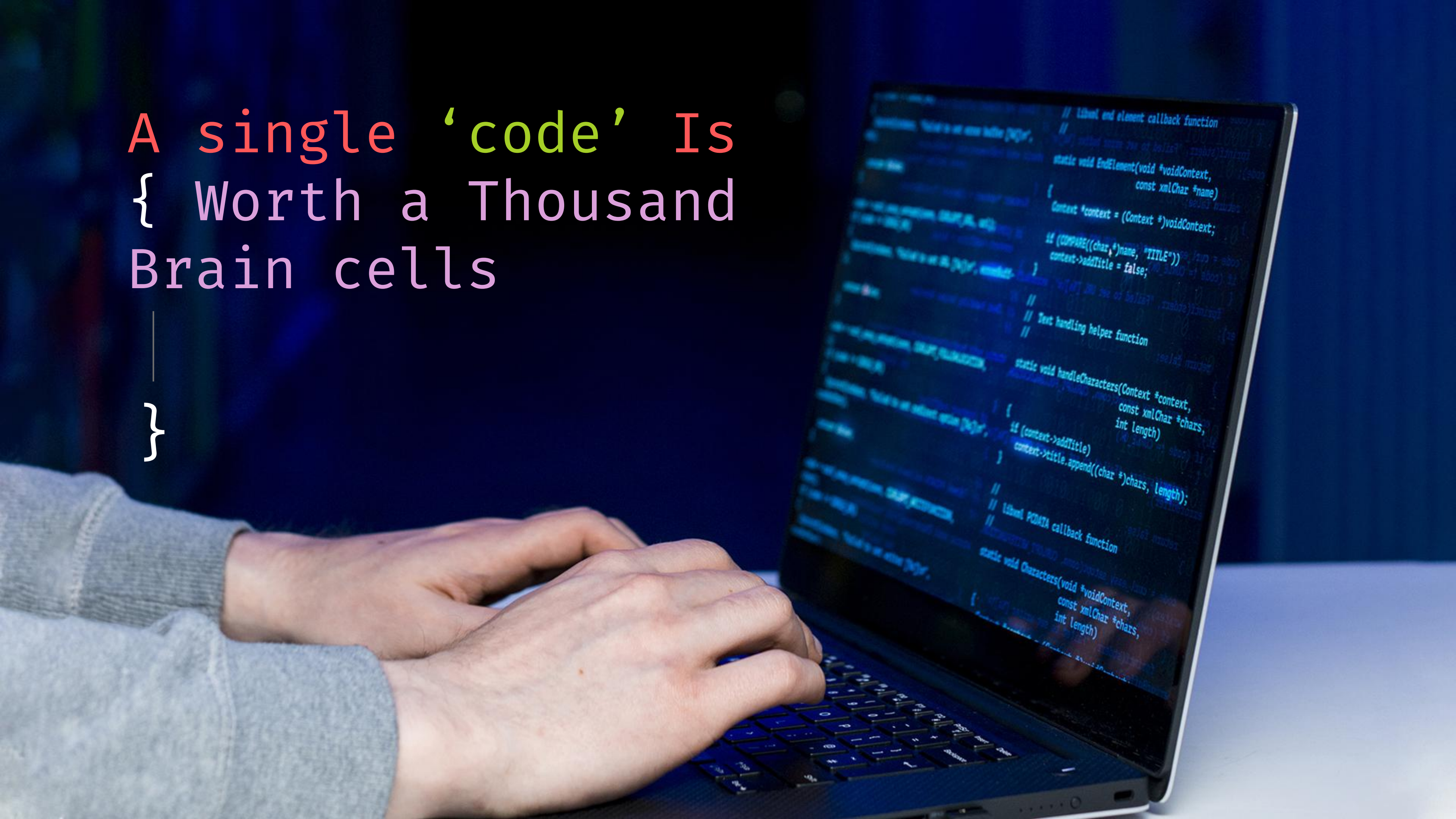
- The do while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax:

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

e.g.
The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

```
do {
  text += "The number is " + i;
  i++;
}
while (i < 10);
```



A single ‘code’ Is { Worth a Thousand Brain cells }