Scripting: JavaScript

JavaScript if, else, and else if

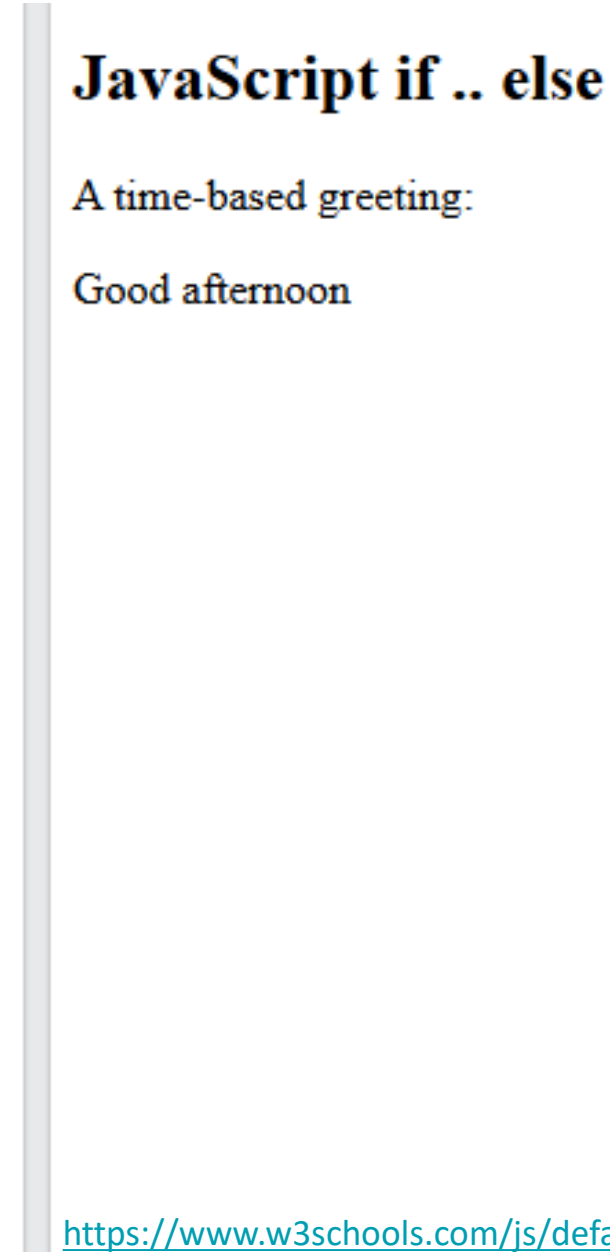
- Allow you to perform different actions for different conditions

```
if (condition) {
    // code here
}
else if (condition) {
    // more code here
}
else {
    // more code here
}
```

```
var x = 5
if (x < 10)
{
    alert("Whoop!");
}
```

```
var x = 5
if (x < 10)
{
    alert("Whoop!");
} else {
    alert("No Whoop!");
}
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript if .. else</h2>

<p>A time-based greeting:</p>

<p id="demo"></p>

<script>
const time = new Date().getHours();
let greeting;
if (time < 12) {
  greeting = "Good morning";
} else if (time < 17) {
  greeting = "Good afternoon";
} else {
  greeting = "Good evening";
}
document.getElementById("demo").innerHTML = greeting;
</script>

</body>
</html>
```



https://www.w3schools.com/js/default.asp

Loops

- Loops allow you to repeat things lots of times
- They use a **condition** to see if you could go round again
- They have a bit of code that is **run every loop**, often used as a counter
- They have a bit of code that is **run once** at the start, often used to set up the counter
- Loops can be created using different constructs
  - While
  - Do … While
  - For

While Loop

```
while (condition)
{
//code executed
}
```

- If the condition is true, the code is executed
- Then the condition is tested again, if it is still true the code block is executed
- This execution will continue until the condition becomes false

```
A = 0;
while (A < 3)
{
console.log("Hello");
A = A + 1;
}
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript While Loop</h2>

<p id="demo"></p>

<script>
let text = "";
let i = 0;
while (i < 10) {
  text += "<br>The number is " + i;
  i++;
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

JavaScript While Loop

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9

x += y
Same as: x = x + y

https://www.w3schools.com/js/default.asp

Do While Loop

```
do
{
//code executed
}
 while(condition)
```

- Similar to While loop
- But code is executed at least once before condition is checked

```
A=0;
do
{
A = A + 1;
console.log("Hello");
}while (A < 3);
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Do While Loop</h2>

<p id="demo"></p>

<script>
let text = ""
let i = 0;

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

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

JavaScript Do While Loop

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9

https://www.w3schools.com/js/default.asp

For Loop

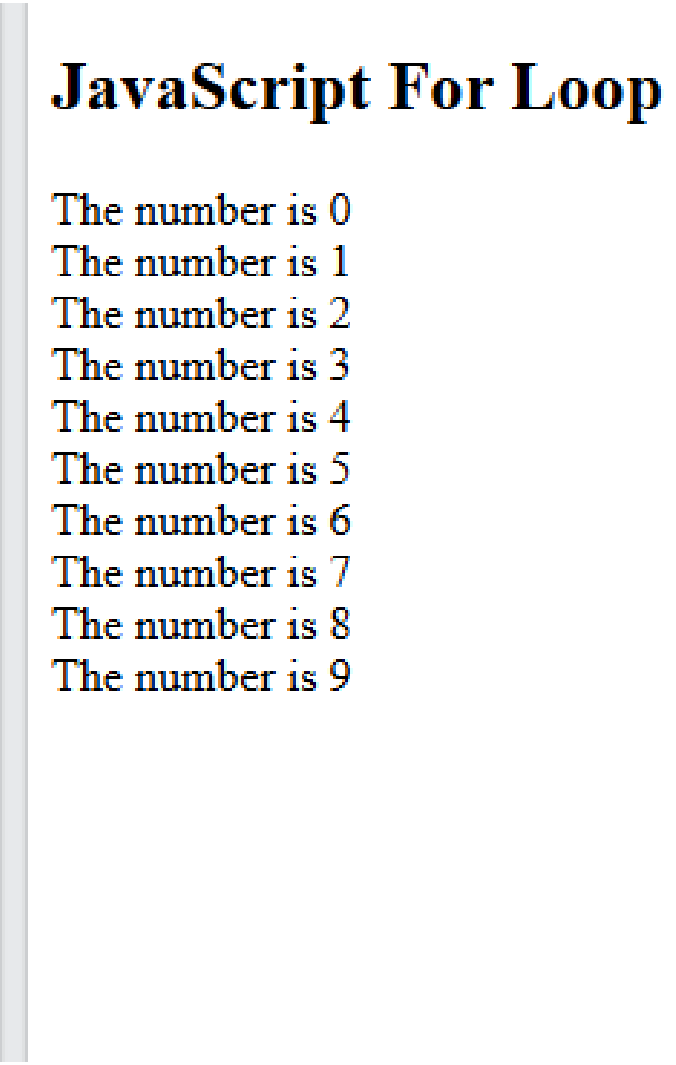
- Similar to other loops but with some configurations

```
for(statement1; statement2; statement3)
{
//code
 }
```

- **statement 1** : executed before loop (e.g. initialization of variables)
- **statement 2**: condition of the loop
- **statement 3**: executed each time after the loop (e.g. increment or decrement)

```
for (var i = 0; i < 10; i++) {
   console.log("Let's count!" + i);
}
```

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript For Loop</h2>

<p id="demo"></p>

<script>
let text = "";

for (let i = 0; i < 10; i++) {
  text += "The number is " + i + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```



https://www.w3schools.com/js/default.asp

Arrays

- A variable created by **`var car`** can store a single value
- But what if you want to store multiple values?
- The **Array** object is used to store a set of values in a single variable name

```
var myArray = [];
var myBreakfast = [“Mushrooms”, “Eggs”, “Bacon”];
const cars = ["Saab", "Volvo", "BMW"];
```

- Creating Array

```
var arrayName = [];
var arrayName = new Array();
```

- You can also create an array, and then provide the elements:

```
const cars = [];
cars[0]= "Saab";
cars[1]= "Volvo";
cars[2]= "BMW";
```

Arrays: Accessing Elements

- Access array element via index
- Index starts from 0

```
var myBreakfast = [“Mushrooms”, “Eggs”, “Bacon”];
myBreakfast[0]; // returns “Mushrooms”
```

- What will be returned by `myBreakfast[3]`?
- This position does not exist in this array, so we will receive an error!

Arrays

There are some built-in methods to perform some operations in an Array:

- length: number of elements in the array
- push() : add new element at the end of an array
- unshift() : add new element at the beginning of an array
- pop() : remove last element
- shift() : remove first element
- concat() : combine two arrays into a new array

```
var breakfast = [“Mushrooms”, “Eggs”, “Bacon”];
breakfast.push(“Beans”); // [“Mushrooms”, “Eggs”, “Bacon”, “Beans”];
var dessert = [“Icecream” , “Cake”];
var food = breakfast.concat(dessert);
```

Array vs Object

- Both Arrays and Objects can be used to store a collection of data

```
var student = new Array ("Sally", "Smith", 20);
student[0]  //returns Sally
```

- Not easy to know which index represents what. We have to memorize!
- Objects overcome this issue by representing "things" with characteristics (also known as properties)
  - A property consists of a key and a value

```
// Basic object syntax

var object = {
  key: 'value'
};
```

```
var student = {
  firstName: 'Sally',
  secondName: 'Smith',
  age=20
};
```

Objects

- Properties in objects can be accessed, added, changed, and removed by using either **dot** or **bracket** notation
- Dot Notation
  - `student.age` //returns 20
- Bracket Notation
  - `student['age']` //returns 20
- You can change the value of a property like this:
  - `student.age=30`
- You can add new properties as:
  - `student.hobbies=['hiking', 'travel', 'reading']`

```
var student = {
  firstName: 'Sally',
  secondName: 'Smith',
  age=20
};
```

Combining loops, arrays, conditions

Given an array of studentGrades, write a JavaScript code to find how many students failed (Grade<40)

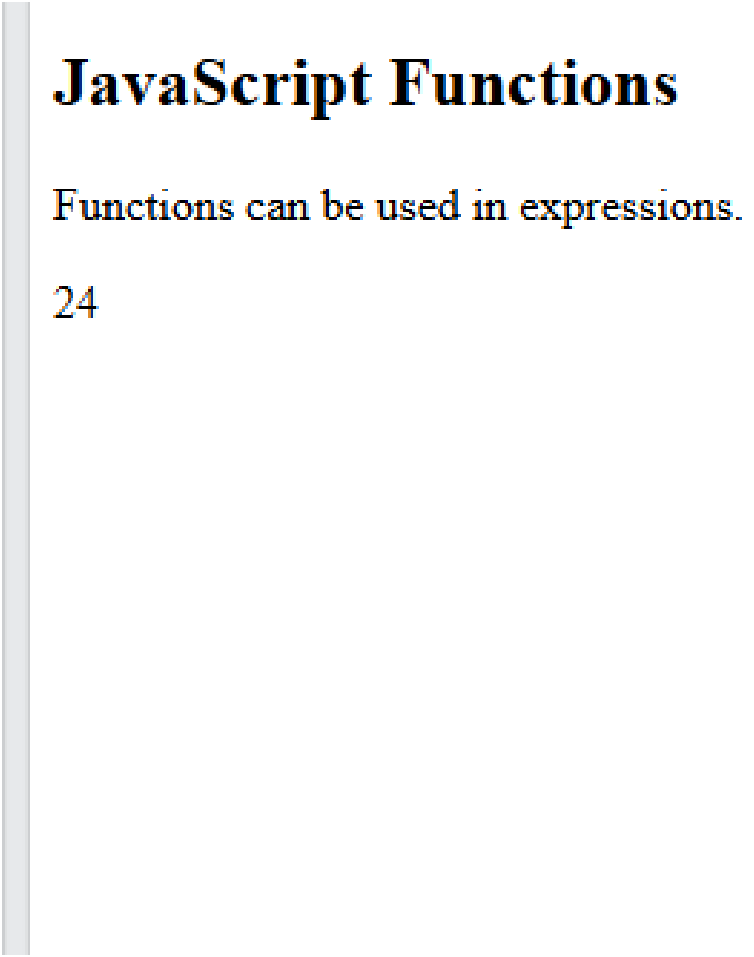
```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>

let text = "";
var studentGrades = [35, 23, 78, 64, 58, 44, 41, 15, 0, 23];
var fails = 0;
for ( var i = 0; i < studentGrades.length; i++) {
    if ( studentGrades[i] < 40 ) {
      fails = fails + 1;
   }

}

text = text + fails + " students failed. ";

document.getElementById("demo").innerHTML = text;

</script>

</body>
</html>
```

5 students failed.

JavaScript Function

- A group of statements to perform a specific task
- JavaScript lets you define functions using the **`function`** keyword
- Functions allow us to repeat and reuse small bits of code
- You can use functions to reuse code for things you do often
- A function will be executed by an event or by a call to that function
- Functions can return values using the **`return`** keyword

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Functions</h2>
<p>Functions can be used in expressions.</p>
<p id="demo"></p>

<script>
function myFunction(a, b) {
  return a * b;
}
let x = myFunction(4, 3) * 2;
document.getElementById("demo").innerHTML = x;
</script>

</body>
</html>
```



https://www.w3schools.com/js/default.asp