JS

## How to make a Burrito?

```
Begin
     Get out the rice cooker
   Fill it with rice
   Fill it with water
   Cook the rice
   Chop the vegetables
   Stir-fry the vegetables
   Taste-test the vegetables
      If the veggies are good
         Remove them from the stove
      If the veggies aren't good
         Add more pepper and spices
      If the veggies aren't cooked enough
         Keep stir-frying the veggies
   Heat the tortilla
   Add rice to tortilla
   Add vegetables to tortilla
   Roll tortilla

End
```

## HTML and CSS recap

- HTML is the markup language that we use to structure and give meaning to our web content, for example defining paragraphs, headings, and data tables, or embedding images and videos in the page.
- CSS is a language of style rules that we use to apply styling to our HTML content, for example setting background colors and fonts, and laying out our content in multiple columns.

## What is Javascript?

JavaScript is a programming language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else.

## Some dynamic things can be done with JS

- Read elements from documents and write new elements and text into documents Manipulate or move text
- Perform mathematical calculations on data
- React to events, such as a user clicking a button
- Retrieve the current date and time from a user ’s computer or the last time a document was modified
- Determine the user’s screen size, browser version, or screen resolution
- Perform actions based on conditions such as alerting users if they enter the wrong information into a form

## Javascript Syntax

Javascript can be implemented by adding Javascript statements between <script></script> tag.

It can be placed anywhere but it is recommended to put it on the head.

## Hello Javascript!

```
html>
   <body>
      <script language="javascript"
type="text/javascript">
         <!--
            document.write("Hello
Javascript!")
         //-->
      </script>
   </body>
</html>
```

## Javascript syntax rules

Semicolons are optional

```
<script language="javascript" type="text/javascript">
   <!--
      var1 = 10
      var2 = 20
   //-->
</script>
```

```
<script language="javascript" type="text/javascript">
   <!--
      var1 = 10; var2 = 20;
   //-->
</script>
```

## Comments in Javascript

- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

## Including Javascript in project.

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.
- Script in an external file and then include in <head>...</head> section.

## Try It : Include JS in header

```
<html>

   <head>

      <script type="text/javascript">
         <!--
            function sayHello() {
               alert("Hello World")
            }
         //-->
      </script>

   </head>

   <body>
      <input type="button" onclick="sayHello()" value="Say Hello" />
   </body>

</html>
```

## Try It: Javascript in body tag

```
<html>

   <head>
   </head>

   <body>

      <script type="text/javascript">
         <!--
            document.write("Hello World")
         //-->
      </script>

      <p>This is web page body </p>

   </body>
</html>
```

## Try It: Javascript link from other file.

```html
<html>

   <head>
      <script type="text/javascript" src="filename.js" ></script>
   </head>

   <body>
      .......
   </body>
</html>
```

```javascript
function sayHello() {
   alert("Hello World")
}
```

## What is a variable?

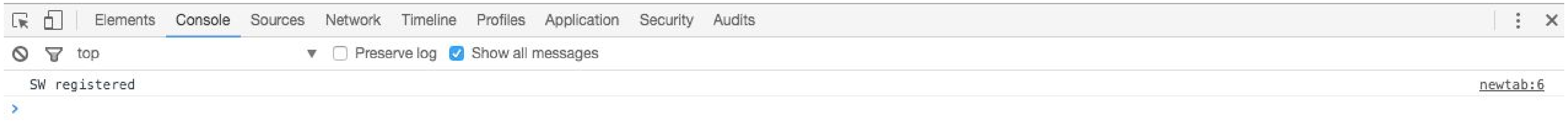
A computer program needs to store data using variables. A variable is an information storage area.

```
var quantity;

quantity = 3;
```

## What just happened?

- **var** is a keyword in Javascript.
- We create a new variable and call it **quantity**.
- **“=”** means an assignment character.
- We then assign the variable with a quantity, eg: 3

## Try it (Use Chrome Developer Tool)



```
SW registered                                                    newtab:6
>
```

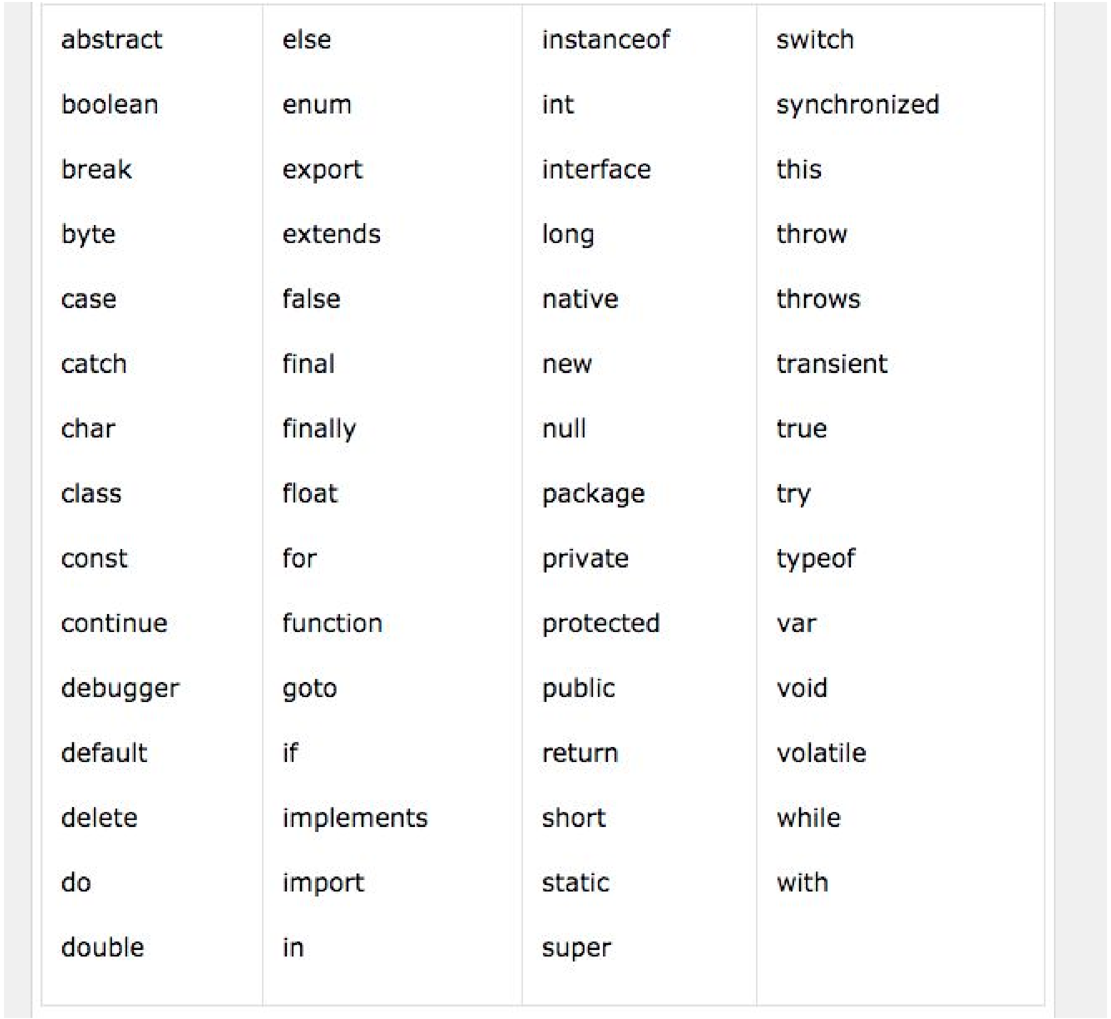
## Try It (Declaring a variable)

```
var a;

console.log(a);
```

```
var a;
a = 3.14;
console.log(a);
```

## Rules for naming variable

- Name must begin with a letter, dollar sign ($) or an underscore.
- It cannot start with a number.
- It can contain letters, numbers, dollar sign or an underscore. It cannot contain dash (-) or a period (.)
- You cannot use the reserved keywords, such as **var**.
- Variables are case sensitive. Score is different that score



| | | | |
|---|---|---|---|
| abstract | else | instanceof | switch |
| boolean | enum | int | synchronized |
| break | export | interface | this |
| byte | extends | long | throw |
| case | false | native | throws |
| catch | final | new | transient |
| char | finally | null | true |
| class | float | package | try |
| const | for | private | typeof |
| continue | function | protected | var |
| debugger | goto | public | void |
| default | if | return | volatile |
| delete | implements | short | while |
| do | import | static | with |
| double | in | super | |

## String

- **String** are sequences of characters, like the letters a - z, spaces, and even numbers.
- The followings represent strings:
  a. “Hello World”
  b. ‘ADA stands for “Asia Dev Academy”’
  c. “4”
- Quote should be straight curly code, be careful when you do copy paste from slide, website or Doc.

## Try It : String

```
var title;
var message;

title = "Welcome to ADA”;
message = 'Welcome to week 2 of ADA! I hope everyone is
having fun';

console.log(title);
console.log(message);
```

## Numbers

- **Integers** are whole numbers, e.g. 10, 400, or -5.
- **Floating point numbers** (floats) have decimal points and decimal places, for example 12.5, and 56.7786543.
- **Doubles** are a specific type of floating point number that have greater precision than standard floating point numbers (meaning that they are accurate to a greater number of decimal places).

## Try It: Numbers

```
var price;
var quantity;
var total;



price = 5;
quantity = 14;

total = price * quantity;
console.log('RM' + total);
```

## Arithmetic operators

| **Operator** | **Purpose** | **Example** |
|---|---|---|
| +(Addition) | Adds two numbers together. | 6 + 9 |
| - (Substraction) | Substracts the right number from the left | 20-5 |
| * (Multiplication) | Multiplies two numbers together. | 3 * 7 |
| / (Division) | Divides the left number by the right | 10/4 |
| % (Remainder/Modulo) | Returns the remainder left over after division. | 8 % 3 (returns 2) |
| ++ | Increment number by 1 | i= 1;<br>i++; |
| -- | Decrement number by 1 | i--; |

## Boolean

- Boolean is a variable that can have a value of **true** and **false**.

```
var hungry;
hungry = true;

If (hungry){
console.log(“Can we have a quick lunch break?”);
}
Else
{
console.log(“I can still handle it for a while”);
}
```

## Arrays

- Arrays are used when we are storing a list of values that might be related to each other.
- You can mix different data types into a single array.

## Try it: Array

```
// Create an array and assign it values.
var colors;
colors = ['white', 'black', 'custom'];

// Show the first item from the array.
console.log(colors[0]);
```

## Array: Things to remember.

- Each item is given an index, index will start at 0.
- You access the items in array using square brackets, eg: colors[2]
- You may count the number of items in an array using method length. Eg: colors.length

## Try it: Array- Things to remember

- Create an array with 4 places that you have visited.
- Print the second item inside the array.
- Print the number of items inside the array using the appropriate method.

## Comparison operator

| Operator | Name | Purpose | Example |
|---|---|---|---|
| `===` | Strict equality | Tests whether the left and right values are identical to one another | `5 === 2 + 4` |
| `!==` | Strict-non-equality | Tests whether the left and right values **not** identical to one another | `5 !== 2 + 3` |
| `<` | Less than | Tests whether the left value is smaller than the right one. | `10 < 6` |
| `>` | Greater than | Tests whether the left value is greater than the right one. | `10 > 20` |
| `<=` | Less than or equal to | Tests whether the left value is smaller than or equal to the right one. | `3 <= 2` |
| `>=` | Greater than or equal to | Tests whether the left value is greater than or equal to the right one. | `5 >= 4` |

# Assignment operator

| Operator | Purpose | Example | Shortcut for |
|---|---|---|---|
| `+=` (Addition assignment) | Adds the value on the right to the variable value on the left, then returns the new variable value | `x = 3;`<br>`x += 4;` | `x = 3;`<br>`x = x + 4;` |
| `-=` (Subtraction assignment) | Subtracts the value on the right from the variable value on the left, and returns the new variable value | `x = 6;`<br>`x -= 3;` | `x = 6;`<br>`x = x - 3;` |
| `*=` (Multiplication assignment) | Multiples the variable value on the left by the value on the right, and returns the new variable value | `x = 2;`<br>`x *= 3;` | `x = 2;`<br>`x = x * 3;` |
| `/=` (Division assignment) | Divides the variable value on the left by the value on the right, and returns the new variable value | `x = 10;`<br>`x /= 5;` | `x = 10;`<br>`x = x / 5;` |

## Try It: Arithmetic operators

```
var subtotal = (13 + 1) * 5; // Subtotal is 70

var shipping = 0.5 * (13 + 1); // Shipping is 7

var total = subtotal + shipping; // Total is 77

console.log(subtotal);
console.log(shipping);
console.log(total)
```

## String Operators

- There is only one string operators which is ‘+’
- It is used to join the strings on either side on it.
- This is called string **concatenation**.

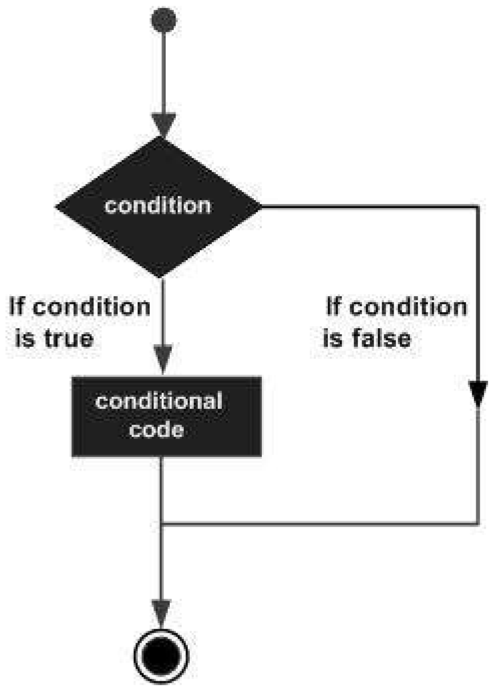
## Try It: String Operators

```
var greeting = ‘Good Morning ';

var name = ‘Michele’;

var welcomeMessage = greeting + name + '!';

console.log(welcomeMessage);
```

## If statements

- If statement evaluates a condition.
- If the condition evaluates to true, any statement in the subsequent code block are evaluated.
- You may also add an else statement. If the condition resolves to false, the second block of statement will run instead.

## If statements (control flow)



## Try it: If statements

```
var score = 75;     // Score
var msg;            // Message

if (score >= 50) {  // If score is 50 or higher
  msg = 'Congratulations!';
  msg += ' Proceed to the next round.';
}

console.log(msg)
```

## Try it: If else statements

```
var pass = 50;      // Pass mark
var score = 75;    // Current score
var msg;              // Message

// Select message to write based on score
if (score > pass) {
  msg = 'Congratulations, you passed!';
} else {
  msg = 'Have another go!';
}
console.log(msg);
```

## Exercise

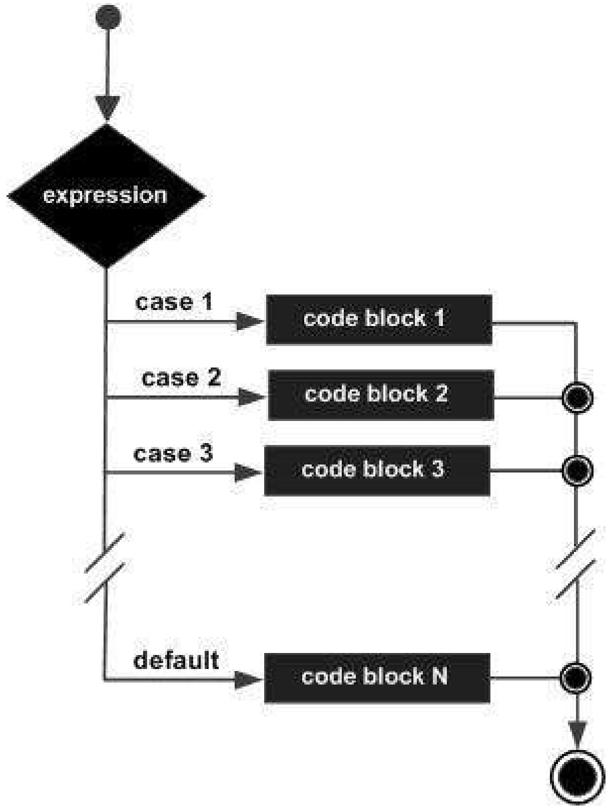
Create a validator for 1 ticket service with the following rules:

1) For less than 18 years old, the ticket price will be 50 % of normal ticket price
2) For 19-40 years old, the ticket price will be normal price.
3) For 40-65 years old, there will be a discount of 25%
4) For Senior citizen, 65 years old and above, it will be same discount as kids. (50 %)

## Switch

- If you have multiple if .. else if.. Statement, you may use switch to cater the block for each scenario.
- A **switch** statement starts with a variable called the switch value .
- You will define each **case** to indicate the possible outcome it the code run.

# Switch: Control flow



## Try It: Switch

```
var msg;        // Message
var level = 2;  // Level

switch (level) {
case 1:
    msg = 'Good luck on the first test';
    break;
case 2:
    msg = 'Second of three - keep going!';
    break;
case 3:
    msg = 'Final round, almost there!';
    break;

default:
    msg = 'Good luck!';
    break;
}
```

## Loop

While writing a program, you may encounter a situation where you need to perform an action over and over again.

In such situations, you would need to write loop statements to reduce the number of lines.

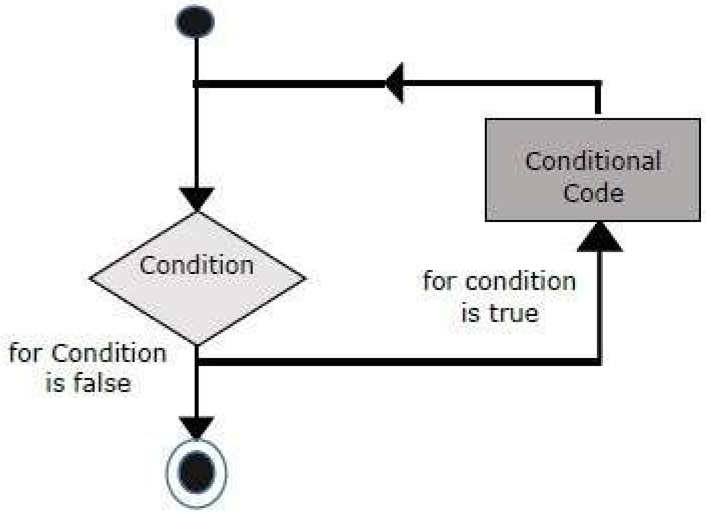
There are few ways to write loop, using **for, for in, while** and **do while.**

## For loop

The '**for**' loop is the most compact form of looping. It includes the following three important parts

- The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- The **iteration statement** where you can increase or decrease your counter.

## For loop : Control flow.



## Try It: For loop

```
var scores = [24, 32, 17];      // Array of scores
var arrayLength = scores.length;// Items in array
var roundNumber = 0;             // Current round
var msg = '';                    // Message

for (var i = 0; i < arrayLength; i++) {

  roundNumber = (i + 1);

  msg += 'Round ' + roundNumber + ': ';

  msg += scores[i] + '<br />';
}
console.log(msg);
```

## Exercise : Multiplication table.

Create a multiplication table where you will enter the number, multiplier and results.

Eg:

1 x 3 = 3

2 x 3 = 6

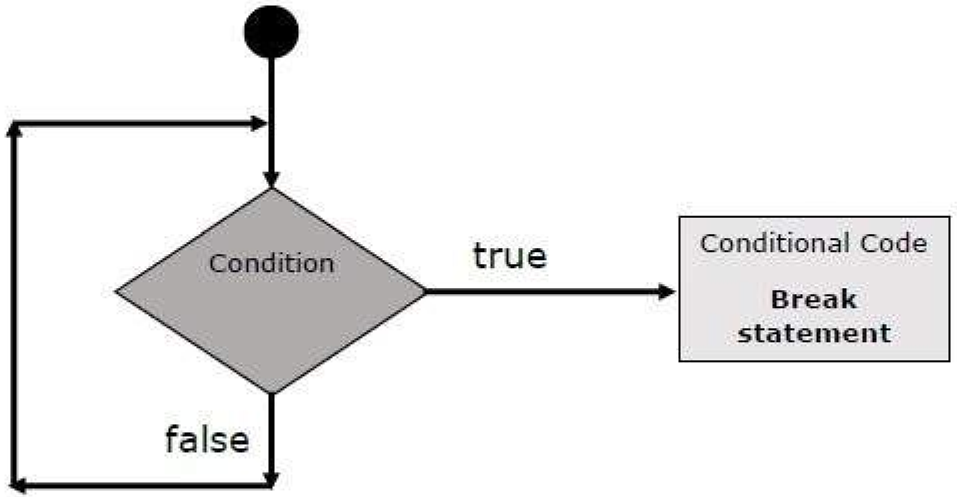
...

12 x 3 = 36

## While loop

The purpose of a **while** loop is to execute a statement or code block repeatedly as long as an **expression** is true. Once the expression becomes **false,** the loop terminates.

## While loop : Control flow.



```
var i = 1;       // Set counter to 1
var msg = '';    // Message

// Store 5 times table in a variable
while (i < 10) {
  msg += i + ' x 5 = ' + (i * 5) + '<br />';
  i++;
}
console.log(msg);
```

## Do While loop

The key difference between a **while** loop and a **do while** loop is that the statement in the code come before the condition.

The statement will at least be executed once even if the condition has never be met.

```
var i = 1;       // Set counter to 1
var msg = '';    // Message



do {
  msg += i + ' x 5 = ' + (i * 5) + '<br />';
  i++;
} while (i < 1);

console.log(msg);
```

## Function

A function is a group of statements that grouped together to perform a specific task.

We regroup the statements so that it can be reused later rather than repeating the same set of statements.

You may need to provide some information to perform a specific task. This is called **parameters**.

If a function require you to provide an answer, the response is known as **return value.**

## Try It: Declaring a Function and Calling It

```
function sayHello() {

document.write(‘Hello’);

}
```

```
sayHello();
```

## Try It: Declaring a function with parameters

```
function getArea(width, height) {

return width * height;

}
```

```
getArea(3, 5);
```

## What is the difference?

```
function getArea(width, height) {

return width * height;

}
```

```
function sayHello() {

document.write(‘Hello’);

}
```

## Getting multiple values out of a function

You may also return multiple value out of a function, by returning it inside an array.

```
function getSize(width, height, depth) {

var area = width * height;

var volume = width * height * depth;

var sizes = [area, volume];

return sizes;

}
```

## Surprise quiz: minMaxAvg

Create a function minMaxAvg that will take an array as parameter and will return the smallest number within the array, max number within the array and the average.

# More String

| Property/Method | Description |
|---|---|
| length | Returns number of characters in the string/ |
| toUpperCase() | Changes string to uppercase characters. |
| toLowerCase() | Changes string to lowercase characters. |
| charAt() | Returns the character at the specified index. |
| indexOf() | Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found. |
| lastIndexOf() | Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found. |

## More String (2)

| Property/Method | Description |
|---|---|
| substring() | Returns character found between two index number where character for the first number is included and the character for the last index is not included. |
| split() | Splits a String object into an array of strings by separating the string into substrings. |
| trim() | Removes whitespace from start and end of string |
| replace() | Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring. |

## Try It : String

Create a string : “Asia Developer Academy”. Do the following operation

1) Get the string length
2) Change the word to uppercase
3) Change the word to lowercase
4) Get the first ‘e’ out
5) Get the index of ‘ev’
6) Get the last index of ‘e’
7) Get ‘Developer’ out using substring
8) Separate the phrase word by word

## Exercise:

Create a function that will take a String as parameters and will capitalize the first character of each letter.

Eg: capitalizeString(“my name is muzaffar”); => My Name is Muzaffar

## More Numbers

| Property/Method | Description |
|---|---|
| isNan() | Checks if the value is not a number |
| toFixed() | Rounds to specified number of decimal places (returns a string) |
| toPrecision() | Rounds to total number of places (returns a string) |
| toExponential() | Returns a string to total number of places |

## Try It : More Numbers

Create a number, eg: 10.23456789;

1) Get the number to 3 decimal places.
2) Return the number to 3 number of places.

## Math Object

| Property/Method | Description |
|---|---|
| Math.PI | Return value of PI |
| Math.round() | Returns the number to nearest integer |
| math.sqrt(n) | Returns the square root of a number. |
| Math.ceil() | Returns the smallest integer greater than or equal to a number. |
| Math.floor() | Returns the largest integer less than or equal to a number. |
| Math.random() | Returns a pseudo-random number between 0 and 1. |

## Try It : Math

Return a random number between 1-100

## Date and Time

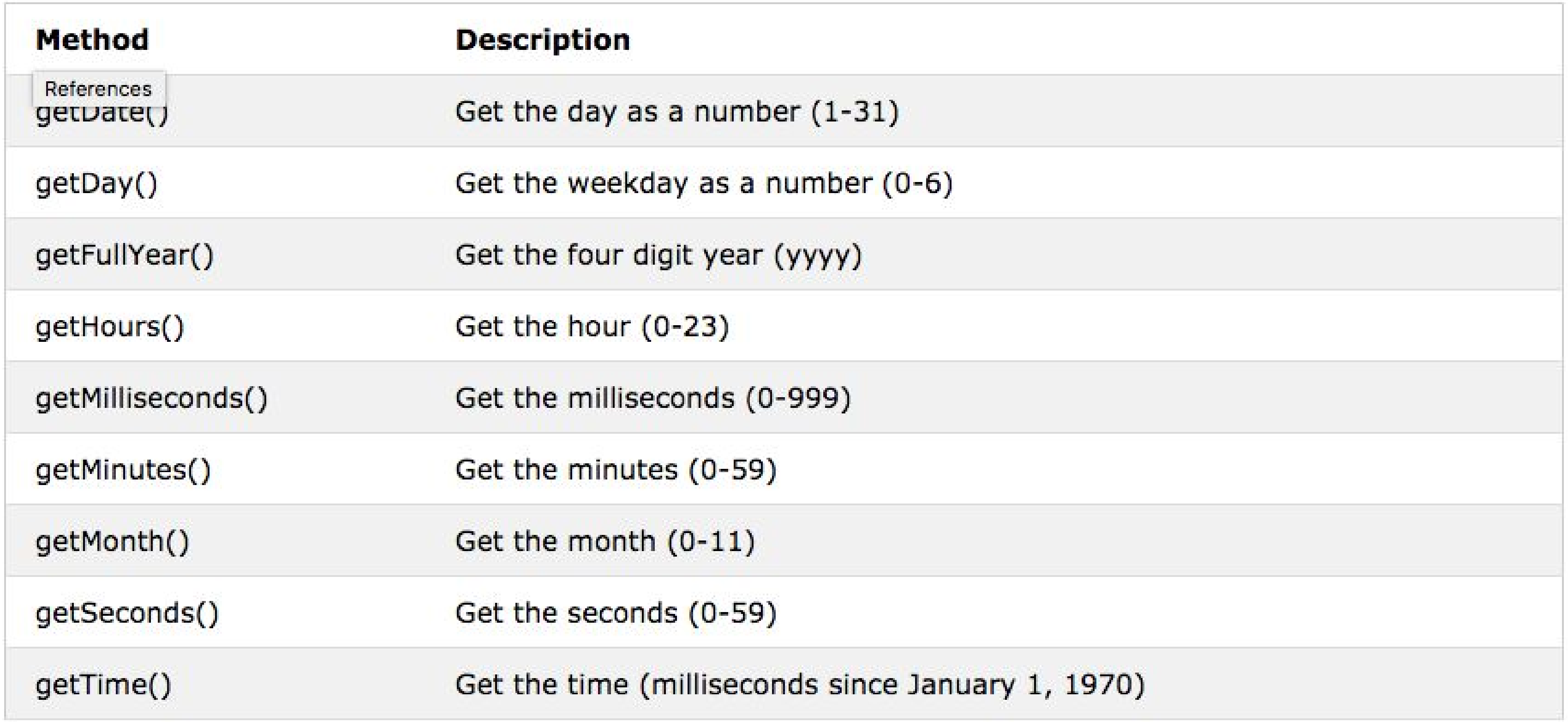
To work with dates, you create an instance of the **Date** object constructor.

```
var today = new Date();
```

You may also set the date and time according to format wishes:

```
var dob = new Date(1996,11,26,15,45,55);

var dob = new Date(‘Dec 26, 1996 15:45:55’);

var dob = new Date(1996,11,26);
```

# Date and Time Object


References

| Method | Description |
|---|---|
| getDate() | Get the day as a number (1-31) |
| getDay() | Get the weekday as a number (0-6) |
| getFullYear() | Get the four digit year (yyyy) |
| getHours() | Get the hour (0-23) |
| getMilliseconds() | Get the milliseconds (0-999) |
| getMinutes() | Get the minutes (0-59) |
| getMonth() | Get the month (0-11) |
| getSeconds() | Get the seconds (0-59) |
| getTime() | Get the time (milliseconds since January 1, 1970) |

## Try It: Date and Time

```
var today = new Date();

var year = today.getFullYear();

var firstClass = new Date('Aug 13, 2017 09:00:00')

var difference = today.getTime()-firstClass.getTime();

 var differenceInDays = difference/86400000

var msg = Math.floor(differenceInDays) + ‘ since we start coding!’;

document.write(msg);
```

## Exercise

Create a function that will greet you based on the current time on the system.

If it is between 7 am to 1 pm, you will greet “Good Morning”

From 1pm to 5 pm, you will greet “Good Afternoon”

From 5 pm to 7 pm , you will greet “Good Evening”

Other than that, “Good Night”