JavaScript

Functions and Objects

Why Functions?

- With functions you can reuse code
- You can write code that can be used many times.
- You can use the same code with different arguments, to produce different results.

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).
- A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**.

JavaScript Functions –contd.,

- 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: {}

JavaScript Functions –contd.,

- 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.

Function Invocation

- The code inside the function will execute when "something" **invokes** (calls) the function:
- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

Function Return

- When JavaScript reaches a return statement, the function will stop executing.
- If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.
- Functions often compute a **return value**. The return value is "returned" back to the "caller"

Example 1

```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Functions</h1>

<p>Call a function which performs a calculation and returns the
result:</p>
<p id="demo"></p>
<script>
function myFunction(p1, p2) {
  return p1 * p2;
}
let result = myFunction(4, 3);
document.getElementById("demo").innerHTML = result;

</script></body></html>
```

Example 2

```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Functions</h1>

<p>Invoke (call) a function to convert from Fahrenheit to
Celsius:</p>
<p id="demo"></p>

<script>
function toCelsius(f) {
  return (5/9) * (f-32);
}

let value = toCelsius();
document.getElementById("demo").innerHTML = value;
</script>
</body></html>
```

JavaScript Objects

- JavaScript variables are containers for data values.
- Objects are variables too. But objects can contain many values.
- The values are written as **name:value** pairs (name and value separated by a colon).
- It is a common practice to declare objects with the const keyword.
- **Const Keyword**
  - Variables defined with const cannot be **Redeclared**
  - Variables defined with const cannot be **Reassigned**
  - Variables defined with const have **Block Scope**

JavaScript Objects

```
const car = {type:"Fiat", model:"500", color:"white"};
```

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

<h2>JavaScript Objects</h2>

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

<script>
// Create an object:
const car = {type:"Fiat", model:"500", color:"white"};

// Display some data from the object:
document.getElementById("demo").innerHTML = "The car type is " +
car.type;
</script></body></html>
```

JavaScript Objects

```
const person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```

Object Properties

The **name:values** pairs in JavaScript objects are called **properties**

Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

Accessing Object Properties

You can access object properties in two ways

objectName.propertyName

Eg.
person.lastName;

objectName["propertyName"]

Eg.
person["lastName"];

Object Methods

- Objects can also have **methods**.
- Methods are **actions** that can be performed on objects.
- Methods are stored in properties as **function definitions**.
- A method is a function stored as a property.

```
const person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  } };
```

Accessing Object Methods

objectName.methodName()

Eg.
name = person.fullName();

If you access a method **without** the () parentheses, it will return the **function definition**

name = person.fullName;

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

<h2>JavaScript Objects</h2>
<p>An object method is a function definition, stored as a property
value.</p>

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

<script>
// Create an object:
const person = {
  firstName: "John“,  lastName: "Doe“,  id: 5566,
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
};

// Display data from the object:
document.getElementById("demo").innerHTML = person.fullName();
</script></body></html>
```

Built-in Objects

- Built-in objects are not related to any Window or DOM object model.
- These objects are used for simple data processing in the JavaScript.

Math Object

- Math object is a built-in static object.
- It is used for performing complex math operations.
- **Math Properties**

Math Property	Description
SQRT2	Returns square root of 2.
PI	Returns Π value.
E \	Returns Euler's Constant.
LN2	Returns natural logarithm of 2.
LN10	Returns natural logarithm of 10.
LOG2E	Returns base 2 logarithm of E.
LOG10E	Returns 10 logarithm of E.

Math Methods

Methods	Description
abs()	Returns the absolute value of a number.
acos()	Returns the arccosine (in radians) of a number.
ceil()	Returns the smallest integer greater than or equal to a number.
cos()	Returns cosine of a number.
floor()	Returns the largest integer less than or equal to a number.
log()	Returns the natural logarithm (base E) of a number.
max()	Returns the largest of zero or more numbers.
min()	Returns the smallest of zero or more numbers.
pow()	Returns base to the exponent power, that is base exponent.

```
<html>
    <head>
        <title>JavaScript Math Object Methods</title>
    </head>
    <body>
        <script type="text/javascript">

            var value = Math.SQRT2;
            document.write("ABS Test Value : " + value +"<br>");



        </script>
    </body>
</html>
```

Date Object

- Date is a data type.
- Date object manipulates date and time.
- Date() constructor takes no arguments.
- Date object allows you to get and set the year, month, day, hour, minute, second and millisecond fields.
- **Syntax:**
  var variable_name = new Date();

  Example:
  var current_date = new Date();

Date Methods

Methods	Description
Date()	Returns current date and time.
getDate()	Returns the day of the month.
getDay()	Returns the day of the week.
getFullYear()	Returns the year.
getHours()	Returns the hour.
getMinutes()	Returns the minutes.
getSeconds()	Returns the seconds.
getMilliseconds()	Returns the milliseconds.
getTime()	Returns the number of milliseconds since January 1, 1970 at 12:00 AM.
getTimezoneOffset()	Returns the timezone offset in minutes for the current locale.
getMonth()	Returns the month.
setDate()	Sets the day of the month.
setFullYear()	Sets the full year.
setHours()	Sets the hours.
setMinutes()	Sets the minutes.
setSeconds()	Sets the seconds.
setMilliseconds()	Sets the milliseconds.
setTime()	Sets the number of milliseconds since January 1, 1970 at 12:00 AM.
setMonth()	Sets the month.
toDateString()	Returns the date portion of the Date as a human-readable string.
toLocaleString()	Returns the Date object as a string.
toGMTString()	Returns the Date object as a string in GMT timezone.
valueOf()	Returns the primitive value of a Date object.

```
<html>
    <body>
    <center>
        <h2>Date Methods</h2>
        <script type="text/javascript">
            var d = new Date();
            document.write("<b>Locale String:</b> " +
d.toLocaleString()+"<br>");
            document.write("<b>Hours:</b> " +
d.getHours()+"<br>");
            document.write("<b>Day:</b> " + d.getDay()+"<br>");
            document.write("<b>Month:</b> " +
d.getMonth()+"<br>");
            document.write("<b>FullYear:</b> " +
d.getFullYear()+"<br>");
            document.write("<b>Minutes:</b> " +
d.getMinutes()+"<br>");
        </script>
    </center>
    </body>
</html>
```

String Object

- String objects are used to work with text.
- It works with a series of characters.
- **Syntax:**
  var variable_name = new String(string);
  Example:
  var s = new String(string);

String Properties

Properties	Description
length	It returns the length of the string.
prototype	It allows you to add properties and methods to an object.
constructor	It returns the reference to the String function that created the object.

String Methods

Methods	Description
charAt()	It returns the character at the specified index.
charCodeAt()	It returns the ASCII code of the character at the specified position.
concat()	It combines the text of two strings and returns a new string.
indexOf()	It returns the index within the calling String object.
match()	It is used to match a regular expression against a string.
replace()	It is used to replace the matched substring with a new substring.
search()	It executes the search for a match between a regular expression.
slice()	It extracts a session of a string and returns a new string.
split()	It splits a string object into an array of strings by separating the string into the substrings.
toLowerCase()	It returns the calling string value converted lower case.
toUpperCase()	Returns the calling string value converted to uppercase.

```
<html>
    <body>
    <center>
        <script type="text/javascript">
            var str = "CareerRide Info";
            var s = str.split();
            document.write("<b>Char At:</b> " +
str.charAt(1)+"<br>");
            document.write("<b>CharCode At:</b> " +
str.charCodeAt(2)+"<br>");
            document.write("<b>Index of:</b> " +
str.indexOf("ide")+"<br>");
            document.write("<b>Lower Case:</b> " +
str.toLowerCase()+"<br>");
            document.write("<b>Upper Case:</b> " +
str.toUpperCase()+"<br>");
        </script>
    <center>
    </body>
</html>
```