Assignment 2

Q1.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Example of console objects</title>

</head>

<body>

<script defer src = "Q1.js">

</script>

<h1>

Methods of console object

</h1>

<ol>

<h2>

<li>

console.log() :

</li>

</h2>

<p>

console.log() is used to log/print/output a message on the console. It can display any form of data such as an array, string, boolean, etc.

</p>

<b>

Example : <br>

</b>

<i>

console.log(1234); <br>

console.log("Some string"); <br>

console.log([0, 1, 2]); <br>

console.log({1 : "xyz", 2 : "abc"}); <br>

</i>

<h2>

<li>

console.error() :

</li>

</h2>

<p>

console.error() is used for testing the code and display error messages. Error messages are displayed/highlighted in red color.

</p>

<b>

Example : <br>

</b>

<i>

console.error("Some random error"); <br>

</i>

<h2>

<li>

console.warn() :

</li>

</h2>

<p>

console.warn() is used to display/log warnings in the console. Highlighted in yellow color.

</p>

<b>

Example : <br>

</b>

<i>

console.warn("Some warning");

</i>

<h2>

<li>

console.clear() :

</li>

</h2>

<p>

As the name suggests, it is used to clear the console. In Chrome, “Console was cleared” message will be displayed after, clearing the console. In Firefox, no message is displayed after clearing the console.

</p>

<b>

Example : <br>

</b>

<i>

console.clear();

</i>

<h2>

<li>

console.time() and console.timeEnd() :

</li>

</h2>

<p>

These functions are used when we want to know/calculate the time spend by a function or block of code to execute. Both functions take a label, which must be the same, and code inside it can be anything.

</p>

<b>

Example : <br>

</b>

<i>

console.time(); <br>

let func = function() <br>

{ <br>

console.log("Function Running"); <br>

} <br>

func(); <br>

console.timeEnd(); <br>

</i>

<h2>

<li>

console.table() :

</li>

</h2>

<p>

Used to generate a table inside the console. Input must be an object/array.

</p>

<b>

Example : <br>

</b>

<i>

console.table({1 : "xyz", 2 : "abc"});

</i>

<h2>

<li>

console.count() :

</li>

</h2>

<p>

console.count() will print the number of times it is called. A label can be added to the console view. By default, the label is “default”.

</p>

<b>

Example : <br>

</b>

<i>

for(i = 0; i<=3; i++) <br>

{ <br>

console.count(); <br>

} <br>

// With some label <br>

for(i = 0; i<=3; i++) <br>

{ <br>

console.count("Some Label"); <br>

} <br>

</i>

<h2>

<li>

console.group() and console.groupEnd() :

</li>

</h2>

<p>

These methods allows to group contents in a seperate intented block. These functions also accepts labels. Both functions take a label, which must be the same, and code inside it can be anything.

</p>

<b>

Example : <br>

</b>

<i>

console.group("Group"); <br>

console.log(1234); <br>

console.log("Some string"); <br>

console.log([0, 1, 2]); <br>

console.log({1 : "xyz", 2 : "abc"}); <br>

console.groupEnd("Group"); <br>

</i>

</ol>

<mark>

<b>

To see the Output of Example, press F12 on the keyboard.

</b>

</mark>

</body>

</html>

Q1.js

// Example of console.clear()

console.clear()

// Example of console.log()

console.log(1234);

console.log("Some string");

console.log([0, 1, 2]);

console.log({1 : "xyz", 2 : "abc"});

// Example of console.error()

console.error("Some random error");

// Example of console.warn()

console.warn("Some warning");

/// Example of console.time() and console.timeEnd()

console.time();

let func = function()

{

console.log("Function Running");

}

func();

console.timeEnd();

// Example of console.table()

console.table({1 : "xyz", 2 : "abc"});

// Example for console.count()

for(i = 0; i<=3; i++)

{

console.count();

}

// With some label

for(i = 0; i<=3; i++)

{

console.count("Some Label");

}

// Example for console.group() and console.groupEnd()

console.group("Group");

console.log(1234);

console.log("Some string");

console.log([0, 1, 2]);

console.log({1 : "xyz", 2 : "abc"});

console.groupEnd("Group");

Q2.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Difference Between var, let and const</title>

</head>

<body>

<script defer src = "Q2.js">

</script>

<center>

<h1>

Difference Between var, let and const

</h1>

</center>

<ol>

<h2>

<li>

var :

</li>

</h2>

<p>

The scope of the variable is global or local i.e. function scoped. <br> The scope is global when, the var variable is declared outside and function/block of code. These variable is availabe to use in the whole code. <br> The var variable declared within a function accessible with that function. <br> If the variable is not declared and it is accessed, it gives "undefined".

</p>

<b>

Example : <br>

</b>

<i>

console.log(declare); // Undefined <br>

var declare; <br>

<br>

function getCount() <br>

{ <br>

var count; <br>

for(var i = 0; i < 10; i++) <br>

{ <br>

var c = i; <br>

count = i - 1; <br>

} <br>

<br>

console.log(i); <br>

console.log(c); <br>

console.log(count); <br>

} <br>

getCount(); <br>

console.log(count); // Reference Error <br>

</i>

<h2>

<li>

let :

</li>

</h2>

<p>

var is also global scoped. Instead of being function scoped, let is block scoped. This means a variable created with let is available inside the block in which it is created as well as nested blocks.<br>

Block here means any code surrounded into curly braces {} like if statement, for loop, while loop, etc. <br> If a variable is not declared and it is accessed, "ReferenceError" will be given.

</p>

<b>

Example : <br>

</b>

<i>

console.log(declar); // Reference Error <br>

let declar; <br>

<br>

function getCount() <br>

{ <br>

let count; <br>

for(let i = 0; i < 10; i++) <br>

{ <br>

let c = i; <br>

count = i - 1; <br>

} <br>

<br>

console.log(i); // Reference Error <br>

console.log(c); // Reference Error <br>

console.log(count); <br>

} <br>

getCount(); <br>

console.log(count); // Reference Error <br>

<br>

let pie = 3.14; <br>

pie = 22 / 7; <br>

</i>

<h2>

<li>

const :

</li>

</h2>

<p>

const is same as let, only difference is that once the value is assigned, it cannot be changed. <br> Once the value is assigned it cannot be changed.

</p>

<b>

Example : <br>

</b>

<i>

const pi = 3.14; <br>

pi = 22 / 7; // Type Error <br>

</i>

</ol>

<center>

<mark>

<b>

To see the Output of Example, press F12 on the keyboard.

</b>

<br>

<i>

Note : All the error raised are commented.

</i>

</mark>

</center>

</body>

</html>

Q2.js

// var section

console.log(declare); // Undefined

var declare;

function getCount()

{

var count;

for(var i = 0; i < 10; i++)

{

var c = i;

count = i - 1;

}

console.log(i);

console.log(c);

console.log(count);

}

getCount();

//console.log(count); // Reference Error

// let section

//console.log(declar); // Reference Error

//let declar;

function getCount()

{

let count;

for(let i = 0; i < 10; i++)

{

let c = i;

count = i - 1;

}

// console.log(i); // Reference Error

// console.log(c); // Reference Error

console.log(count);

}

getCount();

// console.log(count); // Reference Error

let pie = 3.14;

pie = 22 / 7;

// const section

//const pi = 3.14;

//pi = 22 / 7; // Type Error

Q3.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Datatypes in Javascript</title>

</head>

<body>

<center>

<h1>

Datatypes in Javascript

</h1>

</center>

<ul>

<li>

<h2>

Primitive Datatypes :-

</h2>

</li>

<ol>

<h3>

<li>

Number -

</li>

</h3>

<p>

Represents both integers and floating point numbers. Many operations can be performed on numbers, such as addition, subtraction, multiplication, etc. <br> There are special numeric values that, also belong to this datatype such as "Infinity", "-Infinity" and "NaN".

</p>

<h3>

<li>

BigInt -

</li>

</h3>

<p>

In Javascript, the number datatype cannot represent positive numbers greater than (2<sup>53</sup> - 1) or the negative numbers less than -(2<sup>53</sup> - 1). This technical limitation is caused by some internal representation. <br> Most of the purpose is fulfilled by this limit but, in some cases we require to work with really large numbers. <br> BigInt was introduced to represent numbers of arbitary length. A BigInt is created by appending <b>n</b> to the end of integer.

</p>

<h3>

<li>

String -

</li>

</h3>

<p>

In Javascript, string is represented in the quotes. It can be either single or double qoutes i.e. If the string is started wkith single quote, it should end with single qoutes only. Same goes for double quotes.

</p>

<h3>

<li>

Boolean -

</li>

</h3>

<p>

Boolean or logical type can store only two values, either true or false.<br> true means/represents yes or 1.<br> false means/represents no or 0.

</p>

<h3>

<li>

null -

</li>

</h3>

<p>

This is a special/seperate type which doesn't matches with any other datatype. It represents "nothing", "empty" or "unknown value".

</p>

<h3>

<li>

undefined -

</li>

</h3>

<p>

This is also a special type of value/datatype which stands apart from any other value. It means the "any value is not assigned to it".<br> If a variable is declared but, not assigned, then it's value is undefined.

</p>

</ol>

<li>

<h2>

Non-Primitive Datatypes :-

</h2>

</li>

<ol>

<h3>

<li>

Object -

</li>

</h3>

<p>

Object represents instance of a class through, which we can access the class members.

</p>

<h3>

<li>

Array -

</li>

</h3>

<p>

Array can store group of similar type of value. Multiple values can be stored in a single variable.

</p>

<h3>

<li>

RegExp -

</li>

</h3>

<p>

Regular Expression is a sequence of characters that forms a search pattern. Searching pattens can be used for text manipulation and text search. It can be single character or a very complex pattern.

</p>

</ol>

</ul>

</body>

</html>