**ASSIGNMENT 2**

**Question 1**

Methods of Console

The following are the types of console.

* [**console.assert()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/assert)

Log a message and stack trace to console if the first argument is false.

* **[console.clear()](https://developer.mozilla.org/en-US/docs/Web/API/Console/clear)**

Clear the console.

* [**console.count()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/count)

Log the number of times this line has been called with the given label.

* [**console.countReset()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/countReset)

Resets the value of the counter with the given label.

* [**console.debug()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/debug)

Outputs a message to the console with the log level "debug".

* [**console.dir()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/dir)

Displays an interactive listing of the properties of a specified JavaScript object. This listing lets you use disclosure triangles to examine the contents of child objects.

* [**console.error()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/error)

Outputs an error message. You may use [string substitution](https://developer.mozilla.org/en-US/docs/Web/API/console#Using_string_substitutions) and additional arguments with this method.

* **console.exception()**

An alias for error().

* [**console.group()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/group)

Creates a new inline [group](https://developer.mozilla.org/en-US/docs/Web/API/console#Using_groups_in_the_console), indenting all following output by another level. To move back out a level, call groupEnd().

* [**console.groupEnd()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/groupEnd)

Exits the current inline [group](https://developer.mozilla.org/en-US/docs/Web/API/console#Using_groups_in_the_console).

* [**console.info()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/info)

Informative logging of information. You may use [string substitution](https://developer.mozilla.org/en-US/docs/Web/API/console#Using_string_substitutions) and additional arguments with this method.

* [**console.log()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/log)

For general output of logging information. You may use [string substitution](https://developer.mozilla.org/en-US/docs/Web/API/console#Using_string_substitutions) and additional arguments with this method.

* [**console.profile()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/profile)

Starts the browser's built-in profiler (for example, the [Firefox performance tool](https://developer.mozilla.org/en-US/docs/Tools/Performance)). You can specify an optional name for the profile.

* [**console.table()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/table)

Displays tabular data as a table.

* [**console.time()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/time)

Starts a [timer](https://developer.mozilla.org/en-US/docs/Web/API/console#Timers) with a name specified as an input parameter. Up to 10,000 simultaneous timers can run on a given page.

* [**console.timeEnd()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/timeEnd)

Stops the specified [timer](https://developer.mozilla.org/en-US/docs/Web/API/console#Timers) and logs the elapsed time in seconds since it started.

* [**console.timeLog()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/timeLog)

Logs the value of the specified [timer](https://developer.mozilla.org/en-US/docs/Web/API/console#Timers) to the console.

* [**console.trace()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/trace)

Outputs a [stack trace](https://developer.mozilla.org/en-US/docs/Web/API/console#Stack_traces).

* [**console.warn()**](https://developer.mozilla.org/en-US/docs/Web/API/Console/warn)

Outputs a warning message. You may use [string substitution](https://developer.mozilla.org/en-US/docs/Web/API/console#Using_string_substitutions) and additional arguments with this method.

**Question 2**

Difference between var, let and const are as follows:

* Var declarations are globally scoped or function scoped while let and const are block scoped.
* Var variables can be updated and re-declared within its scope; let variables can be updated but not re-declared; const variables can neither be updated nor re-declared.
* They are all hoisted to the top of their scope. But while var variables are initialized with undefined, let and const variables are not initialized.
* While var and let can be declared without being initialized, const must be initialized during declaration.

**Question 3**

JavaScript provides different data types to hold different types of values.

Two types of data types in JavaScript:

1. Primitive data type
2. Non-primitive data type

**Primitive data type**

There are five types of primitive data as follows:

* String - represents sequence of characters e.g. "hello"
* Number - represents numeric values e.g. 100
* Boolean - represents boolean value either false or true
* Undefined - represents undefined value
* Null- represents null i.e. no value at all

**Non-primitive data type**

Types are as follows:

* Object - represents instance through which we can access members
* Array - represents group of similar values
* RegExp - represents regular expression