Lets Upgrade

Assignment Day 1 | 14th July 2020

For any doubts regarding the assignment, ask questions in the Javascript Group in the Community.

Submit the Assignment by 15th July 2020 2:00 PM.

Assignment Submit Form : https://forms.gle/tmBTKNZsW1pmD4SQ8

Question 1 :

Explore and explain the various methods in console function
Explain them
Ex. console.log()
console.warn().
etc...

Question 2 :

Write the difference between var, let and const with code examples.

Question 3 :

Write a brief intro on available data types in Javascript.

Ans 1:

1. **console.assert() Method:-** The console.assert() method writes a message to the console, but only if an expression evaluates to *false*.
   Eg: console.assert(document.getElementById("demo"), "You have no element with ID 'demo'");
2. **console.clear() Method:-** The console.clear() method clears the console. The console.clear() method will also write a message in the console: "Console was cleared".
3. **console.count() Method:-** Writes to the console the number of times that particular console.count() is called.
   Eg:
   ```
   for (i = 0; i < 10; i++) {
         console.count();
         }
   ```

4. **console.error() Method:-** The console.error() method writes an error message to the console. The console is useful for testing purposes.
   Eg: console.error("You made a mistake");

5. **console.group() Method:-** The console.group() method indicates the start of a message group.
   Eg:
   ```
   console.log("Hello world!");
         console.group();
         console.log("Hello again, this time inside a group!");
   ```

6. **console.groupCollapsed() Method:-** The console.groupCollapsed() method indicates the start of a collapsed message group.
   Eg:
   ```
   console.log("Hello world!");
         console.groupCollapsed();
         console.log("Hello again, this time inside a collapsed group!");
   ```

7. **console.groupEnd() Method:-** The console.groupEnd() method indicates the end of a message group.
   Eg:
   ```
   console.log("Hello world!");
         console.group();
         console.log("Hello again, this time inside a group!");
         console.groupEnd();
         console.log("and we are back.");
   ```

8. **console.groupEnd() Method:-** The console.info() method writes a message to the console.
   Eg: console.info(“Hello World!”);

9. **console.log() Method:-** The console.log() method writes a message to the console.
Eg: console.log(“Hello World!”);

10. **console.table() Method:-** The console.table() method writes a table in the console view.
Eg: console.table([“Orange”, “Apple”, “Kiwi”]);

11. **console.time() Method:-** The console.time() method starts a timer in the console view.
Eg:
```
console.time();
for (i = 0; i < 100000; i++) {
 // some code
}
console.timeEnd();
```

12. **console.timeEnd() Method:-** The console.timeEnd() method ends a timer, and writes the result in the console view.
Eg:
```
console.time();
for (i = 0; i < 100000; i++) {
 // some code
}
console.timeEnd();
```

13. **console.trace():-** The console.trace() method displays a trace that show how the code ended up at a certain point.
Eg:
```
function myFunction() {
   myOtherFunction();
}
function myOtherFunction() {
   console.trace();
}
```

14. **console.warn():-** The console.warn() method writes a warning to the console.
Eg: console.warn(“This is a warning!”);

<u>Ans 2:</u> <u>var VS let VS const:-</u>

First, let’s compare var and let. The main difference between var and let is that instead of being function scoped, let is block scoped. What that means is that a variable created with the let keyword is available inside the “block” that it was created in as well as any nested blocks. When I say “block”, I mean anything surrounded by a curly brace {} like in a for loop or an if statement.

var	**let**	**const**
Function Scoped	Block Scoped	Block Scoped
Undefined when accessing a variable before it’s declared.	ReferenceError when accessing a variable before it’s declared.	ReferenceError when accessing a variable before it’s declared and can’t be reassigned.

Turns out, const is almost exactly the same as let. However, the only difference is that once you’ve assigned a value to a variable using const, you can’t reassign it to a new value.

var is function scoped and if you try to use a variable declared with var before the actual declaration, you’ll just get undefined. const and let are blocked scoped and if you try to use variable declared with let or const before the declaration you’ll get a ReferenceError. Finally, the difference between let and const is that once you’ve assigned a value to const, you can’t reassign it, but with let, you can.

<u>Example:</u>

i.
```
var carName = "Volvo";
```

ii.
```
var x = 10;
// Here x is 10
{
  let x = 2;
  // Here x is 2
}
// Here x is 10
```

iii.
```
let carName = "Volvo";
```

iv.
```
var x = 10;
// Here x is 10
{
  const x = 2;
  // Here x is 2
```

```
       }
       // Here x is 10
```

V. `const PI = 3.14159265359;`

<u>**Ans 3:**</u> JavaScript variables can hold many **data types**: numbers, strings, objects and more. JavaScript evaluates expressions from left to right. Different sequenced can produce different results. JavaScript has dynamic types. This means that the same variable can be used to hold different data type.

<u>**Example:**</u>

```
a. var x;           // Now x is undefined
b. x = 5;           // Now x is a Number
c. x = "John";      // Now x is a String
d. var y = 123e5;      // 12300000
   var z = 123e-5;     // 0.00123
e. var x = 5;
   var y = 5;
   var z = 6;
   (x == y)       // Returns true
   (x == z)       // Returns false
f. var cars = ["Saab", "Volvo", "BMW"]; //arrays
g. var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

==