Web Computing - Assignment 2.

Q1.

→

It is important to validate the form submitted by the user because it can have inappropriate values. So, validations is a must to authenticate users. Javascript provides facility to validate the form on the client side, so data processing will be faster than server-side validation. Most of the web developers prefer Javascript form validation.

Through JS, we can validate name, password, email, date, mobile number and much more.

Basic Validation:

First of all, the form must be checked to make sure that all the mandatory fields are filled in. It would require just a loop through each field in the form and check for data.

Data format Validation -

Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.

Q2.

→

Arrow function is one of the features introduced in the ES6 version of JavaScript. It allows you to create functions in a cleaner way compared to regular functions.

Eg. let x = function (x,y) {

return x*y;

}

using arrow functions it can be written as
let x = (x,y) → x*y;

Eg 1. arrow function with no args.

```
let greet () => console.log ("Hello");
greet();
```

Eg 2. arrow func with one arg.

```
let greet = (x) => console.log (x);
greet ("Hello");
```

Eg 3. arrow func (multiline) with multiple args.

```
let sum = (a, b) =>
{  let result = a+b;
   return result;
}
let result = sum(5,7);
console.log (result);
```

Q3.

→ The data structures that have the symbol iterator() method are called iterables.

Eg. Array, Strings, Sets, etc.

```
Const. arr = [1, 2, 3, 4]
Const. arriterator = arr [Symbol.iterator] ();
console.log (arr.iterator);
```

output → 1 2 3 4

Q4.

→ An iterator is an object that is returned by the Symbol.iterator() method.

The iterator protocol provides the next() method to access each element of the iterable at a time.

Eg. const number = [1, 2, 3];
for (let n of number[Symbol.iterator]())
{
  console.log(n);
}

Output: 1 2 3.

Q5.

→ In Javascript, generators provide a new way to work with functions & iterators.

One can stop the execution of a function from anywhere inside the function and continue execution code from a halted position.

Defining a generator function.

function * generator_function()
{ ... }

const generator_obj = generator_function();

Yield is used to pause execution

Example:

```
function *generatorFunc()
{
    console.log("1. Code before yield");
    yield 100;
    console.log("2. Code after yield");
    yield 200;
}
const generator = generatorFunc();
console.log(generator.next());
```

Output:

1- code before the yield.

{value: 100, done: false}

~~2- code of~~

Whenever generator.next() is called, the code up to the first yield is executed. When yield is encountered the program returns the value and pauses the generator function.

Q6.

→ A Javascript function is a block of code designed to perform a particular task. A function is called executed when "something" evokes it.

Defined by function keyword, followed by a name, followed by parantheses ().

Eg. function fun(p1, p2) {
    return p1*p2 }

Q7.

→ Transport Layer Security (TLS) is designed to provide security at the transport layer. TLS was derived from a security protocol called Secure socket layer (SSL).

TLS ensures that no third party may eavesdrop or tamper with any message.

Benefits of TLS are:

i) Encryption.
ii) Interpermeability
iii) Algorithm flexibility
iv) Ease of Deployment
v) Ease of Use.

Q8.

→ Recursion is a process of calling itself. A function that calls itself is called a recursive function. When a function calls itself recursively, such a type of function is called a recursive function, and this approach is called recursion.

A recursion function must have a condition so that it should stop its execution or else the function will be invoked & executed infinitely.

Syntax:

```
function recurse () {
    ...
    recurse ();
    //
}
```

Eg.
```
function factorial (number) {
    if (number <= 2)
        return number;
    else
        return factorial (number-1) * number;
}
factorial (5);
```

Qa.

→ Inheritance enables / allows us to define a class that takes all the functionality from a parent class and allows you to add more.

Using class inheritance, a class can inherit all the methods and properties of another class. Inheritance is an useful feature that allows code reusability.

Eg.
```
class Student {
    constructor (a) {
        this.name = a;
    }
}
class User extends Student
{
    show () {
        console.log ( this.name);
    }
}
var obj = new User ("Yash");
obj.show ();
```

In the above example, we have declared a class student. By using the extends keyword, we can create a new class user that shows the same characteristics as the parent class Student.

Q10., Q13., Q.17., Q19.

→ Done in assignment 1.

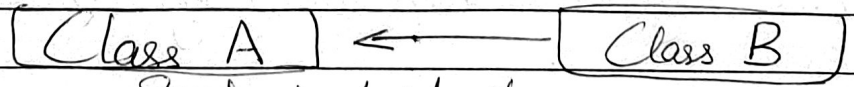
Q16.

→ Types of inheritance
Single level inheritance
Multi level inheritance
Multiple inheritance.

Inheritance can be categorized as Single-level, multilevel and multiple inheritance.
Multiple inheritance is not supported in ESC.

Single-Level Inheritance

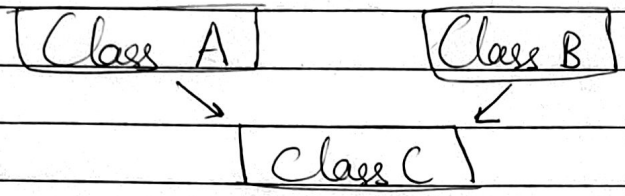
It is defined as the inheritance in which a derived class can only be inherited from only one base class. Allows a derived class to inherit the behaviors and properties of a base class, which enables the reusability of code.



Single level inheritance.

Multiple Inheritance.

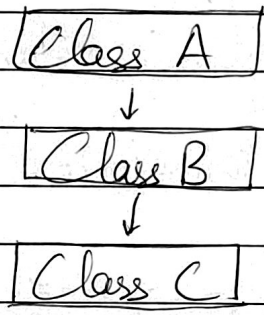
A class can be inherited from multiple classes.
Not supported in ES6.



multiple inheritance

Multi-level Inheritance.

A derived class is inherited from another derived class.



multi-level inheritance.

Q 12)

→ Objects of DOM are:

i) Window Object - object of the browser which is always at the top of the hierarchy.

ii) Document Object - when an HTML document is loaded into a window, it becomes a document object.

iii) Form Object - Represented by form tags.

iv) Link Object - Represented by link tags.

Objects of BOM are:

i) Window object - Represents a window in browser.
ii) JS history object - represents an array of URLs visited by the user.
iii) JS Navigation Object - used for browser detection, can be used to get browser info.
iv) JS screen objects - holds information of the browser screen.

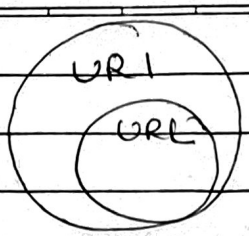
Q(14)

→

	SSL	TLS
i)	Secure Socket Layer	• Transport Layer Security.
ii)	Three versions: 1.0, 2.0 & 3.0	• Four versions: 1.0, 1.1, 1.2, 1.3.
iii)	All versions have been found to be vulnerable.	• 1.0 and 1.1 have been found vulnerable.
iv)	All versions deprecated.	• 1.2 is used widely.
v)	Cryptographic protocol that uses explicit connections to establish secure communication.	• Cryptographic protocol that provides secure communication between web server & client.
vi)	SSL creates a master secret, the message digest of the pre-master secret is used.	• TLS uses a pseudo-random function to generate the master secret.

Q(15)

→ URL stands for Uniform Resource Identifier, it is a character sequence that identifies a logical (abstract) or physical resource connected to the internet. A URI distinguishes one resource from another.



diagrammatic representation of URI & URL.

Q16]

A cookie is an amount of data (information) that persists between a server state side and client side. A web browser stores this information at the time of browsing. A cookie contains the information as a string generally in the form of a name-value pair seperated by semi-colon.

Maintains the state of a user and remembers the users information among all the web pages.

Q18]

Change in the state of an object is known as an event. In HTML, there are various events which represents that some activity is performed by the user or by the browser.

When js code is included in the HTML, js reacts over these events and allow the execution. This process of ~~reacting~~ reacting over the events is called event handling.

Thus ~~HTML to~~ JS handles HTML events via event handlers.

Q 20)

A function is a block of code that performs a specific task. A function is a group of reusable code which can be called anywhere in the program.

This exterminates the need of writing the same code multiple times. JS also supports all the features necessary to write modular code using functions.

i] function without argument and return values.

Syntax:
```
function random() {
    //code
}
```

ii) function without arguments ~~and~~ but a return value.

Syntax:
```
function return_1() {
    //code
    return 1;
}
```

iii) function with arguments and return value.

Syntax:
```
function (arg1, arg2, ....) {
    //code
    return result;
}
```

iv) function with arguments without return values.

Syntax:
```
function (arg1, arg2, ....) {
    //code
}
```

Q21.)

	ES5	ES6
i)	5th edition of ECMA	• 6th edition of ECMA.
ii)	introduced in 2009	• introduced in 2015.
iii)	ES5 supports primitive data types that are string, null, etc	• ES6 supports additional JS data types 'symbol'.
iv)	We could define the variables only using 'var'.	• let & const are two new ways in ES6
v)	has lower performance than ES6	• higher performance
vi)	supported by wide range of communications	• comparitively less support.
vii)	ES5 is time consuming.	• Object manipulation takes less time.
viii)	for loop is used to iterate over elements.	• ES6 introduced the concept of using for loop for performing iterations over values of iterable objects.