CBT - 1

XML HttpRequest

- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background
- The responseText property returns the server response as a text string.

Window.history

- The window.history object contains the browsers history.

Javascript Basics

Javascript Objects : A JavaScript object is a collection of named values

- A primitive value is a value that has no properties or methods
- JavaScript variables can contain single values : var person = "John Doe"; // All strings are objects
- The values are written as name : value pairs (name and value separated by a colon). :
  var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
- Methods are actions that can be performed on objects
- Objects are mutable: They are addressed by reference, not by value.
  var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"}
  var x = person;
  x.age = 10; // This will change both x.age and person.age
- You can add new properties to an existing object by simply giving it a value.
- Delete Keyword
    - The delete keyword deletes a property from an object:
      var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
      delete person.age; // or delete person["age"];
    - The delete keyword deletes both the value of the property and the property itself.
    - After deletion, the property cannot be used before it is added back again.
    - The delete operator is designed to be used on object properties. It has no effect on variables or functions.

    - The delete operator should not be used on predefined JavaScript object properties. It can crash your application.
    - The delete keyword does not delete inherited properties, but if you delete a prototype property, it will affect all objects inherited from the prototype.

- The call() and apply() methods are predefined JavaScript methods.
  var person1 = {
  fullName: function() {
  return this.firstName + " " + this.lastName;
  }
  }
  var person2 = {
  firstName:"John",
  lastName: "Doe",
  }
  person1.fullName.call(person2); // Will return "John Doe"
  whenever you want to invoke an other objects methods on an abject you invoke call() or apply()

AJAX = Asynchronous JavaScript And XML.
- Need for Asynchronous Requests -:
* By sending asynchronously, the JavaScript does not have to wait for the server response, but can instead:
* execute other scripts while waiting for server response
* deal with the response after the response is ready

```
- Synchronous XMLHttpRequest (async = false) is not recommended because the
JavaScript will stop executing until the server response is ready. If the server
is busy or slow, the application will hang or stop.

- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

- XML Http Request
        + Holds the status of the XMLHttpRequest.
                - 0: request not initialized
                - 1: server connection established
                - 2: request received
                - 3: processing request
                - 4: request finished and response is ready

- Using A Callback function
        - A callback function is a function passed as a parameter to another
function.

        - loadDoc("url-1", myFunction1);
          loadDoc("url-2", myFunction2);
```