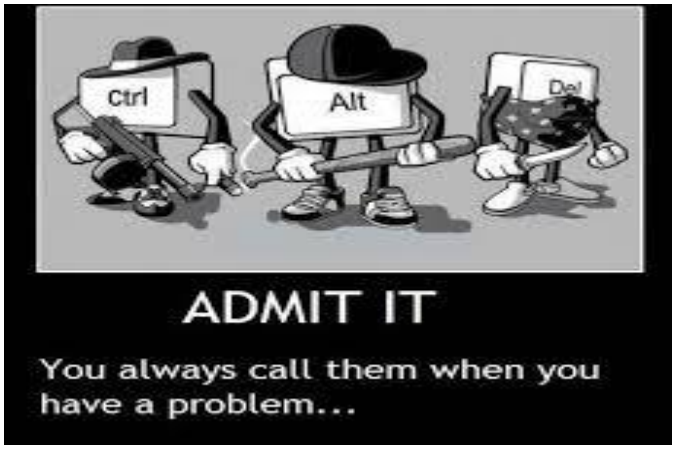
Programming Languages

Constructed language designed to give instructions to a machine

Uses

- Create programs
- Control the behavior of a machine
- Express Algorithms



Programming Languages

```
[6]    L←(Lι':')↓L←,L          ⍝ drop To:
[7]    L←LJUST VTOM',',L       ⍝ mat with one entry per row
[8]    S←¯1++/∧\L≠'('          ⍝ length of address
[9]    X←0⌈⌈/S
[10]   L←SΦ(-(ρL)+0,X)↑L       ⍝ align the (names)
[11]   A←((1↑ρL),X)↑L          ⍝ address
[12]   N←0 1↓DLTB(0,X)↓L       ⍝ names)
[13]   N←,'α',N
[14]   N[(N='_')/ιρN]←' '      ⍝ change _ to blank
[15]   N←0 ¯1↓RJUST VTOM N     ⍝ names
[16]   S←+/∧\' '≠ΦN            ⍝ length of last word in name
```

Types

- Array languages: A+, APL, J, Julia, K, Matlab, Python, IDL
- Authoring languages: Bigwig, PILOT, TUTOR, Lasso
- Command line interface languages: bash, CHAIN, Expect
- Compiled languages: ActionScript, Blue, C, C++, Cobra, Erlang, Java, Objective-C, Python, Scala, Swift, Visual Basic
- Data-oriented languages: MUMPS, SQL, WebQL

```
#include<iostream>
using namespace std;
int main() {

 int first_number , second_number, gcd;

  cout<<"Enter First Number : ";cin>>first_number;
  cout<<"Enter Second Number: ";cin>>second_number;

 for(int i=1;i<=first_number&&i<=second_number;i++){

    if(first_number%i==0 && second_number%i == 0 ){
                        gcd=i;
      }
}
cout<<"Greatest Common Divison (GCD):"<<gcd<<endl;
return 0;
}
```

JavaScript Makes the Web Dynamic

JavaScript was created in 1996 by Netscape (the original web browser) to allow for web scripting, or running a program on a website's server that reacts to user action.

This creates web pages that change and react based on the actions of the user.

JavaScript Uses

Make web pages responsive

Create alerts

Create animation effects

Create cookies

Validate web form data

Detect Visitor's Browsers for Customization

The 3

ECMAScript

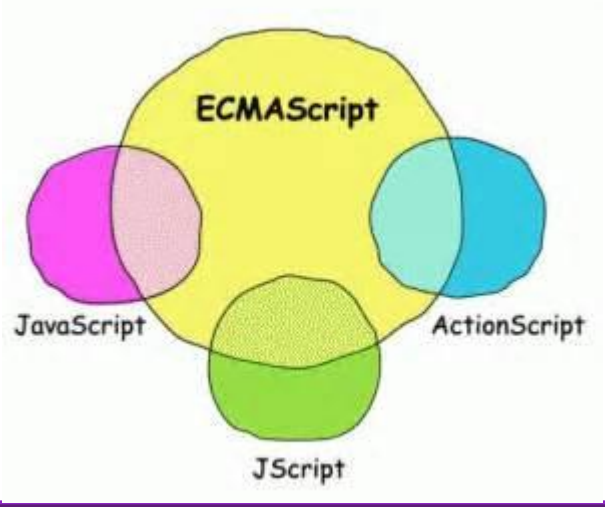
Document Object Model

Browser Object Model

ECMAScript

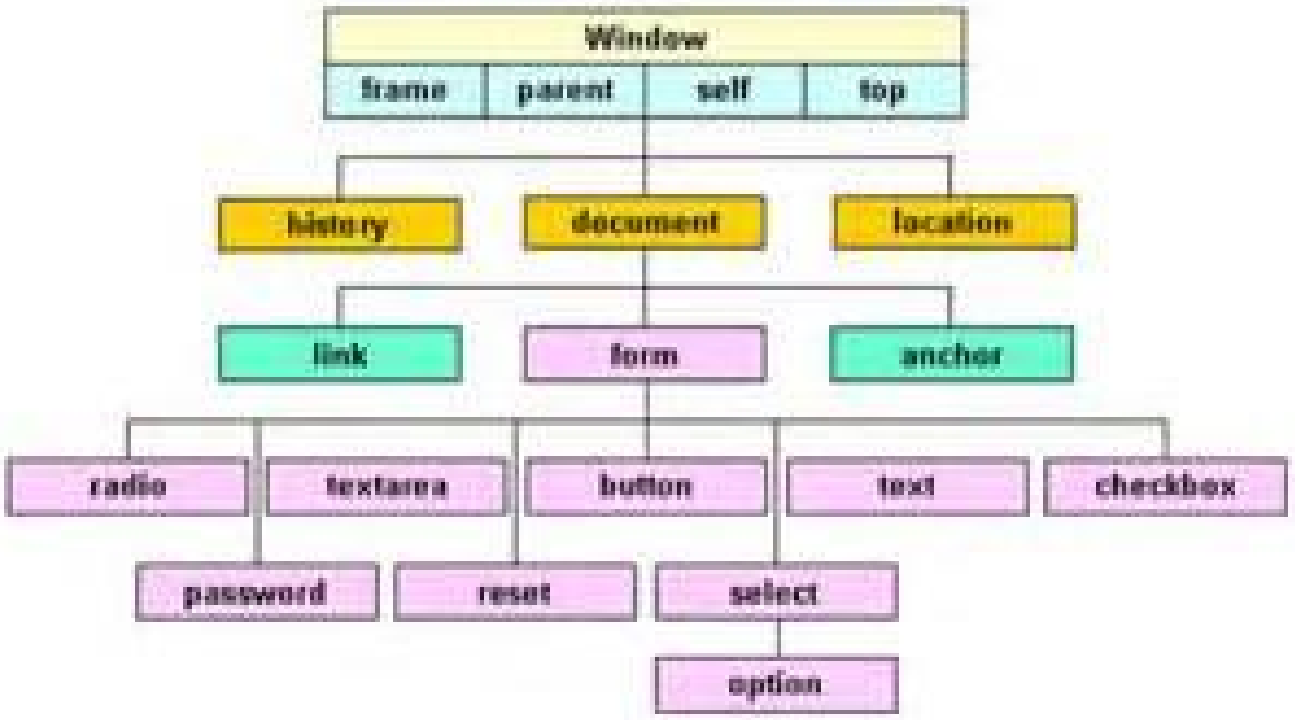
Defines all the properties, methods, and objects of a scripting language. It is a standard for a scripting language.

- ECMA: European Computer Manufacturer's Association
- JavaScript is based on the ECMAScript standard
- Each browser has its own implementation of the ECMAScript interface, which is then extended to contain the DOM and BOM



DOM: Document Object Model

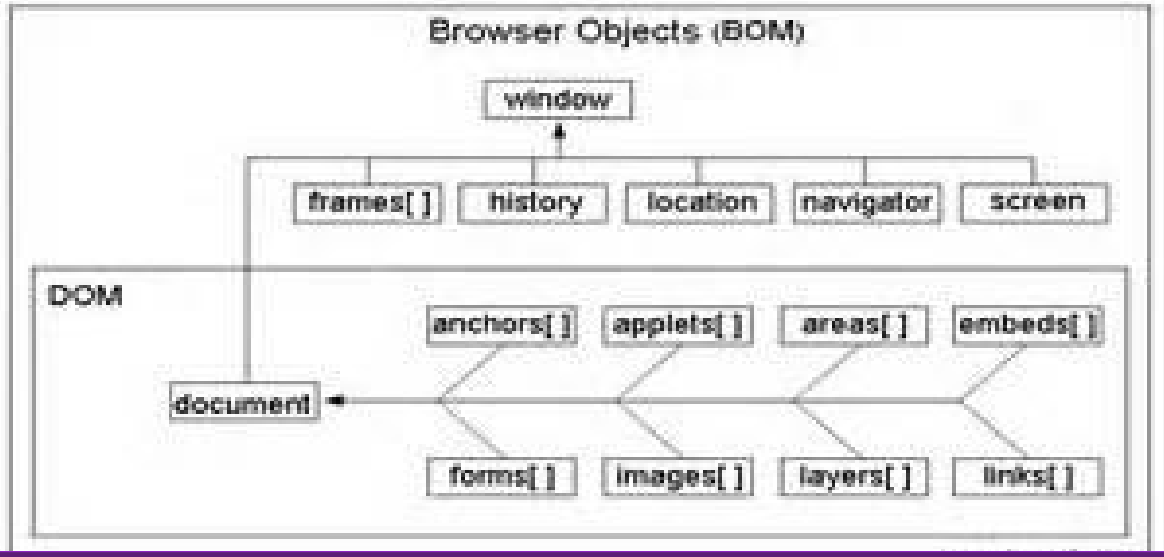
Describes methods and interfaces for working with the content of a webpage



BOM: Browser Object Model

Describes methods and interfaces for interacting with the browser

- Each browser has its own implementation
- Enables JavaScript to talk to the browser
- Window object represents an entire browser window and is supported by all browsers



JQUERY $('a recap')

jQuery is a JavaScript library that simplifies common tasks and hides cross-browser differences

OBTAINING JQUERY

- Allow a Content Distribution Network to serve it
- Download from www.jquery.com and use <script> tag to include in web pages

jQuery FUNCTION $()

4 ways to invoke the jQuery function

- **Pass a CSS selector**

  $('#container'), $('.odd')

- **Pass an element, document or window object**

  $(document), $('div')

- **Pass a string of HTML**

  $(' HTML in jQuery ')

- **Pass a function**

```
$(document).ready(function() {
    // jQuery code goes here
});
```

jQuery Terminology

DOM -> Document Object Model defines the logical structure of documents and the way a document is accessed and manipulated

the jQuery function or $() -> a function that creates jQuery objects and registers handlers to be invoked when the DOM is ready

a jQuery object -> an object returned by a jQuery function

a jQuery function -> a function defined in ‘the jQuery function’

a jQuery method -> a method of a jQuery object returned by the jQuery function

JQUERY AJAX

AJAX [Asynchronous JavaScript and XML]

is a web application programming technique that uses HTTP scripting to load data, without causing page refreshes e.g. Gmail, Google Maps, Yahoo Maps

jQuery includes AJAX utilities to simplify it.

JQUERY AJAX

1 high level utility method

$.load()

(invoked on a DOM element)

4 low level utility functions

$.getScript(), $.getJSON(), $.get(), $.post()

(invoked directly on jQuery)

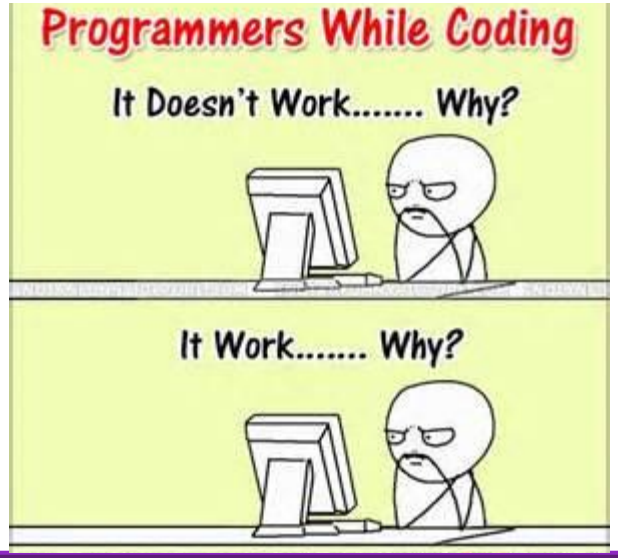
AJAX UTILITY FUNCTIONS

$.getScript() loads and executes JavaScript code

$.getJSON() loads a URL, parses as JSON passes object to callback function

$.get() general purpose URL fetching function

$.post() works like GET but works with data



$.getScript()

jQuery.getScript(url [, success])

```
// Load a library and use it once it loads
$.getScript("js/jquery.my_plugin.js", function() {
        // Use the library we loaded
        $('div').my_plugin();
});
```

$.getJSON()

jQuery.getJSON(url [, data] [, success])

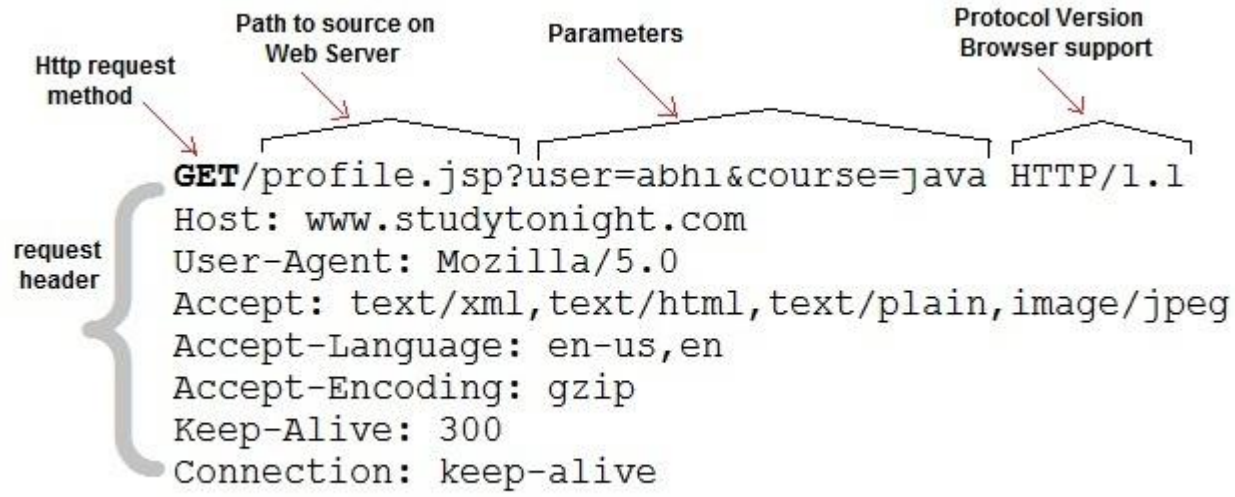
```
// Suppose data.json contains the text: '{"x":1,"y":2}’
$.getJSON("data.json", function(data) {
        // Now data is the object {x:1, y:2}
        console.log(data); // Show data values on the console
});
```

$.get(url [, data] [, success] [, dataType])

```
// Request text and display it in an alert dialog
$.get("debug.txt", alert);
```

$.post(url [, data] [, success] [, dataType])

```
// Send form data using AJAX requests
$.post('test.php', $('#test_form').serialize());
```



$.get() and $.post() can be passed by any of these 6 dataTypes

- text -> returns plain text with processing
- html -> works like 'text' and used by .load() method
- xml -> returns an xml formatted document object
- script -> URL assumed to reference a javaScript file
  $.getScript() uses this type
- json -> URL assumed to reference a JSON formatted file
- jsonp -> URL assumed to reference server-side script that supports JSONP protocol for passing JSON data

Request
More
Information

Submit

$.ajax()

jQuery.ajax(url [, success])

- All AJAX utilities invoke $.ajax() in the background
- Most complicated function in the entire library
- Accepts just a single argument: an options object whose properties specify many details on how to perform the AJAX request

E.g. -> A call to jQuery.getScript() invokes

```
jQuery.ajax({
        type: "GET",            // The HTTP request method.
        url: url,               // The URL of the data to fetch.
        data: null,             // Don't add any data to the URL.
        dataType:"script",      // Execute response as a script.
        success: callback       // Call this function when done.
});
```

$.ajax()

jQuery.ajax(url [, success])

COMMON OPTIONS

type -> specifies HTTP request method

url -> URL to be fetched

data -> Data to be appended to URL

dataType -> specifies dataType to be expected in response

contentType -> specifies HTTP Content-Type header for the request

timeout -> in milliseconds, time before request expires, default is 0

global -> specifies whether jQuery should trigger events that describe progress of AJAX request

$.ajax()

jQuery.ajax(url [, success])

CALLBACK OPTIONS

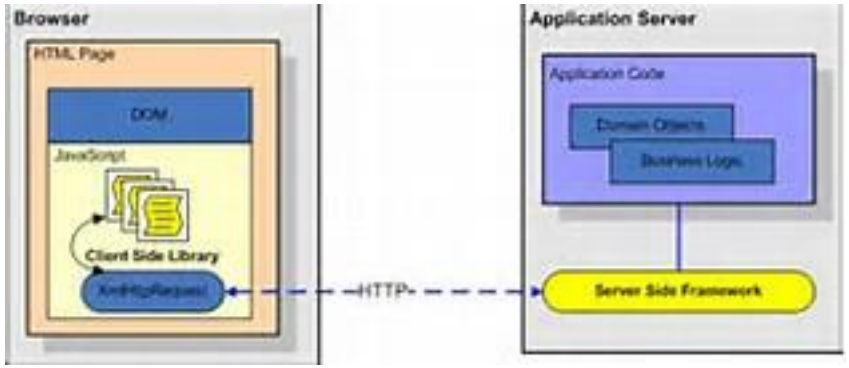
context -> specifies object to be used as context (the this value)

beforeSend -> callback to be invoked before AJAX request is sent to server

success -> call back to be invoked when AJAX request is successful

error -> call back to be invoked when AJAX request is not successful

complete -> call back to be invoked when AJAX request is completed



All AJAX calls invoke a callback function to provide asynchronous notification of the status of the request.

Second argument to the callback is a string with one of the following values:

Success
Not modified
Error
Timeout
parsererror