Web Technology

Basic
- HTML5
- CSS3
- Javascript

Advanced (Frontend)
- AJAX
- JQuery
- JSON
- ANGULAR JS
- REACT JS

Backend
- Full stack
  - (I) Node JS
  - (II) PHP
- Bootstrap 5

Web Application

- Frontend (Client)
- Backend (Server) → Logic (Processing), Data ("Table" / DB)
- "Peers" ← Distributed s/m

* Host – any system connected to a computer network (sharing of "resources")

* Database – collection of inter-related data

* Client - Server Computing
  - a specific device implements the client program and a specific device implements the server program

* Peer to Peer Computing
  - some device can implement both the client and server programs.

Computer Network

↓

"interconnection of n/w of n/w" ← Internet → MAN / LAN → "intra-net" (communication within the LAN) / WAN (communication within WAN is internet)

- Proxy Server: the n/w admin can be considered as a "proxy server" (II) "gateway"
  - data stored in its "cache"
    - for faster access
    - can't hold data for long as it'll get replaced as size is small

- Internet Protocol:
  - set of standards or rules

- Computer Network:
  - Interconnection of autonomous heterogeneous systems.
    - autonomous: own processor, own memory
    - heterogeneous: different platform → operating system + network hardware

"WWW" → World Wide Web
↳ by Tim Berner Lee
↳ web 1.0, web 2.0, web 3.0

- W3C → world wide web consortium

- WWW is a document that is accessed over the internet with the help of some protocols (HTTP)
  - collection of webpages → websites → web applications → web services

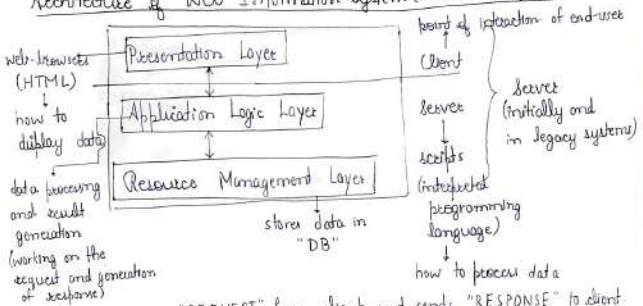
URL (Uniform Resource Locator)

↳ http://www.nitp.ac.in/Department/CSE

(I) "http"/"https"
(II) "www"
(III) "nitp.ac.in" → Domain Name (every host has a unique name over the internet)
(IV) "Department" → folder
(V) "CSE" → subfolder

Protocols
(I) http
(II) WWW
(III) DNS (domain name system)

Architecture of Web Information System (Distributed System)



- Server receives "REQUEST" from client and sends "RESPONSE" to client

"R-R protocol" (HTTP in practice)

ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

Ethernet adapter Ethernet 4:

Connection-specific DNS Suffix . :
Link-local IPv6 address . . . : fe80::11fd:5a96:3178:1887%16
IPv4 address . . . . . : 192.168.56.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Wireless LAN adapter Local Area Connection* 1:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 2:

Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . : fe80::9995:52a0:5390:6325%17
IPv4 Address . . . . . : 192.168.137.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :

Wireless LAN adapter Wi-Fi:

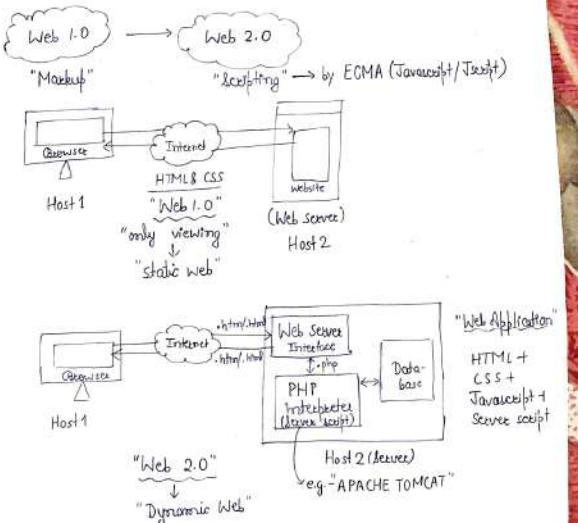
Connection-specific DNS Suffix . :
IPv6 Address . . . . . : 2409:40e4:24:7ee0:8682:4674:5e54:4854
Temporary IPv6 Address . . . . : 2409:40e4:24:7ee0:41ad:826a:f163:6907
Link-local IPv6 Address . . . . : fe80::e7f3:e841:5976:70c%19
IPv4 Address . . . . . : 192.168.52.127
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : fe80::481e:40ff:fef5:c40d%19
192.168.52.207

Ethernet adapter Bluetooth Network Connection:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

Static Web v/s Dynamic Web

Webpage → Website → Web Application



Frontend: HTML + CSS + JScript + AJAX
(Asynchronous Javascript And XML)

- displaying of data
- Extensible Markup language → store and transport of data (XML replaced with JSON (Javascript Object Notation))

"ORKUT" → first social media platform (2006)

"Dynamic Web" → "Social Web"

- single webpage application (SPA) (Ex: Google Maps)
- AJAX

→ by XML

DTD (Document Type Definition)

① <!DOCTYPE PUBLIC "-//W3C //DTD HTML 4.0 / Transcribed // EN">

② <!DOCTYPE PUBLIC "-//W3C //DTD XHTML 1.1/EN" > (HTML 4.0 (HTML 1.0, 2.0, 3.0) / (XHTML 1.1) 1.2 1.3

HTML + XML (XHTML)

③ <!Doctype html> (HTML5)

2014 → "responsive web" → by Google (mobile web) → websites render a/c to screen-size

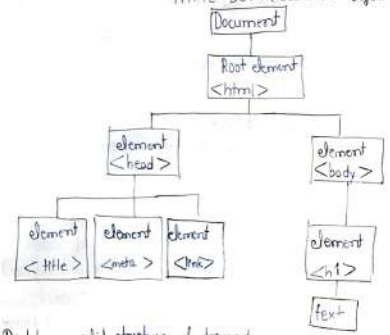
```
<!Doctype html>
<html>
  <head>
    :
  </head>
  <body>
    :
  </body>
</html>
```

<title> My first page </title> ← document title

<meta name=" " description=" "> → keyword → helps in search on web

<style>
  selector {
    property: value;
  }
</style>

HTML DOM (Document Object Model)



Doctype — valid structure of document

element → <tag>

attribute

"Object" → run-time → has attributes

Search → Search engine → "Java"

"Crawler" → searches <meta name=" "> of all web-pages

<style> } CSS </style>

Syntax: selector { property : value; } → ③ "Internal CSS"

element attribute value of attribute

① <style>
element selector → p {
property → color: red; ← value
text-align: center;
}
</style>

② <style>
id selector → #p {
}
</style>

③ <style>
class selector → .centre {
}
</style>

④ <style>
universal selector with star → * {
}
</style>

⑤ <style>
grouped elements (grouping selector) → h1, p, h2 {
}
</style>

⑥ <style>
class selector → p.centre {
}
</style>

```
<body>
  <h1> First Page </h1>
  <p style=" color:red; text-align:center" > Hello World </p>
</body>
```

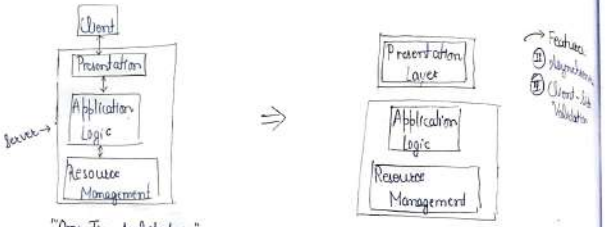
"style" as an "attribute"

① "Inline CSS"

Priority: ① inline ② internal ③ external

```
<head>
  <link rel="stylesheet" href="mystyle.css">
</head>
```

② "External CSS"



(i) API (Application Program Interface)

↳ means/way to interact b/w software components

- mode of interaction: request-response
- medium: API → http

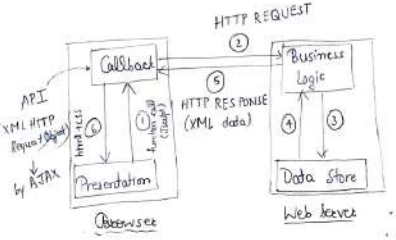
Interaction → synchronous
→ asynchronous
- notification-based interaction (function call)

(i) Javascript → synchronous (client waits for response after request and server waits for request after response)

(ii) AJAX → asynchronous (client and server are freed by introducing a buffer)

Javascript — object-based language
highly interpreted

Java — object-oriented language



- XAMP → by php
- APACHE TOMCAT → by JRE

software components that implement web server

```
<!DOCTYPE html>
<html>
  <body>
    <script> document.write("Hello World"); </script>
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>
  <p id="p1"> Hello </p>
  <input type="button" value="click" onclick="fun1()" />
  <script>
    function fun1() {
      document.getElementById("p1");
      innerHTML = "Hello World";
    }
  </script>
</body>
</html>
```

can be in <head>...</head> also

```
<!DOCTYPE html>
<html>
<body>
  <p id="p1">Hello</p>
  <input type="button" value="click" onclick="loadDoc()" />
  <script>
    function loadDoc(){
      x=new XMLHttpRequest();
      x.open("GET","demo.php");
      x.send();
      x.onload = function(){
      document.getElementById("p1")}
  </script>
</body>
</html>
```

JS Event/DOM Event

→ http request

→ http response

innerHTML = this response text

i) · GET → to get data from server

ii) · POST/ · PUT → to send data to server

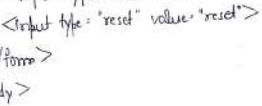
Data Types in Javascript

i) String
ii) Number (floating by default)
iii) Boolean
iv) BigInt
v) Null
vi) Undefined
vii) Object
viii) Symbol

"Viewport" → HTML5 feature

```
<meta name="">
```

for "responsive web" (screen rendering a/c to size)



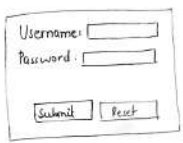

```
var x = Hello;
x = 5;
x = new Array();
```

variable can be redefined either data type

run-time memory (heap)
· object creation

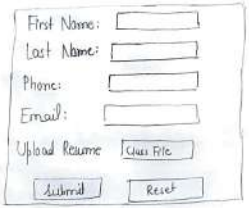
```
let x = Hello;
x = Abhi;
x = 5; ← NOT allowed
```

variable type can be changed




```
<!DOCTYPE html>
<html>
  <head>
    <title>Login Page</title>
  </head>
  <body>
    <form action="/usr" submit="Order.php">
      <label for="text">Username: </label>
      <input type="text" id="U" maxlength="10" required />
      <br><br>
      <label for="text">Password: </label>
      <input type="password" id="pwd" pattern="[a-z,A-Z,0-9]{16}" required />
      <br><br>
      <input type="submit" value="submit">
      <input type="reset" value="reset">
    </form>
  </body>
</html>
```

"Client-Side Validation" → Dynamic Web (happened at server in static web)




```
<!DOCTYPE Html>
<html>
 <head>
   <title>Registration Form </title>
 </head>
 <body>
   <form action="/" submit="order.php" autocomplete="on">
     <label for="text">First Name: </label>
     <input type="text" maxlength="10" required />
     <br><br>
     <label for="text">Last Name: </label>
     <input type="text" maxlength="10" required />
     <br><br>
     <label for="tel">Phone: </label>
     <input type="tel" pattern="[0-9]{4}-[0-8]{2}-[0-9]{7}" >
     <br><br>
     <label for="email">Email: </label>
     <input type="email" autocomplete="off" required />
     <label for="file">Upload Resume </label>
     <input type="file" multiple required />
     <br><br>
     <input type="submit" value="submit">
     <input type="reset" value="reset">
   </form>
 </body>
</html>
```

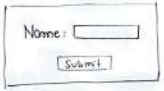
country code — area code — number

Q:) Using Javascript, write the code so that the content changes on clicking on Hello World

Event → listener (listens to the event) → handler (performs the actions of the event)

```
<!DOCTYPE html>
<html>
  <body>
    <p id="p1" onclick="fun()"> Hello World </p>
    <script>
      function fun(id){
          document.getElementById("p1");
          id.innerHTML = "DOM EVENTS";
        }
    <p onclick="func(this)"> Hello World </p>
  </body>
</html>
```

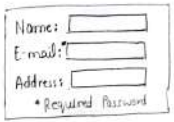
Q:) Write the code so when submit is clicked the entered text converts to uppercase




```
<!DOCTYPE html>
<html>
  <head>
    <script>
      function change (){
          x = go document.getElementById("t1");
          x.value = x.value.toUpperCase();
        }
    </script>
  </head>
  <body>
    <form>
        <label for="text">Name: </label>
        <input type="text" id="t1" required /> <br>
        <input type="button" value="submit" onclick="change()">
</body></html></form>
```

Q) Write the code so as when the checkbox is checked the password should become visible, not visible otherwise

```
<!DOCTYPE html>
<html>
<head>
  <script>
    function toggle(){
      x = getElementById("pwd");
      x.type = "text";
      if (x.type == "text"){
        x.type = "password";}
      else if (x.type = "password"){
        x.type = "text";
      }
  </script>
</head>
<body>
  <form action=" " method="POST">
    <label for="name">Username: </label>
    <input type="text" id="name"> <br>
    <label for="pwd">Password: </label>
    <input type="password" id="pwd"> <br>
    <input type="checkbox" id="show" onclick="toggle()">
    <label for="show">Show password </label><br>
    <input type="submit" value="submit">
  </form>
</body>
</html>
```



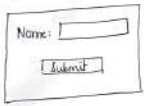
Q) Write the code, so the email field is required and has, border and background, while the other fields are not required and have green border and background (* → also red)

```
<!DOCTYPE html>
<html>
<head>
  <style>
    #email {
      background-color: red;
      border: 1px solid red;
    }
    #name, #address {
      background-color: green;
      border: 1 px solid green;
    }
  </style>
</head>
<body>
  <form action=" " method="POST">
    <label for="name">Name: </label>
    <input type="text" id="name"> <br>
    <label for="email">E-mail: </label><span id="dot">* </span> </label>
    <input type="email" id="email"> <br>
    <label for="address">Address: </label>
    <input type="text" id="address">
    <p> * Required Password </p>
  </form>
</body>
</html>
```

: → pseudo-class selector in CSS

↓

"defining a particular state of element"



Q.) If the form is submitted without any value for name, a pop-up should be displayed.

```
<!DOCTYPE html>
<html>
<head>
    <script>
        function check(){
            x= getElementById("name");
            if (x.value == NULL) {
                alert("Name is required");
                x.focus();
                return false;
            }

    </script>
</head>
<body>
    <form action="" method="POST">
    <label for="name">Name: </label>
    <input type="text" id="name"><br>
    <input type="button" value="submit" onclick="check()">
    </form>
</body>
</html>
```

alert() ⇒ window.alert()
(by default)

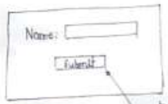
Q.) As soon as the website is opened, a "Welcome" alert should come.

```
<!DOCTYPE html>
<html>
<head>
    <script>function func(){
            alert("Welcome"); }
    </script>
</head>
<body onload="func()">
</body>
</html>
```

Q.) If name <6, alert an error, name >8, alert another error.

```
<!DOCTYPE html>
<html>
<head>
  <script>
    function func(elm, min, max){
        x= elm.value;
        if (x.length >= min && x.length <= max){
            return true;
        }else{
            alert("Please enter between "+max+" and "+min+" characters"); elm.focus();
            return false;
        }
  </script>
<body>
    <form>
        <label>Name </label>
        <input type="text" id="1" /><br>
        <input type="submit" value="submit" onclick="func(document.getElementById('1'),'6','8')" />
    </form>
</body>
</html>
```

<u>Sending Request to Server (Asynchronous)</u>



Q.) It is to display an error if anything but an alphabet is to be written in the name field

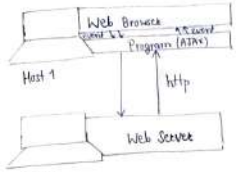
```
<!DOCTYPE html>
<html>
<head>
  <script>
    function check(){
    x = document.getElementbyId("name");
      isup = "/[A-Z]/".test(x);
      islow = "/[a-z]/".test(x);
      if (isup && islow){
        alert("Wrong Input");
        x.focus();
        return false;
      } else { return true;}
    }
  </script>
<body>
  <form action="">
    <label for="name">Name </label>
    <input type="text" id="name">
    <br>
    <input type="submit" value="submit" onclick="check()">
  </form>
</body>
</html>
```

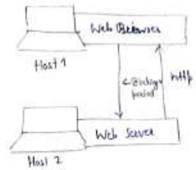
event handler

```
function check(){
  x = document.getElementbyId("name");
  y = /^[A-Z a-z]+$/;
  if (x.value.match(y)) {
      return true;
  }else{
    alert("Wrong Input");
    x.focus();
    return false
  }
}
```

number = /^[0-9]+$/

alphanumeric = /^[A-Z a-z 0-9]+$/

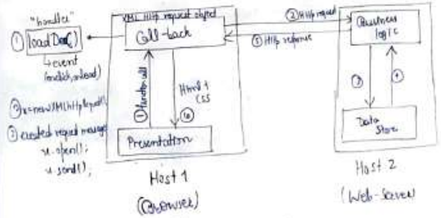




<u>Asynchronous</u>
"Non-blocking"
↓
notification based system

<u>Synchronous</u>
(There is a blocking period once the browser has made a request in which it has to wait for a response before making other requests)
"Blocking"

AJAX → Asynchronous <u>Javascript</u> and XML
- Javascript → to generate events
- XML → for storing and transport of data → XML, JSON




```
x.open( GET/POST, server URL, async );
```

- GET/POST, server URL → mandatory; GET/POST (request method)
- async → optional: '1' (default) or '0'
- x.open → to create http request

Client Request:

```
1 → function loadDoc(){
2       x = new XMLHttpRequest();
        x.open("GET", "order.php");
        x.send();
```

Server Response
- OnReadyStateChange
- Ready State
- Status
- Response Text

GET /example.htm HTTP 1.0 → Request line
User Agent: Mozilla/5.0
Host: example.com
Accept: text/html
Body (empty since nothing is being sent)

Http Request message (for GET) → Request Header

HTTP 1.1 200 → OK → Status line
Date: Sat, 17 Feb 2024 10:13 GMT
Server: Apache/2.4.4.6
Content-type: text/html
Content-length: 1024 ← size of data in bytes
→ Response header

```
<!DOCTYPE html>
<html>
<head>
<title>First Page </title>
</head>
<body>
  <h1>Hello World </h1>
</body>
</html>
```
→ Body

```
<script>
  function loadDoc () {
①←  x = new XMLHttpRequest ();          → XMLhttp object
②←  x.open (GET, "/index.php");        → create http request
     x.send ();   → send has no arguments as we are using "GET"
     x.onreadystatechange= function (){
       if (this.readyState == 4 && this.status = "200")
                  (this → refers to 'x')
       {
③←  document.getElementById ("D1").
          innerHTML = this.responseText;
                        (x.responseText)
       }
     }
</script>
```
② Http request
③ Http response

AJAX

callback → function

loadDoc() [previously loadXMLDoc ()]

http → "R-R" interaction
  x. Request ()
  x. Response ()

Onreadystate Change → function
→ ready State

0: Unopened → open() is not called
1: Opened → open() is called
2: HEADER_RECEIVED → send () is called
3: LOADING → download → x.responseText(); → to store data as text (coming from server)
                    → x.responseXML(); → to store data as XML
4: DONE → complete operation

Status →
- 100
- 200 (DONE)
- 300 (redirection)
- 400 (error at client side)
- 500 (error at server side)

404 → Page Not Found
403 → Forbidden

```
<script>
  function loadDoc (){
Request [
    x= new. XMLhttpRequest ();
    x.open (POST, /index.php), x.setHeader ("Content-Type", "application/x-www-form-urlencoded")
    x.send ("fname = Kumar && lname = Abhishek");
              (key)   (value)   (key)   (value)
Response [
    x.onreadystatechange = function (){
      if (this.readyState== 4 && this.status = "200"){
        document.getElementById ("p1").innerHTML = this.responseText;
      }                          (HTML element)          (Server data)
  }
</script>
```
(x.setHeader → will be added in the Request Header; "application/x-www-form-urlencoded" → HTML form data)

POST / index.php HTTP 1.0
User Agent: Mozilla/5.0
Host: example.com
Accept: text/html
Content type: form-url
Content length: …

URL: https://www.example.com/example.html

- http
- DNS
- IP

Http Request Msg (Generated by OS)

Request line	GET /example.html HTTP 1.1
Request header	User-Agent: Mozilla/5.0 Host: Abc.com Accept: text/html
Request Body	

Http Response Msg

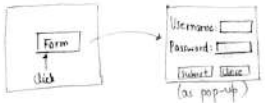
	Status code	Status Phrase
Status line	HTTP 1.1	200 OK

Response Header:
Date: 11th Feb 2025 11:48:56 GMT
Server: Apache/2.4.46
Content-type: text/html
Content-length: 1046

Response Body:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <h1>Hello world</h1>
  </body>
</html>
```

Q)




```
<!DOCTYPE html>
<html>
<head>
  <link href="style.css" rel="stylesheet">
</head>
<body>
  <div class="container">
    <button> Form </button>
  </div>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
  <link href="style.css" rel="stylesheet">
</head>
<body>
  <div id="root">
    <form action="submit.php" method="POST" id="myform">
      <label for="name"> Username: </label>
      <input type="text" id="name" required>
      <br><br>
      <label for="pass"> Password: </label>
      <input type="password" id="pass" required>
      <br><br>
      <input type="submit" value="Submit">
      <input type="button" value="Close" onclick="window.close()">
    </form>
  </div>
</body>
</html>
```

```
<script>
  function openform(){
    document.getElementById("myform").
      style.display="block";
  }
  function closeform(){
    document.getElementById("myform").style.
      display="none";
  }
</script>
```

```
style.css
#myform{
  max-width:200px;
  padding:10px;
  background-color:yellow;
}
#cent{
  display:none;
  position:fixed;
  bottom:0;
  right:20px;
  border:3px solid red;
  z-index:99;
}
```

Object and Data

JS Object → String ← data is shared b/w client and server as strings

String → JSON "Javascript Object Notation" ← to read strings as JS Object

JSON → Mongo DB

Graphics using HTML and Javascript

<!DOCTYPE html>

<html>

<body>

<canvas id="c1" height="300" width="400" style="border 1px solid black">

</canvas>

<script>

x = document.getElementById("C1"); — Step ① → Finding Canvas element

cx = x.getContext("2d"); — Step ② → creating drawing variable

① { cx.moveTo (0,0); → starting point
cx.lineTo (300,200); → ending point }

cx.stroke (); → drawing the line

</script>

</body>

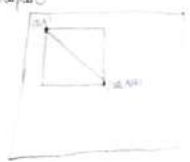
</html>

(XML)

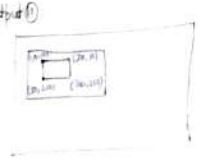
↳ <canvas>, <svg>

(HTML5) (scalable vector graphics)

Output ①



Output ②

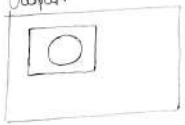


- moveTo (x,y);
- lineTo (x,y);
- stroke ();
- rect (x, y, height, width);
- beginPath ();
- arc (x, y, r, start angle, end angle, counter-clockwise);
  - x, y: arc centre; r: radius
  - counter-clockwise (optional): True / False (default)
- strokeText ();
- fillText ();

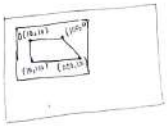
(III)

```
<script>
    x = document.getElementById("C1");
    cx = x.getContext("2d");
    cx.beginPath ();
    cx.arc (95, 50, 40, 0, 2* Math.PI);
    cx.stroke();
</script>
```

Output:

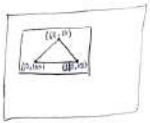


(IV)



```
<script>
    x = document.getElementById("C1");
    cx = x.getContext("2d");
① { cx.moveTo (10,10);
    cx.lineTo (10,100);
    cx.stroke ();
    cx.moveTo (10,10);
    cx.lineTo (200,100);
    cx.stroke ();
    cx.moveTo (10,10);
    cx.lineTo (200,10);
    cx.stroke ();
```

(v)



```
<script>
    c = document.getElementById("C1");
    cx = c.getContext("2d");
    cx.moveTo(60, 10);
    cx.lineTo(110, 100);
    cx.lineTo(10, 100);
    cx.lineTo(60, 10);
    cx.stroke();
</script>
```

(vi)


```
<script>
    c = document.getElementById("C1");
    cx = c.getContext("2d");
    cx.font = "50px Arial";
    cx.strokeText("Hello World", 50, 100);   → text with no color
                (2)
    cx.fillText("Hello World", 50, 100);   → text with black color
</script>
```

Addressing (Unit-I)

- Software Address
  - Port no.
  - IP address
- Hardware Address
  - MAC (Medium Access Control)

(i) IP address — talks about identifying a host on a N/w (unique for each host on N/w)

(ii) Domain Name

- A host can have multiple IP addresses if it is connected to multiple networks (unique for each)

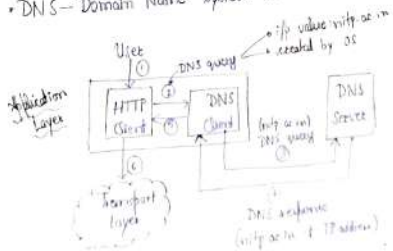
Application Layer	← HTTP, DNS, SMTP, FTP, DHCP, TFTP
Transport Layer	← Port no.
Internet / Network Layer	← IP address
Host-to-Network Layer (Data link + Physical Layer)	

TCP/IP (ARPANET) model

URL: https://www.nitp.ac.in

- https → HTTP (Port no.: 80) → called by OS
- www.nitp.ac.in → Domain Name (Host Name)

- DNS — Domain Name System (Port no.: 53)

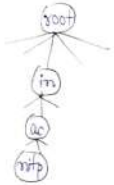


DNS

- Resolver (DNS client)
- Server

DNS → generic domain (.com, .edu, .gov, ...)
→ country domain (.us, .in, ...)
→ inverse domain
  ↳ i/p: IP address
  ↳ o/p: domain name

URL: https://www.rdtp.ac.in
- www: host
- rdtp: generic domain
- ac: country domain
- in: root domain



"domain tree"

Identification of Host on Internet
↳ IP Address

IPv4	IPv6
(32-bit)	(128-bit)
(Classful)	(Classless)

↳ ① Two-level hierarchy
② Three-level hierarchy
} classfull → 5 classes — 'A', 'B', 'C' (general use), 'D', 'E' (reserved) → Address space

'D': (multi-cast) group communication

IPv4 → 2^{32}
IPv6 → 2^{128}
} Address space

Max. net: 2^{24} (for class 'A')

Address space	(0-255) Host id	Host id	Host id	Host id
2^{31} ← A	0 (0-127)			
2^{30} ← B	Net id: 10 (128-191)	Net id	Host id	Host id
2^{29} ← C	Net id: 110 (192-223)	Net id	Net id	Host id
2^{28} ← D	1110 (224-239)			
2^{28} ← E	1111 (240-255)			

- Bits are reserved to identify the classes

- 0 → no value
- 127 → loop back address
- 255 → broadcast

} Addresses NOT assigned to any host on a n/w

IPv4 Address:
17.121.34.10 → class 'A'

① Two-level Hierarchy
- Network Address (Net id)
- Host Address (Host id)

Max. no. of host:
A – 2^{24}
B – 2^{16}
C – 2^{8}

Net id | Host id
201.24.67.32 → class 'C'
Net id: 201.24.67.0

- Router performs AND operation with default mask for that class

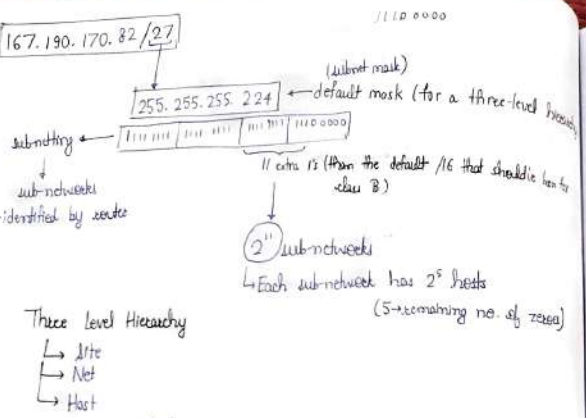
201.24.67.32
AND 255.255.255.0 ← default mask for class 'C'
201.24.67.0

Default Masks:
A – 255.0.0.0
B – 255.255.0.0
C – 255.255.255.0

130.11.252.234/16 (11111111.11111111.0.0)
255.255.0.0 ← default mask

↳ CIDR (Classless Inter Domain Routing)
/8 /16 /24 (for two level hierarchy)

(ii) 167.190.170.82/27



Q.) What'll be the subnet default mask of a class B n/w divided into 4 sub-networks?

$4 \to 2^2$ — here, $4 \to 2^2$

∴ 2 extra 1's than /16 — if $5 \to 2^3$ (nearest higher power)

255.255.1100 0000.0000 0000

255.255.192.0

Q.) A Network is divided into 4 sub-networks. One of the address in subnetwork is 141.14.120.77. What is the sub-network address?

Subnets = $4 \to 2^2$

Subnet Mask: 255.255.192.0

141.14.120.77

AND 255.255.192.0

141.14.128.0 ← Subnet Address

120 → 01 11 1000

192 → 1100 0000

0100 0000

167.199.170.82/27

(i) Find the no. of addresses in the N/w

(ii) Find the first address.

Subnet Mask: 255.255.255.224

∴ 11 extra 1's are there

∴ There are 2^{11} sub-networks and each has 2^5 hosts.

So, 2^5 = 32 addresses are possible for this sub-network

224 = 1110 0000

82 = 0101 0010

0100 0000

Subnet First Address (and First): 167.199.170.82

AND 255.255.255.224

167.199.170.64.0

Q.) Find the subnet mask in the following cases:

i) 1024 subnets in class A → 255.11111111.11000000.00000000 = 255.255.192.0

ii) 256 subnets in class B → 255.255.11111111.0 = 255.255.255.0

iii) 32 subnets in class C → 255.255.255.11111000 = 255.255.255.248

Q.) Your network IP is 192.168.1.0. Is it possible to create 30 subnetwork with 4 host per network?

192.168.1.0 → class C

255.255.255.0

30 subnetworks is closest to 2^5 i.e. 5 (1) and 4 host → 2 zeroes (2^2) → 2 digits

Hence, ~~No~~ Yes — 192.168.1.0/24 ← CIDR not — Subnet Mask: 255.255.255.248

Q.) An organisation has been granted IP: 211.17.180.0/24. The admin wants to create 30 subnets.

(i) Find the subnetwork

(ii) Find the no. of addresses in each subnet

(iii) Find first and last address of first subnetwork

(iv) " " " " last "

IP: 211.17.180.0/24

(i) Subnet Mask: 255.255.255.0

(ii) Each subnet can have 2^8 = 256 addresses

(iii) 211.17.180.0

(iv) 211.17.180.0

IP: 211.17.180.0/24 +30 subnetworks

(i) Subnet Mask: 255.255.255.1111 1000
: 255.255.255.248

(ii) Each subnet has $2^3 = 8$ address

(iii) First subnet:

1st address: 211.17.18.0
AND 255.255.255.248
211.17.18.0

last address: 211.17.18.7

Range: 211.17.18.0 — 211.17.18.7
Out of which .0 and .7 are NOT assigned so only .1 — .6 are available for users

(iv) last subnet

Range: ~~211.17.18.248~~ — ~~211.17.18.255~~

1st Subnet: 211.17.18.0 — 211.17.18.7
2nd Subnet: 211.17.18.8 — 211.17.18.15
3rd Subnet: 211.17.18.16 — 211.17.18.23
4th Subnet: 211.17.18.24 — 211.17.18.31
last (5th) Subnet: 211.17.18.~~248~~ 32 — 211.17.18.~~255~~ 39

Q.) 192.168.1.0

Subnet Mask: 255.255.255.240
1111 0000

Range

Subnet 1: 192.168.1.0 — 192.168.1.15
Subnet 2: 192.168.1.16 — 192.168.1.31
Subnet 3: 192.168.1.32 — 192.168.1.47

IP Address

"Software Address" → Network/Internet layer → "Host"

↳ ① IPv4 (IP address)
- ↳ net, host
- ↳ net, subnet, host

Web Server ← http → "Application" (Services)
Name Server ← DNS →

② Port Number (0–65535) → Transport layer
"Logical address" (for application running on host)
↳ read by TCP/UDP

- well-known (0–1023) → for applications/services (protocols)
- registered port (1024–49151) → organisations (by IANA)
- ephemeral port (49152–65535) → "lab" (temporary) (private)

"Hardware Address" → Data Link Layer
↳ MAC (Medium Access Control)

"Socket" → IP address + Port No.

Cookies

↳ 4 KB max.

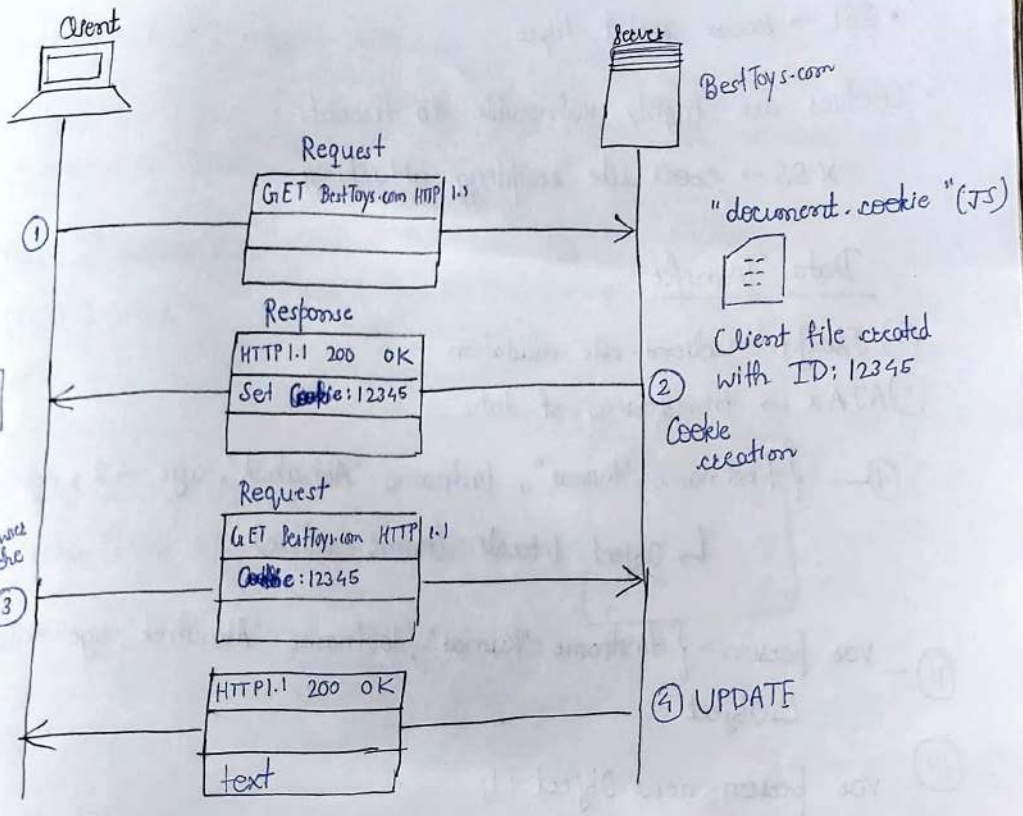
(key : value) → ".txt file"

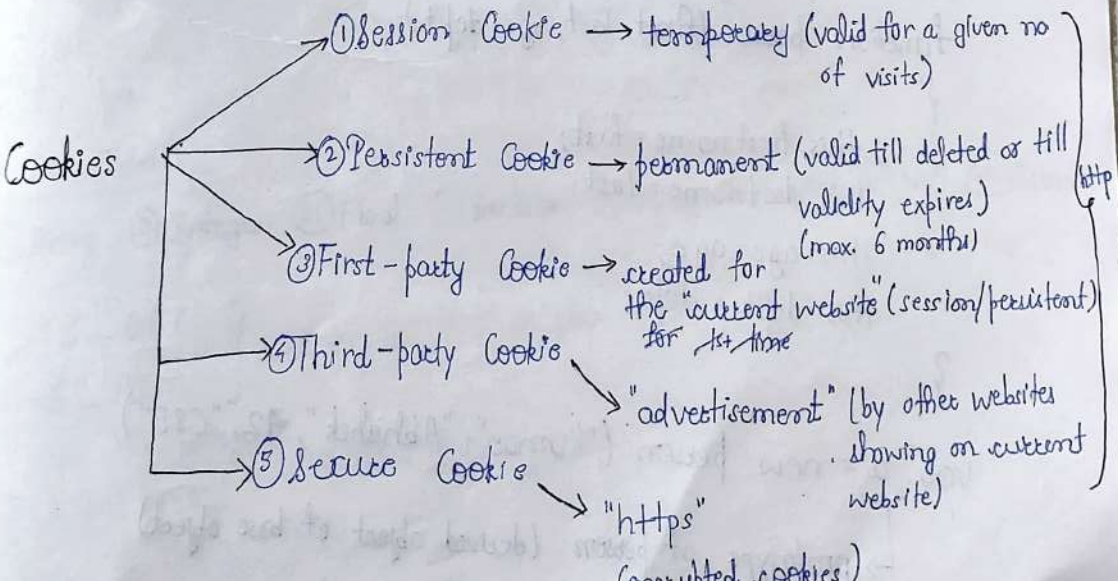
"HTTP" —disadvantage→ stateless — the server doesn't remember previous requests made by the user

"Session" → web
↳ "user-tracking" by "web server" with the help of cookies

- ~~always~~ created at server side and gets stored in the client (for user-tracking)
- can also be created by the browser (document.cookie in JS)

"Session-tracking" ←





Parameters of Cookie:

① Name → key (mandatory)

② Value → 12345 (in above case)

③ Expiration Date → expiry (session by default)

④ Valid Path → URL (https://www.BestToys.com/index.html)

⑤ Valid Domain → "BestToys.com"

⑥ Secured Connection → SSL (https)

- SSL → secure socket layer
- Cookies are highly vulnerable to threats.
  XSS → cross-site scripting attacking

Data Transfer

① JScript → client-side validation
② AJAX → transferring of data

(i) — { firstname: "Kumar", lastname: "Abhishek", age: 42, dept: "CSE" }
↳ Object Literals (name–value)

(ii) — var person = { firstname: "Kumar", lastname: "Abhishek", age: 42, dept: "CSE" }
↳ Object

(iii)
```
var person = new Object();
person = "--";

function person(first, last, age, dept)
{
    this.firstname = first;
    this.lastname = last;
    this.age = age;
    this.dept = dept;
}

var a = new person("Kumar", "Abhishek", 42, "CSE");
```
↳ prototype of person (derived object of base object)
(known as inheritance in C++, Java)

JSON → Java
↳ convert
receiv

Hypertext Pre

```
<!DOCTYPE
<html>
<body>
<?php
echo
?>
</body>
</html>
```

$x
$txt
$y

Data Ty

① String

$_
$_

JSON → Javascript Object Notation

↳ converts object to string for sending data and vice-versa while receiving data

Hypertext Preprocessor (PHP)

Personal Home Page (First abbreviation)
"open-source"
"case-sensitive"

```
<!DOCTYPE html>
<html>
<body>
  <?php
    echo "Hello World"; (ECHO/ëcHO)
  ?>
</body>
</html>
```

php syntax :

```
<?php

?>
```

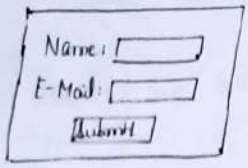
$x → defining a variable 'x' in php
$txt
$y

Data Types

(I) String (II) Integer (III) Float (IV) Boolean (V) Array (VI) Object (VII) Null (VIII) Resource

$_GET
$_POST
} used for GET/POST in php

Q)



when the submit is clicked, it should give a welcome msg.

```
<!DOCTYPE html>
<html>
<body>
  <form action="welcome.php" method="POST">
  <label for="name">Name:</label>
  <input type="text" id="name" name="N1">
  <br><br>
  <label for="email">E-Mail:</label>
  <input type="email" id="email" name="I1">
  <br><br>
  <input type="submit" value="submit">
  </form>
</body>
</html>
```

for post
URL: http://localHost/welcome.php

for get
URL: http://localHost/welcome.php?
"N1⇒abc; I1⇒xyz"

welcome.php → on server

```
<!DOCTYPE html>
<html>
<body>
  Welcome <?php echo $_POST["N1"]; ?> <br/>
  Your Mail Id <?php echo $_POST["I1"]; ?> <br/>
</body>
</html>
```

* POST sends the data in the form of an array [key1⇒value1, key2⇒value2]
(GET also creates the array)

[N1⇒xyz, I1⇒abc]

Key / value → input data held by form elements
↓
form controls (here N1, I1)

setcook

<?ph
$
$
se
?>

(at server) creating cookie → should come before <!DOCTYPE html> and <html>

<body>
<?php
if (!

} else

}
?>

$_GE
$_POS
$_cook
$_SES

setcookie (name, value, expire, path, domain, secure);

name — mandatory

value, expire, path, domain, secure — optional

```php
<?php
    $ C_name = "user";
    $ C_value = "Abhishek";
    setcookie ( $C_name , $C_value, time() + 86400X30, "/");
?>
```

- $C_name → key
- $C_value → value
- time() + 86400X30 → expiry
- time() − any value (true) → way to delete cookie (past expiry)

(at server) creating cookie

```
<!DOCTYPE html>
<html>
```

```
<body>
<?php
    if (! isset ($_cookie([C_name])) {
        echo "cookie name" $ C_name "not set"
    }else{
        echo "cookie" $ C_name "is set*<br/>";
        echo "cookie value" $ C_value;
    }
?>
```

retrieving the cookie (at server)

Scope of variable in PHP

1. local
2. global
3. static

global php variables:

- $_GET
- $_POST
- $_COOKIE
- $_SESSION

```php
<?php
function test(){
    $x=0; (local variable)
    echo $x;
    $x++;
}
test(); → x=0
test(); → x=0
test(); → x=0
?>
```

```php
<?php
function test(){
    static $x=0; (static variable)
    echo $x;
    $x++;
}
test(); → x=0
test(); → x=1
test(); → x=2
?>
```

state
↓
userdata → "session" (login to logout)
↓
webapp

```php
<?php
print_r ($_SESSION);
?>
```

DBMS (RDBMS) → "Tables" ↔ Rows / Tuples

DBMS (RDBMS) → DDL – Data Definition language
- array
- primitive type

DBMS (RDBMS) → DML – Data Manipulation language
- create
- update
- delete
- read

demo.php

```php
<?php
session_s
?>
<!DOCTYPE
<html>
<head> </
<body>
<?php
setting a session variable
$_
$_
echo
?>
</body>
</html>
```

logout.php

```php
<?php
session
session
session
?>
```

demo.php

```
<?php
    session_start();
?>
<!DOCTYPE html>
<html>
<head></head>
<body>
<?php
    $_SESSION["User1"] = "pen 1"      // setting a session variable
    $_SESSION["User2"] = "pen 2"
    echo $_SESSION["User 1"]    // print_r ($_SESSION)
?>
</body>
</html>
```

logout.php

```
<?php
    session_start();
    session_destroy();  → session destroy
    header("location demo.php");  → Redirect
?>
```

• XAMPP → web-server environment implemented by php

• php.ini → php configuration file

↳ "File Uploads" (default: off)

① >


```
<!DOCTYPE html>
<html>
  <body.><form action="upload.php" method="POST" enctype="multi-part/form-data">
    <label>Name: </label>
    <input type="text" id="n1" /> <br/>
    <label>E-mail: </label>
    <input type="email" id="e1" /><br/>
    <label> File </label>
    <input type="file" id="f1" /> <br/>
    <input type="submit" value="submit">
    </form>
  </body>
</html>
```

"content-type" in HTTP header

upload.php

```
<?php
  $target_dir = "upload";
  $target_file = $target_dir . basename ($_files["f1"]);
  $uploadOK=1;
  $imageFileType = strtolower (pathinfo ($target_file, PATHINFO_EXTENSION));
  if (isset ($_POST["Submit"]){
      $check = getimagesize ($_FILES["f1"]);
  if ($check !== false) {
      echo "File is an image" $check["mime"]
      $uploadOK=1 }else{
          echo "File Not image"
          $uploadOk=0 }
?>
```

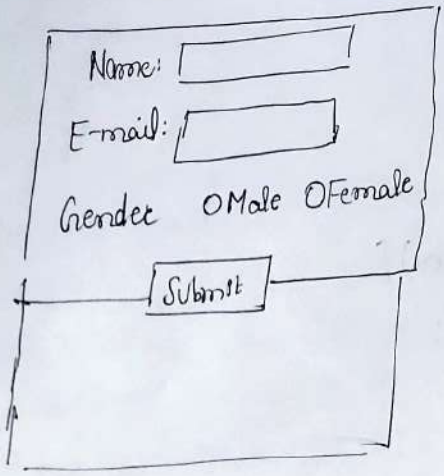
```
if file_exists [$target_file] {
    echo " File already exists";
    $uploadOK=0;

}
if ($FILES["f1"]["size"] >5000 000) {
    echo "File capacity/size oversized";
    $uploadOK=0;

}
if ("$imageFileType" = "jpg" && $imageFileType="jpeg" && $imageFileType="unset
 && imageFileType="png" && imageFileType="gif") {
    echo " File Type Not Supported";
    $uploadOK=0;

 }
?>
```

Q.)




```
<!DOCTYPE html>
<html>
<head></head>
<body>
  <form method="POST" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  <label>Name:</label>
  <input type="text">
  <label>E-mail</label>
  <br>
  Leaving this.
  <label>Gender</label>
  <input type="radio">Male <br>
  <input type="radio">Female <br>
  <input type="submit" value="submit">
</form>
```

* $_SERVER["PHP_SELF"] → loops back to same page

* if action is blank, then php part should be before <form>.

~~Diff b/w ...~~

```
<?php
if ($name ... $name: $ima: $greater = " ";
  if ($_SERVER["REQUEST METHOD"] == POST){
       $name = $_POST["n1"];
       $ima = $_POST("i1"];
       $gender = $_POST["g1"];
  }
  echo "<b> Your Input </b>";
       echo $name;
       echo "<br>"
       echo $email;
       echo "<br>";
       echo $gender;

       ?>
}
```

* htmlspecialchars () → converts the input fields to HTML entities (&lt → <, &gt → >)
* $_SERVER["PHP_SELF"] → loops back to the same page

demo.js

```
var http = require('http');
http.createServer(function(req,res){
    req.write("Hello World");
    req.end();
}).listen(8080);
```

→ module
→ server object
→ res → response
req → request

Node.JS
↓
"NPM"
(faster than php)

↳ "Hello world" is considered text

```
var http = require('http');
http.createServer(function(req,res){
    res.writeHead(200, {"content-type": "text/html"});
    res.write("Hello World");
    res.end();
});
```

→ "Hello World" is considered part of .html file

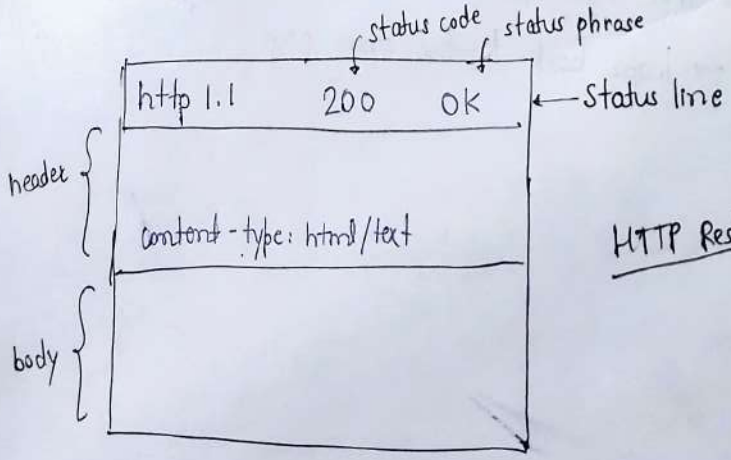
http://localhost/8080

then in browser

in CMD

path: node demo.js

C://User/Admin/nodejs/



HTTP Response

```
<!DOCTYPE html>
<html>
<body>
  <h1> First Page </h1>
  <p> Hello World </p>
</body>
</html>
```

index.html

```
var http = require('http');
var f    = require('fs');
http.createServer (function(req,res){
    f.readFile ("index.html", function (err, data){
        res.writeHead (200, {"content-type":"text/html"});
        res.write (data);
        return res.end ();
    });
}).listen(8080);
```

demo.js

FTP → 20, 21
↑ ↑
ftp pre-ftp

http → 80

http 1.1 200 OK
content-type: text/html
<html> <body> </body> </html>

Response msg

```
var http = require('http');                                    (*)
http.createServer (function(req, res){                         (**)
    res.writeHead (200, {"content-type": "text/html"});
    res.write ('<form action='fileupload' method='POST' enctype="multi-part/formdata">');
    res.write ('<input type="file" id='f1'>');
    res.write ('<input type="submit" value="Submit">');
    res.write ("</form>");
    return res.end();
}.listen(8080);
```

used to read file

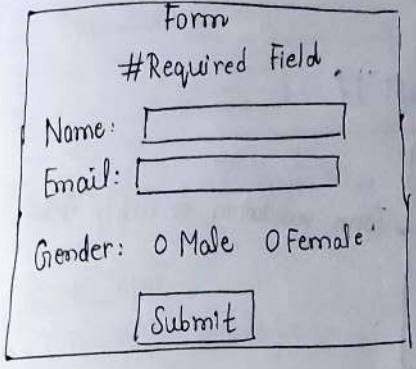
```
(*) → var x = require ('form-data');
      http.createServer (function(req, res){
(**) →  if (req.url == "\fileupload"){
          var x2 = new formidable.IncomingForm();
          x2.parse (req, formdata (err, fetchfile) {
          res.write ("File Uploaded");
          res.end();
      });
}).listen(8080);
```

Parse the file.

"JSON" database → MongoDB

- taking object from JS and translated to ~~script~~ string
- taking string from server and translated to object
- stores data as "key : value" pairs.

FORM VALIDATION – PHP



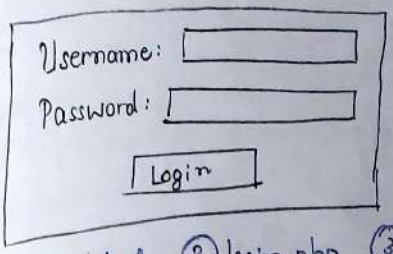

```
<!DOCTYPE html>
<html>
  <head>
  <style>
    .empty{color:red}
  </style>
  </head>
  <body>
    <form method="post" action="<?php echo htmlspecialchars("/$_SERVER[PHP_SELF]
                                                              ?>>
    Name: <input type="text" id="N1">
    <span class="error"><?php echo $nameerr ?></span><br><br>
    Email: <input type="email" id="e1">
    <span class="error"><?php echo $emailerr ?></span><br><br>
    <?php    Gender: <input type="radio" id="g1">Male
      $name=F        <input type="radio" id="g1">Female ##
    <span class="error"><?php echo $genderserr ?></span><br>
    <input type="submit value="submit" id="submit">
  </form>
```

**

```
<?php
    $name = $email = $gender = "";
    $nameerr = $emailerr = $gendererr = "";
    if ( $_SERVER ["REQUEST_METHOD"] = "POST"){
        if (empty( $_POST ["N1"]){
            $nameerr = "Value required ";
        } else{
            $name = $POST["N1"];
        }
        if (empty ($_POST ["e1"]){
            $emailerr = "Value required";
        }else{
            $email = $_POST["e1"];
        }
        if (empty ($_POST["g1"]){
            $gendererr = "Grender can't be empty";
        }else{
            $gender = $_POST ["g1"];

        }
        echo "<h1>Input Value</h1>";
        echo $name;
        echo "<br>"
        echo $gender;
        echo "<br>"
        echo $email;
        echo "<br>"
?>
</body>
</html>
```

Form validation for empty field.



① login.html ② login.php ③ index.php ④ logout.php

login.html

```
<!DOCTYPE html>
<html>
<body>
  <form method="post" action="login.php">
    Username: <input type="text" id="N1"><br>
    Password: <input type="password" id="pwd"><br>
    <input type="submit" value="login" name="submit">
  </form>
</body>
</html>
```

login.php

```
<?php
  session_start();
  if(isset($_POST["submit"])){
    $loginarr = array('Abhi'=>'12345', 'Kumar'=>'67890', 'user123'=>'admin',
                      'administrator'=>'admin@123');
    $username = isset($_POST['N1'])? $_POST['N1']: "";
    $password = isset($_POST['pwd'])? $_POST['pwd']: "";
    if(isset($loginarr[$username]) && $loginarr[$username]==$password){
      $_SESSION['UserData']['User1'] = $loginarr[$username];
      header("location: index.php");
      exit; } else{ echo <span style="color:red">Invalid Credentials</span>;}
?>
```

index.php

```
<body>
<?php
    session_start();  → forwarding "user-key" from login.php
    if (isset ($_SESSION['User Data']['User'])){
        header ("location: login.php");
        exit;
    }
?>
<h1> Welcome </h1> <br>
<a href="logout.php" > Logout </a>
</body>
```

logout.php

```
<?php
    $server  Session_start();  → forwarding "user-key" from previous page
    Session_unset();  ——→ destroying "User 1"
    header ("location: login.php");
    exit;
?>
```

* Server doesn't hold cookie value.
* Session stored at server side → can be accessed by user-key (cookie)

```
<!DOCTYPE html>
<html>
  <head>
    <script src="jquery-name 3.7.1.js"></script>
    <script>
      document.ready (function (){
        $ (button). click (function (){
          $ ("p"). hide ();
        });
      });
    </script>
  </head>
  <body>
    <h1>jQuery </h1>
    <p> Hello World </p>
  </body>
</html>
```

jQuery library → CDN → by Google, Microsoft

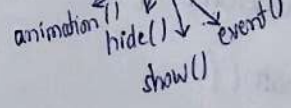
jQuery

Syntax: $ (selector). action ();

query/find html element

action to be performed on selected element

- $(this). hide ();
- $("p"). hide();
- $("#test"). hide();
- $(". test"). hide();




```
"event" ← document. ready (function (){
  ...
  ...
  ...
});
```

called when browser is opened/closed

```
<script>
  document.. ready (function (){
    alert ("Hello World");
  });
</script>
```

Javascript	JQuery
· Dynamic	· Dynamic and responsive
	· No. of lines of code is less

```
<script>
    document.ready (function (){
        $("p") on ({
          mouseenter , function (){
            $(this). CSS ("background-color":"red"};
          mouseleave, function () {
            $(this) . css ("background-color", "blue" );
          mousedown, ("background-color", "green") function(){
                $(this). css ("background-color", "green");
          }
</script>
<script>
    document. ready (function () {
          $ (       ). click (function () {
          $ ("p")
          });
</script>
<script>
    document. ready (function (){
        $(button). onclick (function ( ){
            $("div"). animate ({
                height += "250px",
                left  = "300px",
                width  +="250px"
            });
    });
</script>
```

```
<script>
  document.ready (function (){
        d = $("div");
  d.animate ({right : "2500px", left = "300px"});
  d animate ({ height : toggle});
  d animate ({left :"400px"
  d. animate ($(left), 400px);
</script>

<script>
  document
```

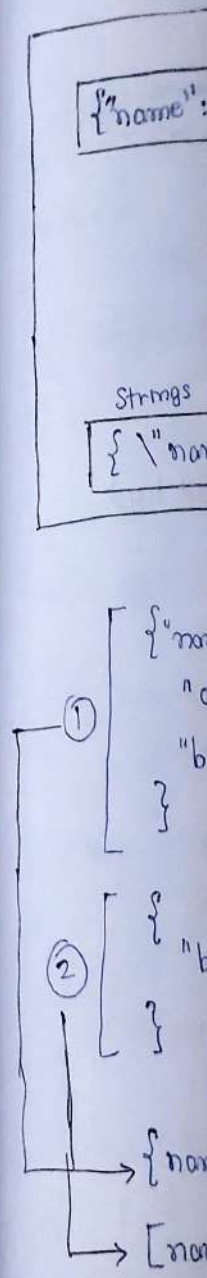
```
<script>
    document.ready(function #(){
        $(button).click(function(){
            $get("index.php", function(data, status){
                alert("Data" + data + "status" + status);
            });
        });
    });
</script>
```

mandatory URL, optional callback

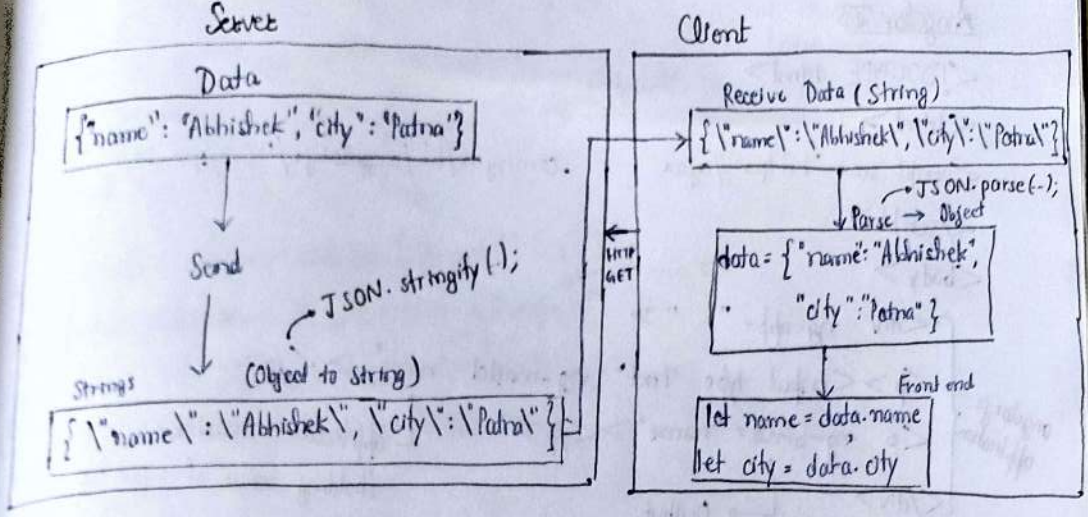
① x.send(); → HTTP GET

⑪ x.send(args); → HTTP POST

```
<script>
    document.ready(function(){
        $(button).click(function(){
            $post("index.php", {name: "Abhishek", city: "Patna"}, function(data, status){
                alert("Data" + data + "Status" + status);
            });
        });
    });
</script>
```



* JSO
* JSON
• In JS



① {"name": "Abhishek",
"age": 30,
"branch": "CSE"
}

② {"branch": ["CSE", "ECE", "EE", "ME"]
}

{name: value} → JSON → Object

[name: value] → Array

Data Types (supported in JSON)

- String
- Object
- Boolean
- Null
- Array
- Number

Undefined, function, date
↳ Not supported in JSON

- JSON. stringify (object) → string
- JSON. parse (string) → object

- In JS, variables can be undefined but objects can't.

Angular.JS

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>
    <div ng-app="">
    <p><input type="text" ng-model="name"/></p>
    <p ng-bind="name"></p>
    </div>
</body>
</html>
```

angularjs application

application variable (holding value of input tag)

Output:
`<p>name.value</p>`

Angular.js application ← ng-app
ng-model
ng-bind
} Angular.JS directives

ng-model → binding HTML control data to application data

ng-bind → binds application data to HTML view

```
<div ng-app="">
    <p>Expression: {{5+5}}</p>
</div>
```

{{5+5}} → 10

Expressions: {{ ... }}

```
<div ng-app="" ng-init="a=5;b=2">
    <p>Expression: {{ a+b }}</p>
</div>
```

ng-init="a=5;b=2" — application variables

(Angular.js application, module)

```
<div ng-app="myAPP"
     ng-controller="x">
    {{ fname + " " + lname }}
</div>
```

controller → controls the variables of the application (here fname, lname)

properties of object $scope

```
<script>
  var app = angular.module("myAPP", []);
  app.controller('x', function ($scope){
        $scope.fname = "Kumar";
        $scope.lname = "Abhishek";
  });
</script>
```

object

(M, V, C)

model → "data"

view → "front-end" (HTML)

controller → "function ()" (working on data)

Objects → Properties, Methods

- "highly local in nature"
- "properties of object can't be accessed outside of it, so better security"
  - data can be accessed through methods (public)
  - (private)

list.js

```
<script>
    var app = angular.module("myApp", []);
    app.controller("myCtrl", function ($scope){
        $scope.list = [
            { name : "Kumar", branch : "CSE" },
            { name: "Abi"   , branch: "ECE" },
            { name: "XYZ"   , branch: "EE" },
        ];
    });
</script>
```

Display the data in a list.

```
<body>
  <div ng-app="myApp" ng-controller="myCtrl">
    <ul>
      <li ng-object repeat="x in list">
          {{ x.name + ", " + x.branch }}
      </li>
    </ul>
  </div>
  <script src="list.js"></script>
</body>
```

- ng-repeat → loop

angular.js directive (ng) click
ng-focus
ng-blue
ng-dbclick
ng-keypress
ng-keydown
ng-mousemove

ng-mousedown
ng-mouseup

$name;

list.js

```
var app = angular.module("myAPP", []);          // model
app.controller("myCtrl", function($scope, $http){   // controller
    $http.get("welcome.html")
        .then(function(response){
            $scope.msg = response.data;
        });
});
```

(application data (property of $scope))

$http → XMLHttp request object (AJAX)

.get .post .put .jsonD .delete .data

Show data from file (welcome.html).

```
<body>
  <div ng-app="myAPP" ng-controller="myCtrl">
    <p>{{ msg }}</p>   ← binding msg (response.data) to <p> element
  </div>
  <script src="list.js"></script>
</body>
```

response
- .data
- .config
- .status
- .statusTxt
- .header

Show data from a student.php file.

```
<div ng-app="myApp" ng-controller="myCtrl">
  <table border="1">
    <tr ng-repeat="x in list | orderBy:'branch'">
      <td>{{x.name}}</td>
      <td>{{x.branch}}</td>
    </tr>
  </table>
</div>
<script> var app = angular.module("myApp", []);
  app.controller("myCtrl", function($scope, $http) {
    $http.get("student.php").then(function(response){
      $scope.list = response.data.records;
    }, function(response){
      $scope.list = "Dataset loaded";
    });
</script>
```

Q:) On clicking, it should count

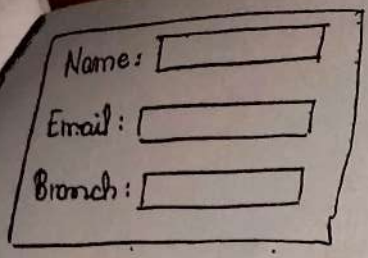
```
<div ng-app="myApp" ng-controller="myCtrl">
    <button ng-click="count=count+1"> Click </button>
    <p> {{count}} </p>

</div>
<script>
    var app= angular.module("myApp",[]);
    app.controller("myCtrl", function($scope){
        $scope.count=0;
    });
</script>
```

Q:) On clicking on button, a dropdown menu should appear/disappear

```
<div ng-app="myApp" ng-controller="myCtrl">
    <button ng-click="                    "> Click </button>
    <div ng-show="display">
        <h1> ... </h1>
        <p> ... </p>
        <p> ... </p>

    </div>
</div>
<script>
    var app = angular.module("myApp",[]);
    app.controller("myCtrl", function($scope){
        $scope.display = false;
        $scope.myfunc=function(){
        $scope.display = !$scope.display;
        }
    });
</script>
```




```
{
  name: "Abhishek",
  email: "abc@mail.com",
  branch: "CSE"
}
```

Bind the data to their respective fields:

```
<script>
    var app = angular.module("myApp", []);
    app.controller("myCtrl", function ($scope){
        $scope.user = {name: "Abhishek", email: "abc@mail.com", branch: "CSE"};
    });
</script>
<div ng-app="myApp" ng-controller="myCtrl">
    <form action="index.php">
        Name: <input type="text" ng-bind=ng-model="user.name"/> <br>
        Email: <input type="email" ng-model="user.email"/><br>
        Branch: <input type="text" ng-model="user.branch"/> <br>
        <input type="submit" value="Submit"/>
    </form>
</div>
```

<svg> → scalable vector graphics