Internet Programming

Unit I

Web Essentials

Web Essentials : Clients - Servers - Communication ; HTTP protocol : Request and Response Messages - Functionalities of Web Client and Web Server ; Web server : Vulnerabilities - Attacks and their prevention ; HTML5 : Table - List - Image - Form - Semantic elements - CSS3 : Types of style sheets - selectors - box model - rule cascading - inheritance transformations - Transitions - Animations

* **Application program** → a program designed to perform specific functions directly for the user

* **Web Application** → Application that is accessed via the web browser over a network like internet / intranet

* **Internet Programming** → Includes a broad variety of technologies spanning diverse areas such as

  - protocols for communication networks
  - interfaces to databases
  - programming of GUI

* Internet Protocol

Function - transfer data from source device to destination device

→ The IP source software creates a packet representing the data. It has:

Header - source and destination IP addresses, length of data

Data

→ If the destination is on another LAN, the packet is sent to a gateway that connects to more than one network.

* Limitations of IP

→ no guarantee of packet delivery - packets can be dropped

→ communication is one-way (source to destination)

* Transmission Control Protocol (TCP)

→ adds the concept of a connection on top of an IP

→ provides guarantee that packets are delivered

→ provides 2-way (full-duplex) communication

→ TCP also adds the concept of a port:

→ The TCP header contains a port number representing an application program on the destination computer

→ Some port nos. have standard meanings (eg: port 25 is for SMTP)

→ Other port numbers are available first-come first-serve to any application

* User Datagram Protocol (UDP)

→ Similar to TCP in that it builds on top of IP, and provides the concept of a port

→ Unlike TCP in the sense that there is no connection concept, and there is no transmission guarantee.

→ UDP is advantageous over TCP, since it is lightweight, so it is faster for one-time messages.

* Domain Name Service (DNS)

→ analogous to a phone book for the internet

→ maps host names and IP addresses

→ uses UDP for communication

→ Host names have labels separated by dots - the final label is a top-level domain eg. .com, .in, .org

* Analogy for Protocols

IP - telephone network

TCP - calling someone, talking to them, hanging up

UDP - calling someone and leaving a message

DNS - directory assistance

* Higher Level Protocols

- SMTP
- FTP
- HTTP

* World Wide Web

→ The WWW is the collection of machines (web servers) on the Internet that provide information, particularly HTML, via HTTP.

→ Machines that access information on the web are called webclients. A web browser is a software used by an end user to access the web.

* What the Server and Client do

Web Client (Web Browser)

→ Implemented over a TCP connection. The interaction is as follows:

(i) user enters web address in browser

(ii) browser uses DNS to locate IP address

(iii) browser opens TCP connection to server

(iv) browser sends HTTP request over connection

(v) server sends HTTP response to browser over connection

(vi) browser displays body of response in the client area of the window

Web Server

(i) receives HTTP request via TCP

(ii) maps host header to specific virtual host

(iii) map request URI to specific resource w/ the virtual ho

→ If its a file, return file in HTTP response

→ if it a program, run the program and return the output in the HTTP response

(iv) Map type of resource to appropriate MIME type and use to set Content-type header in HTTP response

(v) Log info about the request and response.

* URI, URN, URL

↓

uniform resource identifier

URN - Uniform Resource Name

→ can be used to identify resources with unique names, such as books (ISBNs are unique)

→ The scheme is urn

URL - Uniform Resource Locator

→ specifies location at which a resource can be found

→ some URL schemes include http, https, ftp, mailto & file

→ A URL usually has the following components:

(i) Protocol - technology that will be used to transfer the data, http or https

(ii) Domain - refers to domain name - google.com

(iii) Path - section and page on the site

(iv) Query - contains data that is being passed to the page.

* Client Caching

→ A cache is a local copy of information obtained from some other source.

→ most web browsers use cache to store requested resources so that the subsequent requests to the same resource will not necessarily require a HTTP request/response

eg. an icon appearing multiple times in a web page.

Cache Advantages

- much faster than HTTP request / response
- Less network traffic
- Less load on server

Cache Disadvantage

- cached copy of resource may be invalid - inconsistent with remote version

Validating cached resource

- Send HTTP HEAD request and check the Last modified or ETag header in the response.
- Compare current date/ time with the expire sent in the response.

* Web Application Security Vulnerabilities

* Authentication and Authorization

Authentication - verifying that a user is the person they say they are

Authorization - granting a user access to a specific resource or permission to perform a particular action.

* Authentication Vulnerabilities

→ issues that affect authentication processes and make website and applications susceptible to security attacks in which an attacker can masquerade as a legit user.

→ happens because of weak passwords or poor authentication design and implementation.

(A) Brute Force Attack

→ gain illegal access by entering large nos. of randomly / pre-generated combinations of usernames and passwords until they find the one that works

(B) Weak Login Credentials

→ Attackers look for accounts with easy-to-guess passwords. They try common credentials like 'admin', 'password'

(C) Username Enumeration

→ Attackers determine if usernames are valid or not. This lowers the cost for other attacks, like brute force attacks / weak credential attacks.

(D) HTTP Basic Authentication

→ sending a username and password w/ each request

→ If appropriate security protocols like TLS session encryption are not used for all communication, it is easy for attackers to steal the credentials.

(E) Poor Session Management

→ includes no session timeouts, exposure of session IDs in URL and poor session invalidation

F. Staying Logged In

→ A 'Remember me' or 'Keep me logged in' generates a cookie that lets one skip the process of logging in.

→ This can lead to a cookie-based authentication vulnerability if an attacker can predict a cookie or deduce its generation pattern

* SQL Injection

→ A code injection technique that can destroy databases.

→ SQL Injection is the placement of malicious code in SQL statements via web page input.

→ It occurs when a user is asked for input, like their username / user id

→ In a GET request, the input is sent to the URL and it can be hacked / changed.

If the original query is:

select * from users where username = "admin" and password = "ssn"

This can be modified as

. . . . username = " OR 1=1 -- admin" . . . . . "

1=1 always returns true.

→ By returning True for sql queries always, the malicious user can get into the website

→ In the POST method, there is no explicit input. Even then malicious content can be sent in the input field.

* HTML Tags

(i) <title> → Title of webpage

(ii) h1, h2, h3, h4, h5, h6 → header tags

(iii) <p> → paragraph tag

(iv) <pre> → places text just as written, with formatting

(v) <img src ...> → for image placement

eg. <img src = "... jpg" height = "200"
width = "200" />

(vi) <a> → for links

 My website

(vii) , <b>, <i>, <big>, <small>

↓ ↓ ↓

emphasized bold italics

(viii)
 → horizontal line break

(ix) <hr> → draw a horizontal line

(x) → ~~in-di~~ block element

(xi) → in-line element

* Table Creation

```
<table border="1" cellspacing="20" cellpadding="20">

    <tr> <th> Sno </th> <th> Name </th></tr>

    <tr> <td> 1 </td> <td> Pooja </td></tr>

    <tr> <td> 2 </td> <td> Pranav </td></tr>

</table>
```

* Lists

- ordered list
- unorderd list
- Definition list

(1) Ordered list

```
<ol>
    <li> one </li>
    <li> Two </li>
</ol>
```

1. One
2. Two

(2) Unordered list

```
<ul>
    <li> one </li>
    <li> two </li>
```

- one
- two

(3) Definition list

```
<dl>
<dt> Girls </dt>
<dd> Pooja </dd>
<dd> Rita </dd>
```

```
<dt> Boys </dt>
<dd> Pranav </dd>
<dd> Sanjai </dd>
```

Girls :

Pooja

Rita

Boys :

Pranav

Sanjai

☆☆☆

* Image Maps

→ <map> defines an image map.

→ An image map is an image with clickable areas. The areas are defined with <area> tags.

```
<map name = "planet">
    <area shape = "rect" coords = "0,100,200,300"
          href = "planet.htm" >
</map>
```

for the image

```
< image src = "planet.jpg" usemap = "#planet">
```

* Destination Anchors

* Destination Anchors → use <a> tags with IDs to navigate between portions of the page.

* Forms

- Create a form with text, textarea, email, checkbox, radio, dropdown menu and a submit button. Make use of the required attribute wherever needed.

```
<form>
<h1> My Form </h1>
<label> Name : <input type = "text" name = "name" />
                                                required
</label>
<br>
<label> Comments : <input type = "text area" name = "comment"
                                                />
</label>
<br>
<label> Email Address : <input type = "email" name = "email" />
</label>
<label> Things you liked </label>
```

Separate labels + values for checkboxes

```
<label> Site Design <input type = "checkbox" value =
                                        "site design" />
</label>
<label> Products <input type = "checkbox" value = "product
                                                />
</label>
```

<label> Will you come again? </label>

<label> Yes <input type = "radio" name = "group">

</label>

use the same names

<label> No <input type = "radio" name = "group ">

</label>

<label> Rate us </label>

<select>

<option> 10 </option>

<option> 9 </option>

⋮

</select>

<input type = "Submit" value = "submit />

</form>

* Grouping Related Elements in a Form

<fieldset>

<legend> Date </legend>

<label> - - - - </label>

⋮

</fieldset>

HTML References – " < &
' >

* Additional Attributes

(i) required

(ii) autofocus – speed up data entry if the cursor is placed in the first field

(iii) placeholder – text helps in providing the users on how to fill

* Semantic Elements

<header>
<nav>
<aside>
<article>
<footer>

* Form Input Types

(i) password
(ii) button
(iii) number (up-down arrows to changing no)
(iv) date (pop up calendar)
(v) color (color chart)
(vi) range (scrollable range)
specify min = " " max = " "
(vii) time
(viii) email
(ix) url

* Audio and Video Content

```
<audio>
    <source src = "      .mp3" type = "audio/mp3">
</audio>
```

```
<video>    controls = "controls"  autoplay = "autoplay">
    <source src = "      .mp4"  type = "video/mp4">
</video>
```

---

* CSS

- Adding css to HTML
  (i) @import url ("style.css")
  (ii) <link rel = "stylesheet" href = "style.css">
  (iii) within style tag

* Selectors

→ css has selectors of the following form:

```
selector string {
        property name : declaration
}
```

declaration block

Selector Types
(i) single element — p { }
(ii) multiple element types — h1, h2 { }
(iii) by id — #p1
(iv) universal — *
(v) by class — .{ }

* Styling anchor elements

a : link { color: }

a : hover { }

a : visited { }

a : active { }

* Styling descendant elements

ul ol li { Font-family: }

*** CSS Rule Cascade

1. Style attribute
2. rule with selector
   - (i) id #
   - (ii) class .
   - (iii) descendant / element type
   - (iv) universal

* CSS Inheritance

→ property value is inherited from nearest ancestor element that has a value for that property.

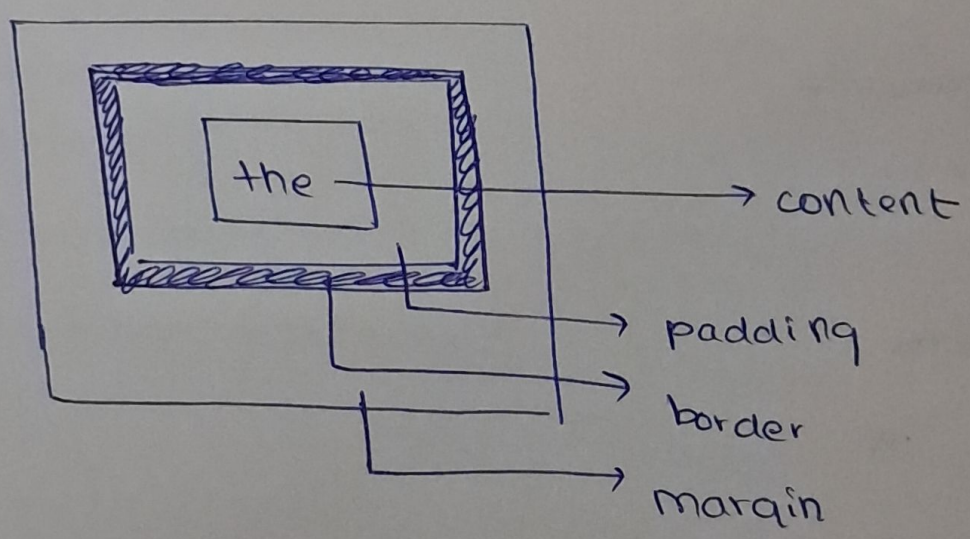
→ Property values can be:
- (i) specified
- (ii) computed (w.r.t parent)
- (iii) actual

* Specifying Font Size

- percentage — font-size : 85%.
- absolute size — font-size : large / smal / xx-large
- relative size (w.rt parent) — font-size : smaller / larger

***

CSS Box Model



* Positioning of Elements

(i) Static - default value, all elements are in the order they appear in the document

(ii) Relative - element is positioned relative to its normal position

(iii) Absolute - the element is positioned absolutely to its first positioned parent

use position:

(iv) Fixed - related to browser window

(v) Sticky - positioned based on scroll position

* Floating Elements

→ put in a span

→ use float: _____

* Transitions, Transformations and Animations

→ To create a transition effect, must specify two things:

→ the css property you want to add an effect to

- duration of the effect

eg1
```
div {
    width:
    height:
    background:
    transition: width 2s;
}
```

eg2
```
div: hover {
    width: 300px
}
```

Speed and Curve of Transition

transition-timing-function: linear
ease
ease-in
ease-out
ease-in-out

Transition Delay

transition-delay: 1s

Transition + Transformation

transform: rotate(180deg);

* Animations

→ gradually change from one style to another

→ must specify key-frames for the animations

eg

```
@keyframes my-animation {
    from { background-color : red; }
    to { background-color : blue; }
}

div {
    width :
    height :
    background-color : red
    animation-name : my animation;
    animation-duration : 4s;
    animation-delay : ____
    animation-iteration-count : ___
    animation-direction : reverse / alternate
}
```

can also specify 0%. 25%. etc

* Background-Images

use border-image : ____

properties : (i) repeat
(ii) round
(iii) stretch

* Images

rounding image → border-radius :

for responsive images use %.

Centering an image

```
img {
    display: block
    margin: auto
    width:     %.
}
```

Fitting Images in a div

```
div {
    width: ____
    height: ____
    border: ____
}
```

<img src = " " object-fit:

contain width = 100% height=100%>

Text in an Image

→ Make image parent w/ relative CSS

→ Set text to absolute CSS with coordinates

Columns

column-count

column-gap

column-rule

column-span

column-width

HTTP Request-Response Structure

HTTP is based on the request-response communication model where the client sends a request, and the server sends back a response.

HTTP is a stateless protocol, that is, the protocol does not require the server to remember anything about the client in between requests.

HTTP Request Structure

Each HTTP request message has the same basic structure:

- Start line
    - Has the format **GET/ HTTP /1.1 – HTTP Request Method/ Request URI /HTTP Version**
    - The HTTP Request Methods that can be used are:
        - **GET-** used if a link is clicked or an address is typed in the browser. There is no body in the request of a GET method.
        - **POST-** Used when the submit button is clicked on a form. The form information is contained in the body of the request.
        - **HEAD-** Requests that only the header fields and no body be returned in the response.
- Header fields
    - Has the format **field name : field value**
    - Some common header fields are:
        - Host – Host name from URL
        - User-Agent- type of browser sending the request
        - Accept
        - Connection- the value of close tells the server to close the connection after a single request/response
        - Content-Type
        - Referer- URL of the document that contains the link that supplied the URL for this HTTP Request
- Blank line
- Message body

HTTP Response Structure

The HTTP Response has the following structure:

- Status Line
  - Has the format **HTTP/1.1 200 OK- HTTP version/status code/Reason phrase (for human use)**
  - The status code is a three digit number, where the first digit conveys the class of the status code
    - 1 – Informational
    - 2 – Success
    - 3 – Redirection
    - 4 – Client Error (eg 400 bad request, 401 unauthorized, 404 not found)
    - 5 – Server Error
- Header Fields
  - Connection, Content-type, content-length
  - Date- data and time at which the response was generated
  - Locaation- alternate URI for redirection
  - Last-modified
  - Expires
  - ETag
- Blank Line
- Optional Body

CAT 1 Q Paper Answers

1. **Develop a HTML program to create links to different sections within the same web page**

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

```
    <h1>My Website</h1>
<header>
    <nav>
        <a href="#section1">Section 1</a>
        <a href="#section2">Section 2</a>
        <a href="#section3">Section 3</a>
        <a href="#section4">Section 4</a>
    </nav>
</header>

<section id="section1">
    <h2>Section 1</h2>
    <p>This is the content of section 1.</p>
</section>

<section id="section2">
    <h2>Section 2</h2>
    <p>This is the content of section 2.</p>
</section>

<section id="section3">
    <h2>Section 3</h2>
    <p>This is the content of section 3.</p>
</section>
```

```
<section id="section4">
    <h2>Section 4</h2>
    <p>This is the content of section 4.</p>
</section>


</body>
</html>
```

2. **Explain in detail the different types of style sheets with the example of changing the style of a paragraph using all three methods.**

    a. Inline Styling

    ```
    <body> <p style="color: blue; font-size: 20px;">This is a paragraph styled using inline CSS.</p> </body>
    ```

    b. Embedded Styling

    ```
    <html>
    <head>
     <style> p { color: green; font-size: 18px; } </style>
    </head>

    <body>
    <p>This is a paragraph styled using internal CSS.</p>
    </body>
    </html>
    ```

    c. External Stylesheet Linking

    Index.html

    ```
    <!DOCTYPE html>
    <head>
       <link rel="stylesheet" href="styles.css">
    ```

```
</head>

<body>
   <p>This is a paragraph styled using external CSS.</p>
</body>
</html>

Style.css
p {
   color: red;
   font-size: 16px;
}
```

3. **Explain in detail the process of CSS rule cascading with the following assumption. Assume that the author, user and user agent style sheets for a HTML document are as follows:**

Author
Div{color:blue}
P{color: green;
Font-size : smaller !important}
.hmm{color: red}

User
P{color:white;
Background-color:black;
Font-size:larger !important;}
Body{color: yellow}

User Agent
Body{color:black}

The CSS specification defines a priority order when multiple CSS rules apply to the same element. The cascading order is as follows:
 Importance (!important):
User !important rules

Author !important rules
Normal (non-!important) rules
 Origin:
User styles- defined by users
Author styles- defined by author
User agent styles- default browser styles
 Specificity:
Inline styles (highest specificity)
ID selectors
Class, attribute, and pseudo-class selectors
Element and pseudo-element selectors (lowest specificity)
 Order:
Later rules override earlier rules if they have the same specificity and origin.

In the given code, the following priority order would apply for each.

Body Tag
User Agent: body { color: black; }
User: body { color: yellow; }

The body color would be yellow since the user has higher priority than the user agent styles.

Paragraph Tag
Author: p {
  color: green;
  font-size: smaller !important;
}

User: p {
  color: white;
  background-color: black;
  font-size: larger !important;
}

In this case- for the font-size, though both have !important, the user font size will apply since user has higher priority than author.
For the color, the user has higher priority, so the color white would be used.
The author has not specified any background color so the user's black background will be applied.

Therefore the final styling for the <p> tag would be:
P{
Color:white;
Background-color:black;
Font-size:larger:}

Div Element
Only the author has a specification for this so the div{color:blue} would be applied by default.

.hmm- Only the author specifies this, so the .hmm would have a color of red.

Old Q Paper Answers for Unit 1

1. **Explain the syntax for creating Image Maps**

```
<html lang="en">
<body>
  <!-- Image with the usemap attribute -->
  <img src="example-image.jpg" alt="Example Image" usemap="#examplemap" width="600" height="400">

  <!-- Image map definition -->
  <map name="examplemap">
```

```
    <!-- Rectangular area -->
    <area shape="rect" coords="50,50,150,150" href="https://example.com"
    <!-- Circular area -->
    <area shape="circle" coords="300,200,50" href=https://example.com" >
    <!-- Polygonal area -->
    <area shape="poly" coords="400,50,450,150,500,50"
href="https://example.com">
  </map>
</body>
</html>
```

2. **Using an appropriate selector, write the CSS style rule to make the h1 content to blue.**
   <h1 id = "internetprogramming" class="sixth">This should be blue</h1>

```
<style>
 #internetprogramming{
  color: blue;
 }
</style>
```

3. **Develop code for a drop-down menu on hovering a nav element using HTML and CSS.**

```
<html lang="en">
<head>
   <link rel="stylesheet" href="styles.css">
</head>
```

```
<body>
   <nav>
      <ul>
         <li class="dropdown">
            <a href="#">Menu</a>
            <ul class="dropdown-content">
               <li><a href="#">Link 1</a></li>
               <li><a href="#">Link 2</a></li>
               <li><a href="#">Link 3</a></li>
            </ul>
         </li>
      </ul>
   </nav>
</body>
</html>
```

Set the display style to block.