# The World Wide Web (WWW)

Kameswari Chebrolu

# Background

- Enormously popular application that provides a tremendous wealth of information
- Origins: 1989 Tim Berners-Lee (CERN) proposed mechanism to distribute high-energy physics data (reports, photos, blueprints etc)
  - Proposal eventually lead to World Wide Web (WWW)
- 1993, first graphical browser Mosaic was released
- 1994, W3C (world wide web consortium) was formed to develop web and standards

# Jargon

- **Web page** consists of base **HTML** file which includes several referenced **objects**
  - Object can be other HTML files, JPEG images, Java applets, audio files,……
  - Text/Image that links to another page is called a **hyperlink** (often highlighted by some means)
- Each object is addressable by a **URL** (Uniform Resource Locator)
  - E.g. http://www.iitb.ac.in/images/header/iitb_logo.gif

Protocol

Host Name

Path

# Jargon

- Web pages are viewed by a program called a **browser**
  - E.g. Internet Explorer, Google Chrome, Mozilla Firefox
- Web pages are written in Hyper Text Markup Language (**HTML**)
  - Describes the structure of a web page using tags
  - Tags: insert paragraphs ( para ), insert images (<img src="url">), include code (<script>code</script>)
- HTML, CSS and Javascript form a technology triad to enable web

# Sample HTML

Run »

```html
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>

<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

## This is a Heading

This is a paragraph.

From https://www.w3schools.com/html/

# Cascading Style Sheets (CSS)

- Used for describing the presentation of a document (layout, colors, fonts etc)
- Separates document content from presentation.

```
body {
   background-color: lightblue;
}
h1 {
   color: navy;
   margin-left: 20px;
}
```

See Demo:
https://www.w3schools.com/css/css_intro.asp

# Javascript

- Javascript (nothing to do with java) is a scripting language
- Permits developers to present interactive and dynamic web content
  - Example: On mouse over, pop up window.

```
< script type="text/javascript" >
function hello() { alert("Hello world!" ); }
< /script>
< img src="picture.jpg"
onMouseOver="javascript:hello()" >
```

Run »

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

<h2>My First JavaScript</h2>

<button type="button"
onclick="document.getElementById('demo').innerHTML = Date()">
Click me to display Date and Time.</button>

<p id="demo"></p>

</body>
</html>
```

## My First JavaScript

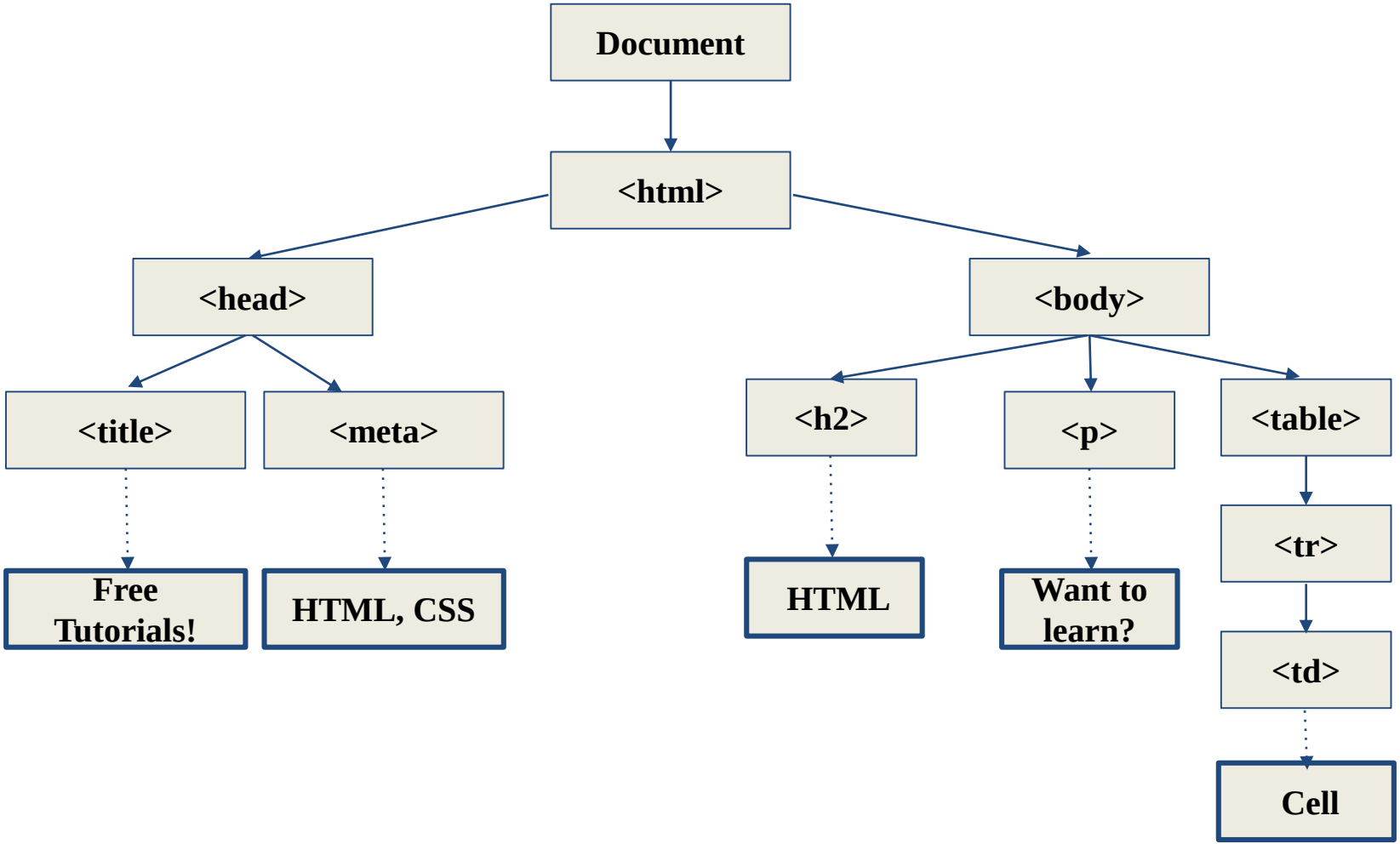
Click me to display Date and Time.

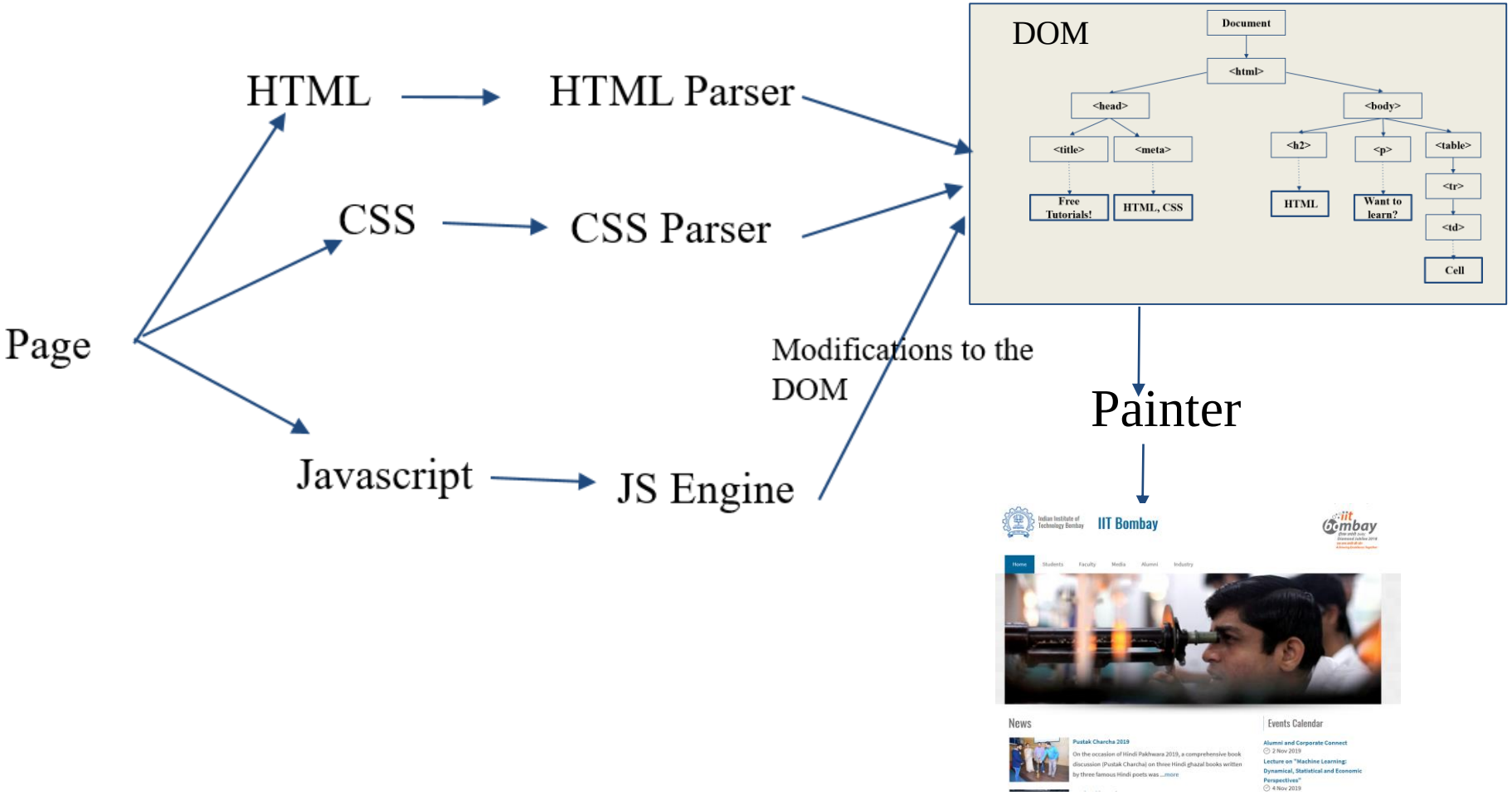
Fri Oct 18 2019 07:53:39 GMT+0530 (India Standard Time)

https://www.w3schools.com/js/tryit.asp?filename=tryjs_myfirst

# Document Object Model (DOM)

- An application programming interface (api) that extracts a tree structure out of HTML
  - Each node is an object representing a part of the document
  - Objects can be manipulated programmatically vis JavaScript


Document
<html>
<head>
<body>
<title>
<meta>
<h2>
<p>
<table>
<tr>
<td>
Free Tutorials!
HTML, CSS
HTML
Want to learn?
Cell

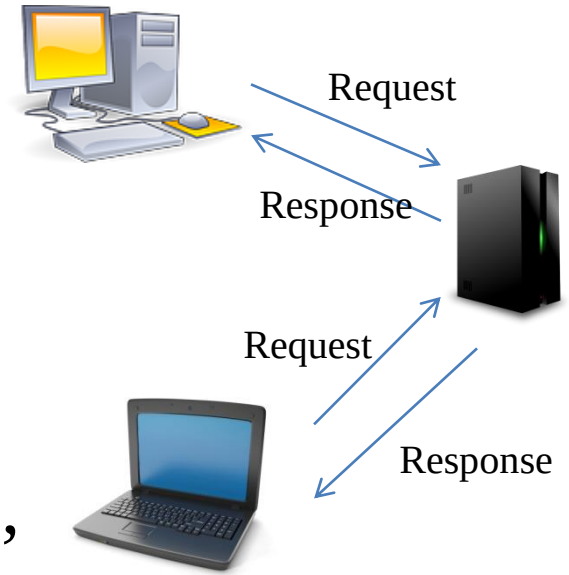

Page
HTML
HTML Parser
CSS
CSS Parser
Javascript
JS Engine
Modifications to the DOM
DOM
Document
<html>
<head>
<body>
<title>
<meta>
<h2>
<p>
<table>
<tr>
<td>
Free Tutorials!
HTML, CSS
HTML
Want to learn?
Cell
Painter
IIT Bombay
Home
Students
Faculty
Media
Alumni
Industry
News
Events Calendar
Pustak Charcha 2019
Alumni and Corporate Connect
Lecture on "Machine Learning: Dynamical, Statistical and Economic Perspectives"

# Plugins/Helpers

- Plugins: Code-module which run within the browser
  - Plugin is removed from browser's memory after job done
  - E.g. Flash player, java plugin
- Helper Applications: A complete program that runs as a separate process
  - E.g. Acrobat reader for pdf; Microsoft word for .doc

# Hyper Text Transfer Protocol (HTTP)

- The protocol employed by Web application
- Based on client-server model
  - Client (browser) requests web objects
  - Server responds with status code and requested object (if present)
- Operates over TCP, server port 80 (http), 443 (https)
- Stateless protocol i.e. no user information stored across requests



# HTTP Message Format

- Two types of messages: Request and Response
- Request Message:

GET /~chebrolu/ HTTP/1.1
Host: www.cse.iitb.ac.in
User-agent: Mozilla/5.0
Connection: close
Accept-language: fr
(blank line)

Header Lines

Request Line

| | | | | | | |
|---|---|---|---|---|---|---|
| Method | sp | URL | sp | Version | cr | lf |
| Header field name: | sp | Value | cr | lf | | |
| Header field name: | sp | Value | cr | lf | | |
| Header field name: | sp | Value | cf | lf | | |
| cr | lf | | | | | |
| Entity Body | | | | | | |

# Methods

| Method | Description |
|---|---|
| GET | Request for a web object |
| HEAD | Request for header fields (no body); Useful for debugging, get time of last modification |
| PUT | Upload an object to a specified path on a web server; body of request contains the object; used with web publishing tools |
| POST | Similar to PUT, except that object contained in body is “appended”; often used when user fills forms; |
| DELETE | Remove the object |
| TRACE | Asks server to echo incoming request; Useful for debugging |
| CONNECT | Used to facilitate secure connection when using Proxy servers |
| OPTIONS | Query server about its properties or that of an object |
| PATCH | Similar to PUT except permits partial modification to an object instead of a full replacement |

# Web Forms

- Allows users to upload content in the form of name-value pairs to a web server
- Can be achieved both via GET and POST
  - GET (preferable for querying database) http://www.example.com/form.php?name=pappu&age=53
  - POST: name, value is included in the request body (preferable when there is a resulting action, e.g. sending email or writing to database)
    - Accidental navigation that triggers GETs does not mess things up

# Response Message Format

Response Line

HTTP/1.1 200 OK
Date: Sat, 02 Nov 2013 12:31:22 GMT
Server: Apache
Last-Modified: Fri, 02 Jan 2009 15:55:54 GMT
ETag: "424c001-fd-f9d8d680"
Accept-Ranges: bytes
Content-Length: 253
Content-Type: text/html; charset=UTF-8

Header Lines

data data data data data ...

| | | | | | | |
|---|---|---|---|---|---|---|
| Version | sp | Status Code | sp | Phrase | cr | lf |
| Header field name: | sp | Value | cr | lf | | |
| Header field name: | sp | Value | cr | lf | | |
| Header field name: | sp | Value | cr | lf | | |
| cr | lf | | | | | |
| Entity Body | | | | | | |

# Sample Status Code and Phrases

| Status Code | Phrase | Description |
|---|---|---|
| 200 | OK | Request successful, information enclosed |
| 301 | Moved Permanently | Object moved; new url under Location: |
| 400 | Bad Request | Request could not be understood |
| 404 | Not Found | Requested object not found on server |
| 503 | Service Unavailable | Server is currently unavailable (overloaded) |
| 505 | HTTP Version not supported | Server does not support the HTTP version |

# Web Sessions and Cookies

- Session captures interaction between client and server
- Client information often should persist across loading of various web pages
  - E.g. User is authenticated or items in cart
- But HTTP is a stateless protocol. What to do?

# Use of GET/POST

- Server generates some invisible code capturing user's session information
- Inserts it into page delivered to client using hidden fields
- Each time user navigates to a new page, code (executed by browser) passes the user's session information to server
- The cycle repeats
- This method is particularly susceptible to man-in-the-middle attacks unless HTTPs is used

# Cookies

- Server sends ‘small amount of data (cookie)’ to store at client
- Every time client contacts the server (same domain), browser sends cookie (of the domain) to server

# Cookie Structure

Browsers need to support at least 4KB cookie size

- Name of cookie
- Value of cookie
- Expiry of cookie
- Path the cookie is good for
- Domain of the cookie
- Type of connection needed

If no expiration date, cookie deleted when user exits browser → **Nonpersistent Cookie**

# Example

- Browser to Server

  Post /users/info HTTP/1.1

  [form contents which includes user identification]

- Server to Browser

  HTTP/1.1 200 OK

  Set-Cookie: Customer=“Pappu”

  (Server also maintains an entry in backend database)

# Example

- Browser to Server

  GET /info.html HTTP/1.1

  Cookie: Customer=“Pappu”

[Server accesses backend to retrieve information logged and acts accordingly]

# Things to Note

- Only hosts within a domain can set a cookie for that domain
- A subdomain can set a cookie for a higher-level domain, but not vice versa
  - mail.example.com could access cookies set for example.com or mail.example.com
  - example.com cannot access cookies set for mail.example.com
- Normally cannot set cookies for top-level domains such as .edu or .com (enforced at the browser level)

- Path field: cookie can be accessed within a specific subdirectory of the web site
  - Defaults to the root directory of a given domain
- Secureflag: Cookie be transmitted using HTTPS
- HTTP-Onlyflag: scripting languages cannot access/manipulate cookies stored on the client

# Session ID/Token

- Associates a given session with a particular client via a unique identifier (Id or token)
- Server sets this via a non-persistent cookie (server may also use persistent cookies on top)
- A session ID should be hard to guess (hence random; sometimes protected by MAC)

# Summary

- Web provides a tremendous wealth of information
- HTML, CSS and Javascript triad help enable web
- HTTP protocol helps in client-server communication
  - Methods, Forms, Cookies and Tokens