# Web Technologies (HTML CSS JavaScript)

Short notes for B.Tech CSE / IT · BCA · MCA · M.Tech

Uncodemy Handwritten Notes · free for students

## I. Internet and Web Basics

- **Internet** = a global network of networks. **World Wide Web (WWW)** = a service on the internet made of linked web pages.
- WWW was proposed by Tim Berners-Lee (1989).
- **Client** (browser) sends a request; **server** sends back a response.
- **URL** = address of a resource: `protocol://host:port/path?query#fragment`.
- **DNS** converts a domain name into an IP address.

### ➤ HTTP

- **HTTP** = stateless request-response protocol. **HTTPS** = HTTP over TLS (encrypted).
- Default ports: HTTP `80`, HTTPS `443`.

| Method | Use |
|---|---|
| GET | Fetch data (data in URL) |
| POST | Send data in the body (forms) |
| PUT | Replace a resource |
| PATCH | Partly update a resource |
| DELETE | Remove a resource |

| Status code | Meaning |
|---|---|
| 1xx | Informational |
| 2xx | Success (200 OK, 201 Created) |
| 3xx | Redirect (301 Moved Permanently, 302 Found) |
| 4xx | Client error (400, 403 Forbidden, 404 Not Found) |
| 5xx | Server error (500, 503) |

**EXAM POINT** GET vs POST: GET puts data in the URL, can be bookmarked and cached, and has length limits. POST sends data in the body and is used for sensitive or large data.

# 2. HTML

- **HTML** = HyperText Markup Language. It gives structure to a page.
- Elements are written with tags: `<p>text</p>`. Some are empty: `<br>`, `<img>`, `<hr>`.

```
<!DOCTYPE html>
<html>
<head>
  <title>My Page</title>
</head>
<body>
  <h1>Hello</h1>
  <p>First page.</p>
</body>
</html>
```

## Common tags

- Headings `<h1>` to `<h6>`, paragraph `<p>`, link `<a href="">`, image `<img src="" alt="">`.
- Lists: `<ul>` unordered, `<ol>` ordered, `<li>` item, `<dl>` description list.
- Table: `<table>`, `<tr>` row, `<th>` header cell, `<td>` data cell. Merge cells with `colspan`, `rowspan`.
- Block elements (`<div>`, `<p>`, `<h1>`) start on a new line. Inline elements (`<span>`, `<a>`, `<b>`) do not.

## Forms

- `<form action="url" method="post">` holds inputs.
- Input types: `text`, `password`, `email`, `number`, `radio`, `checkbox`, `date`, `file`, `submit`.
- Other controls: `<select>` + `<option>`, `<textarea>`, `<label for="id">`.
- Attributes: `name`, `value`, `placeholder`, `required`, `maxlength`.

### HTML5 additions

- Semantic tags: `<header>`, `<nav>`, `<main>`, `<section>`, `<article>`, `<aside>`, `<footer>`.
- Media: `<audio>`, `<video>`. Graphics: `<canvas>`, `<svg>`.
- Web storage: `localStorage` (no expiry), `sessionStorage` (cleared when the tab closes).

> **TIP** Always give `<img>` an `alt` text. It helps screen readers and shows when the image fails to load.

## 3. CSS

- **CSS** = Cascading Style Sheets. It controls presentation (colour, layout, fonts).
- Syntax: `selector { property: value; }`

### Ways to add CSS

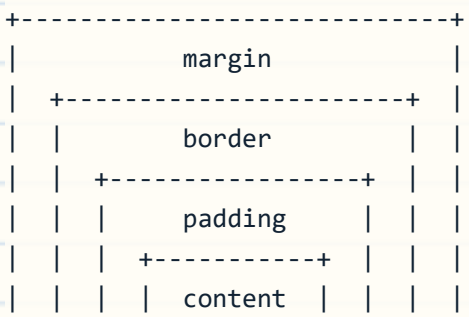
| Type | How | Priority |
|---|---|---|
| Inline | `style=""` attribute | Highest |
| Internal | `<style>` in head | Same level as external, later one wins |
| External | `<link rel="stylesheet" href="a.css">` | Same level as internal, later one wins |

### Selectors

- Element `p`, class `.box`, id `#top`, universal `*`, group `h1, h2`.
- Descendant `div p`, child `div > p`, attribute `input[type="text"]`.
- Pseudo-class `a:hover`, `li:first-child`. Pseudo-element `p::first-line`.

> **FORMULA** Specificity order: inline (1,0,0,0) > id (0,1,0,0) > class / attribute / pseudo-class (0,0,1,0) > element (0,0,0,1). `!important` overrides all.

### Box model



**EXAMPLE** width 200px, padding 10px, border 5px (content-box). Total width = 200 + 2(10) + 2(5) = 230px (margin not counted).

- `box-sizing: border-box` makes width include padding and border.

### ➤ Positioning and layout

- `position`: `static` (default), `relative`, `absolute` (to nearest positioned parent), `fixed` (to viewport), `sticky`.
- `display`: `block`, `inline`, `inline-block`, `none`, `flex`, `grid`.
- **Flexbox** = one-dimensional layout. Key: `justify-content` (main axis), `align-items` (cross axis), `flex-direction`.
- **Grid** = two-dimensional layout: `grid-template-columns: 1fr 2fr;`.
- **Responsive design**: media queries, e.g. `@media (max-width: 600px) { ... }`.

## 4. JavaScript Basics

- **JavaScript** = scripting language that adds behaviour to pages. Runs in the browser (and on servers with Node.js).
- Add it with `<script>` inline or `<script src="app.js">`.

### ➤ Variables

| Keyword | Scope | Re-assign | Hoisted |
|---|---|---|---|
| `var` | Function | Yes | Yes (as undefined) |
| `let` | Block | Yes | Yes, but in temporal dead zone |
| `const` | Block | No | Yes, but in temporal dead zone |

- Data types: number, string, boolean, undefined, null, bigint, symbol (primitive) and object.
- `typeof null` returns `"object"` (a known quirk).

### ➤ Operators

- `==` compares after type conversion. `===` compares value and type.

**EXAMPLE** `5 == "5"` is true, `5 === "5"` is false. `"5" + 2` = `"52"`, `"5" - 2` = 3.

# 5. Functions, Objects and Arrays

```
function add(a, b) { return a + b; }
const mul = (a, b) => a * b;      // arrow function
let student = { name: "Aman", age: 20 };
console.log(student.name);
```

- **Closure** = an inner function that remembers variables of its outer function even after it returns.
- Array methods: `push`, `pop`, `shift`, `unshift`, `slice`, `splice`, `map`, `filter`, `reduce`, `forEach`.
- `JSON.stringify(obj)` converts to text, `JSON.parse(text)` converts back.

**EXAMPLE** `[1, 2, 3].map(x => x * 2)` gives `[2, 4, 6]`. `[1, 2, 3, 4].filter(x => x % 2 == 0)` gives `[2, 4]`.

# 6. DOM and Events

- **DOM** = Document Object Model. The browser stores the page as a tree of objects that JS can change.
- Select: `document.getElementById("id")`, `querySelector(".cls")`, `querySelectorAll("p")`.
- Change: `el.innerHTML`, `el.textContent`, `el.style.color`, `el.classList.add("x")`.
- Create: `document.createElement("li")`, `parent.appendChild(child)`.

## ➤ Events

- Common: `click`, `submit`, `change`, `keyup`, `load`, `mouseover`.
- `el.addEventListener("click", fn)`.
- **Event bubbling**: event goes from the target up to its parents. **Capturing**: from the top down.
- `event.preventDefault()` stops the default action (e.g. form submit). `event.stopPropagation()` stops bubbling.

### Form validation

```
document.querySelector("form").addEventListener("submit", e => {
  const email = document.getElementById("email").value;
  if (!email.includes("@")) {
    alert("Invalid email");
    e.preventDefault();
  }
});
```

# 7. Asynchronous JavaScript

- JS is single-threaded. The **event loop** runs callbacks when the call stack is empty.
- **Callback** = function passed to run later. Too many nested callbacks = "callback hell".
- **Promise** = object for a future value. States: pending, fulfilled, rejected.
- **async / await** = cleaner syntax for promises.
- **AJAX** = updating part of a page without reload. Older: `XMLHttpRequest`. Modern: `fetch()`.

```
async function load() {
  const res = await fetch("/api/users");
  const data = await res.json();
  console.log(data);
}
```

EXAM POINT `setTimeout(fn, 0)` still runs after the current code finishes, because it waits in the queue for the event loop.

# 8. XML, JSON and Server Side

### XML vs JSON

| Point | XML | JSON |
|---|---|---|
| Format | Tags | Key-value pairs |
| Size | Larger | Smaller |
| Parsing in JS | Needs a parser | `JSON.parse()` |
| Schema | DTD, XSD | JSON Schema |

- **Well-formed XML**: one root, every tag closed, proper nesting, case-sensitive. **Valid XML** = well-formed + follows its DTD / XSD.

### ➤ Server side basics

- Server-side languages: PHP, Node.js, Java (Servlets, JSP), Python.
- **Cookie** = small data stored in the browser and sent with every request. **Session** = data stored on the server, linked by a session ID.
- **Static page** = same for every user. **Dynamic page** = built at request time.
- **REST API**: resources at URLs, HTTP methods as actions, usually JSON data, stateless.

## 9. Web Security Basics

- **XSS (Cross-Site Scripting)**: attacker injects script into a page. Fix: escape output, Content Security Policy.
- **SQL injection**: input changes an SQL query. Fix: prepared statements.
- **CSRF**: a forged request using the user's logged-in session. Fix: CSRF tokens, SameSite cookies.
- **CORS**: browser rule controlling which other origins may call your server.

## 10. Most Asked Questions (practice)

1. Explain HTTP request-response. Difference between GET and POST.
2. Write an HTML form for student registration with validation.
3. Explain the CSS box model with a diagram and a calculation.
4. Types of CSS and the cascading / specificity rule.
5. Difference between Flexbox and Grid.
6. Difference between var, let and const. What is hoisting?
7. What is the DOM? Write JS to add an item to a list on button click.
8. Explain event bubbling and capturing.
9. What is a Promise? Explain async / await and fetch.
10. XML vs JSON. Cookie vs session.

> **TIP** Most marks come from writing code: an HTML form, a CSS layout and a JS validation script. Practise writing them by hand with correct tags.