Web Designing

Short notes for beginners · HTML · CSS · BCA · diploma · freshers

Uncodemy Handwritten Notes · free for students

1. Web Basics

- The web works on a **client-server** model: the browser (client) sends an HTTP request, the server returns a response (HTML, CSS, JS, images). The browser then renders the page.
- Three building blocks: **HTML** = structure and content, **CSS** = style and layout, **JavaScript** = behaviour and interaction. Design = planning how the site looks and feels.

➤ Parts of a URL

Part	Example	Meaning
Protocol	https://	Rules for communication (HTTPS is secure)
Domain	www.example.com	Name of the website, resolved by DNS to an IP
Path	/blog/post-1	Page or resource on the server
Query	?id=5&lang=en	Extra data sent to the server
Fragment	#contact	Jumps to a section of the page

- Common HTTP status codes: 200 OK, 301 moved permanently, 404 not found, 500 server error. Domain name + hosting = website live on the internet.
- Tools: VS Code (editor), Live Server extension, Chrome DevTools (F12 to inspect and edit live), Git and GitHub, Figma for design.

2. HTML Structure and Tags

- Every page starts with <!DOCTYPE html>, then <html lang="en">, a <head> (title, meta, CSS links) and a <body> (visible content).
- **Element** = opening tag + content + closing tag. **Attributes** give extra info: <a href="url" target="_blank">, <img src="a.jpg" alt="text">. Void tags like <br>, <hr>, <img>, <input> have no closing tag.

Tag	Use
h1 to h6	Headings, h1 most important (one per page)
p, br, hr	Paragraph, line break, horizontal line
a	Link (href)
img	Image (src, alt, width, height)
ul, ol, li	Unordered, ordered list and list items
table, tr, th, td	Table, row, header cell, data cell
div, span	Generic block container, generic inline container
strong, em	Important text (bold), emphasis (italic)

- **Block elements** (div, p, h1, ul) start on a new line and take full width. **Inline elements** (span, a, strong, img) stay in the line and take only needed width.
- **Semantic tags** give meaning: <header>, <nav>, <main>, <section>, <article>, <aside>, <footer>. They help SEO, screen readers and clean code.

> **REMEMBER** Use alt text on every image, a single h1 per page, and the meta viewport tag: <meta name="viewport" content="width=device-width, initial-scale=1">.

3. Forms, Tables and Media

- <form action="/submit" method="post"> collects input. GET shows data in the URL (search), POST sends in the body (login, signup).

Input type	Use
text, email, password	Single line text, email check, hidden characters
number, date, range	Numeric, date picker, slider
checkbox / radio	Many choices / one choice (same name)
file, submit, reset	Upload file, send form, clear form

- Other controls: <textarea>, <select> with <option>, <button>. Link a <label for="id"> to every input. Validation attributes: required, minlength, maxlength, pattern, min, max.
- Media: <audio controls>, <video controls>, <iframe> (embed YouTube or maps). Use <picture> and srcset to give the browser different image sizes.
- Tables are for data only, not for page layout. Use rowspan and colspan to merge cells, <thead>, <tbody>, <tfoot> to group rows.

4. CSS Basics and Selectors

- Syntax: selector { property: value; }. Three ways to add CSS: **inline** (style attribute), **internal** (<style> in head), **external** (<link rel="stylesheet" href="style.css">). External is best practice.

Selector	Example	Selects
Element	p	All paragraphs
Class	.card	Elements with class="card"
ID	#header	The one element with that id
Descendant / child	nav a / nav > a	Any a inside nav / direct child only
Group	h1, h2	Both h1 and h2
Pseudo-class	a:hover	State such as hover, focus, first-child
Pseudo-element	p::first-line, ::before	Part of an element, generated content

- **Specificity** decides which rule wins: inline = 1000, id = 100, class / attribute / pseudo-class = 10, element = 1. If equal, the later rule wins. !important overrides, so avoid it.

EXAMPLE Selector #nav .link a:hover = 100 (id) + 10 (class) + 1 (element) + 10 (pseudo-class) = 121. It beats .link a (11).

- Colours: names, HEX (#1F3A93), rgb(31,58,147), rgba (with opacity), hsl. Units: px (fixed), % (parent), em (parent font size), rem (root font size, 1rem = 16px by default), vw / vh (viewport).

5. Box Model and Positioning

- Every element is a box: **content → padding → border → margin**. Padding is inside the border, margin is outside and is transparent.
- Use box-sizing: border-box so that width includes padding and border. Default content-box adds them on top of the width.

EXAMPLE width 200px, padding 20px, border 5px. content-box total = 200 + 2x20 + 2x5 = 250px. With border-box the total stays 200px.

position	Behaviour
static	Default, normal flow
relative	Offset from its normal place, still takes its space
absolute	Placed relative to nearest positioned parent, removed from flow
fixed	Fixed to the viewport, stays on scroll (sticky header)
sticky	Normal until a scroll point, then sticks

- z-index controls stacking order of positioned elements. **display**: block, inline, inline-block, none (removes element), flex, grid. visibility: hidden hides but keeps space. **overflow**: visible, hidden, scroll, auto.
- Centering: block with width → margin: 0 auto; text → text-align: center; anything → flexbox with justify-content and align-items center.

6. Text, Backgrounds and Effects

- Text: font-family (add a fallback like sans-serif), font-size, font-weight, line-height (1.5 reads well), letter-spacing, text-align, text-decoration, text-transform. Load web fonts from Google Fonts.
- Backgrounds: background-color, background-image, background-size: cover, background-position, background-repeat, linear-gradient(to right, #fff, #000).
- Effects: border-radius (rounded, 50% = circle), box-shadow: 0 4px 12px rgba(0,0,0,.2), opacity, transform (translate, scale, rotate), filter (blur).
- **transition**: smooth change, e.g. transition: all 0.3s ease on hover. **animation**: define @keyframes name { from {...} to {...} } then animation: name 2s infinite.
- Use CSS variables for easy theming: :root { --main: #1F3A93; } then color: var(--main). Keep contrast readable (dark text on light background).

7. Flexbox and Grid

- **Flexbox** = one dimensional layout (row or column). Parent: display: flex; flex-direction; justify-content (main axis); align-items (cross axis); flex-wrap; gap. Child: flex: 1; order; align-self.

justify-content	Result
flex-start / flex-end	Items at start / end
center	Items centred
space-between	Equal gaps between, none at edges
space-around / space-evenly	Equal space around / everywhere

- **CSS Grid** = two dimensional layout (rows and columns). display: grid; grid-template-columns: repeat(3, 1fr); gap: 16px; grid-column: span 2 to span items.

EXAMPLE Responsive card grid in one line: grid-template-columns: repeat(auto-fit, minmax(250px, 1fr)); cards fill the row and wrap by themselves.

- Use Flexbox for navbars, buttons and alignment in a line. Use Grid for full page layouts and card galleries. They can be combined.

8. Responsive Design

- **Responsive design** = one website that adapts to every screen size. Three ingredients: fluid grids (% / fr units), flexible images (max-width: 100%; height: auto) and media queries.
- Media query: @media (max-width: 768px) { .menu { display: none; } }. Use **mobile first**: write base CSS for phones and add @media (min-width: 768px) for larger screens.

Device	Common breakpoint
Phones	Up to 576 px
Tablets	768 px
Laptops	992 px
Desktops	1200 px and above

- **Bootstrap** uses a 12 column grid with classes like col-md-6 (half width from 768 px). **Tailwind CSS** uses utility classes (flex, p-4, md:grid-cols-3). Frameworks speed up work but learn plain CSS first.
- Use clamp() for fluid text: font-size: clamp(1rem, 2.5vw, 2rem). Test on real phones and with Chrome DevTools device mode (Ctrl+Shift+M).

9. SEO, Performance and Publishing

- SEO basics: unique <title> under 60 characters, meta description about 150 to 160 characters, one h1, semantic HTML, descriptive alt text, clean URLs, fast loading, mobile friendly.
- Speed: compress images, use WebP, add loading="lazy", minify CSS / JS, limit fonts, avoid heavy libraries. Aim for good Core Web Vitals (LCP, CLS, INP).
- Accessibility: sufficient contrast, keyboard focus styles, labels, aria-label where needed, readable font size at least 16 px.
- Publishing: buy a domain, use free hosting like GitHub Pages, Netlify or Vercel, or a paid host. Push code with Git. Check with W3C validator and Lighthouse.

10. Most Asked Questions (practice)

1. Difference between HTML, CSS and JavaScript. How does a browser show a page?
2. Block vs inline elements. What are semantic tags?
3. Three ways to add CSS. Which is best and why?
4. Explain the CSS box model. What does border-box do?

5. Explain CSS specificity with an example.

6. Difference between position relative, absolute, fixed and sticky.

7. Flexbox vs Grid. When do you use which?

8. What are media queries? What is mobile first design?

9. How do you create a registration form with validation?

10. Build a responsive navbar and card layout (practical).

TIP Build 5 small projects: personal portfolio, landing page, restaurant menu, blog layout and a clone of a simple site. Host them and share the links in your resume.