HTML CSS

Tags, Flexbox, Grid and Layouts, explained simply

HTML and CSS Notes · easy to understand · free for students

1. Start with HTML and CSS

- **HTML** builds the **structure** of a web page: headings, text, images, links, forms.
- **CSS** controls the **look**: colors, fonts, spacing and layout. **JavaScript** adds the behaviour.

IDEA Think of a human body. HTML is the skeleton, CSS is the clothes and style, JavaScript is the movement.

Language	Job	Example
HTML	Content and structure	Heading, paragraph, image
CSS	Design and layout	Color, spacing, flexbox
JavaScript	Interaction	Button click, form check

EXAMPLE Basic page skeleton. Save it as index.html.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>My Page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Hello World</h1>
</body>
</html>
```

- Code goes in a text editor like **VS Code**. Open `index.html` in the browser. The `Live Server` extension refreshes it for you.

2. HTML Tags

- A **tag** wraps content: `<p>text</p>` has an opening tag, content and a closing tag. Some tags are self-closing, like `<img>` and `<br>`.
- **Attributes** give extra information inside the opening tag: `<a href='page.html'>`.

Tag	Use
<h1> to <h6>	Headings, h1 is the biggest
<p>	Paragraph
<a href>	Link to another page
<img src alt>	Image. Always write alt text
	Bullet list, numbered list, list item
	Generic boxes for grouping and styling
  <hr>	Line break, horizontal line
<b> <i>	Bold, italic text

EXAMPLE Common tags together.

```
<h1>Main title</h1>
<p>This is a <b>bold</b> and <i>italic</i> word.</p>
<a href="https://example.com" target="_blank">Visit</a>
<img src="cat.jpg" alt="A cat" width="200">
<ul>
  <li>HTML</li>
  <li>CSS</li>
</ul>
```

➤ Block and inline elements

- **Block** elements start on a new line and take full width: `div`, `p`, `h1`, `ul`.
- **Inline** elements stay in the line and take only needed width: `span`, `a`, `b`, `img`.

EXAMPLE A simple table.

```
<table>
  <tr><th>Name</th><th>Age</th></tr>
  <tr><td>Asha</td><td>20</td></tr>
</table>
```

3. Forms and Semantic HTML

- A **form** collects user data. Each input needs a `name`. A `label` linked with `for` and `id` makes the field easy to click.

EXAMPLE Sign up form.

```
<form action="/signup" method="post">
  <label for="email">Email</label>
  <input type="email" id="email" name="email" required>
  <input type="password" name="pass" placeholder="Password">
  <select name="city"><option>Delhi</option><option>Pune</option></select>
  <textarea name="msg"></textarea>
  <button type="submit">Sign up</button>
</form>
```

- *Input types:* `text`, `email`, `password`, `number`, `checkbox`, `radio`, `date`, `file`. *Useful attributes:* `required`, `placeholder`, `disabled`.

➤ Semantic tags

- ***Semantic*** *tags tell the meaning of a section. They are better for SEO, screen readers and readable code than many plain* `div` *tags.*

Tag	*Meaning*
`<header>`	*Top part: logo, title, navigation*
`<nav>`	*Group of navigation links*
`<main>`	*Main content of the page (use once)*
`<section> <article>`	*A topic block / a stand-alone post*
`<aside>`	*Sidebar or related content*
`<footer>`	*Bottom part: copyright, links*

4. CSS Basics

- *CSS rule =* ***selector*** *+* ***declarations****. The selector picks the element and each declaration is* `property: value;`.
- *Three ways to add CSS:* ***inline*** *(`style=` attribute),* ***internal*** *(`<style>` tag),* ***external*** *file linked with* `<link>`*. Use the external file in real projects.*

EXAMPLE *A small style.css.*

```
/* style.css */
body { font-family: Arial, sans-serif; margin: 0; }
h1 { color: #e8532b; text-align: center; }
.card { background: #fff; padding: 16px; border-radius: 8px; }
#menu a:hover { text-decoration: underline; }
```

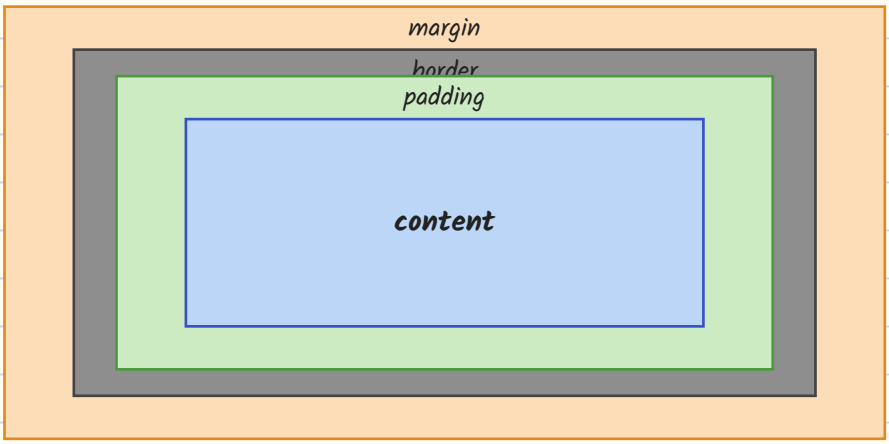
Selector	*Example*	*Picks*
Element	`p`	*All paragraphs*
Class	`.card`	*Elements with class='card'*
ID	`#menu`	*The one element with id='menu'*
Group	`h1, h2`	*Both h1 and h2*
Descendant	`div p`	*Any p inside a div*
Child	`ul > li`	*li directly inside ul*
Hover	`a:hover`	*Link when mouse is over it*

REMEMBER ***Specificity****: inline style beats ID, ID beats class, class beats element. If equal, the later rule wins. Avoid* `!important`.

Color format	Example
Name	red, tomato
Hex	#ff5733
RGB / RGBA	rgb(255, 87, 51), rgba(0, 0, 0, 0.5) (last is see-through)
HSL	hsl(11, 100%, 60%)

5. The Box Model

- Every element is a box made of four layers: **content**, **padding** (inside space), **border** (the edge), **margin** (outside space).



EXAMPLE Box with all layers.

```
.box {
  width: 200px;
  padding: 20px;
  border: 2px solid #333;
  margin: 16px auto;       /* auto left and right = centered */
  box-sizing: border-box;  /* width includes padding and border */
}
```

- Shorthand order is **top right bottom left** (clockwise): `margin: 10px 20px 10px 20px`. Two values mean up-down and left-right: `padding: 10px 20px`.

REMEMBER Always add `*, *::before, *::after { box-sizing: border-box; }` at the top. Then width means the full visible width, which is much easier.

6. Display, Position and Units

display	Behaviour
block	New line, full width, width and height work
inline	Stays in line, width and height do not work
inline-block	Stays in line, width and height work
none	Element is hidden and takes no space
flex / grid	Layout systems (next sections)

position	Behaviour
static	Default, normal flow
relative	Normal flow, can be shifted. Also a parent for absolute children
absolute	Placed inside the nearest positioned parent using top, left...
fixed	Stays at the same place on screen while scrolling
sticky	Normal at first, then sticks when you scroll to it

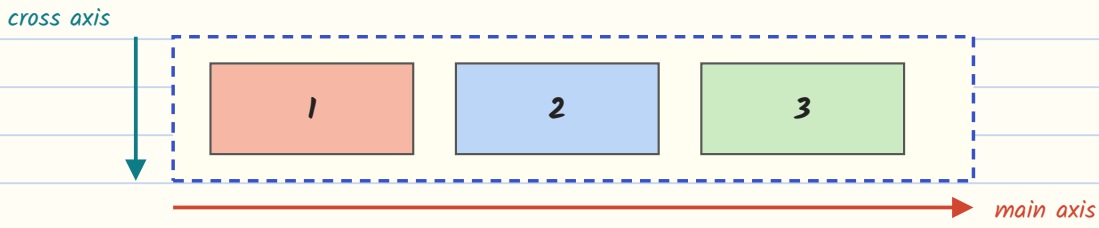
EXAMPLE Sticky navbar and a badge on a card.

```
nav { position: sticky; top: 0; }
.card { position: relative; }
.badge { position: absolute; top: 8px; right: 8px; }
```

Unit	Meaning
px	Fixed pixels
%	Percent of the parent size
em	Relative to the parent font size
rem	Relative to the root (html) font size. Good for text and spacing
vw / vh	1% of screen width / height

7. Flexbox

- ***Flexbox*** arranges items in one direction, a row or a column. Put `display: flex` on the ***parent*** and it controls its children.



Container property	What it does
flex-direction	`row` (default) or `column`. This sets the main axis
justify-content	Space along the **main axis**: start, center, space-between, space-around
align-items	Space along the **cross axis**: stretch, center, flex-start, flex-end
flex-wrap	`wrap` lets items go to the next line
gap	Space between items, like `gap: 16px`

EXAMPLE Center anything in the page.

```
.center {
  display: flex;
  justify-content: center;   /* left to right */
  align-items: center;       /* top to bottom */
  height: 100vh;
}
```

EXAMPLE Navbar and wrapping cards.

```
nav { display: flex; justify-content: space-between; align-items: center; }
.cards { display: flex; flex-wrap: wrap; gap: 16px; }
.card { flex: 1 1 250px; }   /* grow, shrink, base width */
```

- Item properties: `flex: 1` (take free space), `align-self` (change one item), `order` (change the order).

REMEMBER With `flex-direction: column` the main axis turns vertical. Then `justify-content` works up-down and `align-items` works left-right.

8. CSS Grid

- **Grid** arranges items in rows and columns together (2 directions). Put `display: grid` on the parent.
- The **fr** unit means a fraction of the free space. `repeat(3, 1fr)` makes 3 equal columns.

EXAMPLE Three equal columns.

```
.grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);   /* 3 equal columns */
  gap: 16px;
}
.wide { grid-column: span 2; }             /* this item takes 2 columns */
```

Property	Meaning
grid-template-columns / rows	Define the size of columns and rows
gap	Space between rows and columns
grid-column / grid-row	Place or stretch an item, like `span 2`
grid-template-areas	Draw the layout with names
place-items	Center items inside their cells

Responsive cards with no media query

EXAMPLE Columns appear as space allows.

```
.cards {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));
  gap: 16px;
}
```

Full page layout with areas

EXAMPLE Header, sidebar, main and footer.

```
.page {
  display: grid;
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer";
  grid-template-columns: 220px 1fr;
}
header { grid-area: header; }
aside  { grid-area: sidebar; }
main   { grid-area: main; }
footer { grid-area: footer; }
```

	Flexbox	Grid
Direction	One (row or column)	Two (rows and columns)
Works from	Content outward	Layout inward
Best for	Navbar, buttons, centering	Page layout, card galleries

9. Responsive Layouts

- **Responsive** means the page looks good on phone, tablet and laptop. First add the viewport tag in `<head>` (see the skeleton in section 1).
- **Mobile first**: write CSS for the small screen first, then use `min-width` media queries to improve it for bigger screens.

EXAMPLE Media query.

```
.cards { display: grid; grid-template-columns: 1fr; }  /* mobile */

@media (min-width: 768px) {
  .cards { grid-template-columns: repeat(3, 1fr); }  /* tablet and up */
}
img { max-width: 100%; height: auto; }   /* images never overflow */
```

Screen	Common breakpoint
Phone	Up to about 480px (base styles)
Tablet	768px and up
Laptop	1024px and up
Large screen	1280px and up

Layout need	Best tool
Navbar in a row	Flexbox
Center a box	Flexbox
Gallery of cards	Grid
Sidebar and content page	Grid with areas
Buttons in a line	Flexbox

- Use relative units (`rem`, `%`, `vw`) and `max-width` instead of fixed widths. `clamp(1rem, 2vw, 2rem)` makes fluid font sizes.

10. Colors, Fonts and Effects

EXAMPLE CSS variables, hover and transition.

```
:root { --brand: #e8532b; }

.btn {
  background: var(--brand);
  color: #fff;
  padding: 10px 20px;
  border: none;
  border-radius: 6px;
  transition: 0.3s;
}
.btn:hover {
  transform: scale(1.05);
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.2);
}
```

Property	Example
font-family	'Poppins', Arial, sans-serif (first is tried first)
font-size / weight	1.2rem, 700
text-align / line-height	center, 1.6
background	linear-gradient(to right, #ff7e5f, #feb47b)
border-radius	8px, or 50% for a circle
box-shadow	0 4px 12px rgba(0,0,0,0.2)
transition / transform	Smooth change / move, scale, rotate

- Pseudo-classes react to state, like `:hover`, `:focus`, `:first-child`. Pseudo-elements like `::before` and `::after` add extra decoration.

II. Projects and Practice

Project	What you practice
1. Personal portfolio page	Semantic tags, box model, images
2. Responsive navbar	Flexbox, media queries, hover
3. Landing page	Sections, buttons, gradients, flexbox
4. Pricing cards	Grid, box-shadow, transitions
5. Photo gallery	Grid auto-fit, images, hover effects
6. Blog layout	Grid areas, typography, responsive design
7. Sign up form page	Forms, labels, validation, centering

➤ Common mistakes

Mistake	Fix
No viewport meta tag	Add it, or the page will not look right on phones
Missing alt on images	Always write a short alt text
Using div for everything	Use header, nav, main, section, footer
Width plus padding is too big	Use `box-sizing: border-box`
justify-content not working	Parent needs `display: flex`

➤ Practice questions

1. What is the difference between block, inline and inline-block?
2. Explain the CSS box model with a diagram.
3. Difference between class and id selectors? What is specificity?
4. What is semantic HTML and why use it?
5. Explain flexbox: main axis, cross axis, justify-content, align-items.
6. Flexbox vs Grid. When do you use which?
7. How do you make a website responsive?
8. Build a 3-column card layout that becomes 1 column on mobile.

- Build one small page every week. Copying a real website's layout is great practice.