



By: Mohamed Saied Hassan

CSS

From HTML Structure to Professional Web Pages

Lecture 2 — Frontend Development

CSS Fundamentals



Chapter

00

What Will We Learn?

- Html Multimedia & Iframe
- What is CSS?
- CSS Syntax
- Ways to apply CSS
- CSS Selectors
- Colors & Backgrounds
- Text & Fonts
- Borders
- Box Model
- Margin & Padding
- Cascade & Specificity
- Display
- Pseudo-classes
- Flexbox
- Practical Project

MultiMedia?

```
<audio controls>
```

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

```
</audio>
```

```
<video controls>
```

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

```
</video>
```

Iframe?

```
<iframe src="https://example.com"></iframe>
```

Common uses:

- Maps
- Videos
- External documents

Consider security, privacy and performance.

Remember HTML

HTML describes the structure of the page.

HTML answers:

What is this element?

Examples

- Heading, Paragraph, Button
- Image, Form, Navigation

HTML

```
<h1>Mohamed Saied</h1>
```

```
<p>Frontend Developer</p>
```

```
<button>Contact Me</button>
```

But HTML Alone Is Not Enough

The browser gives us the default appearance.

Problems

- Limited visual design
- Poor spacing
- Default colors
- Default typography
- Difficult to create professional layouts

Question

How do we control the appearance?

Pure HTML

```
<h1>My Profile</h1>
```

```
<p>Frontend Developer</p>
```

```
<button>Contact Me</button>
```

Enter CSS

CSS allows us to control:

- Colors, Fonts, Sizes
- Spacing, Borders, Backgrounds
- Positioning and Layout
- Responsive behavior
- Visual states

HTML

|

v

Structure

CSS

|

v

Presentation + Layout

Why Separate HTML and CSS?

What happens if we want to change blue to red? We have to modify many elements.

Instead: one rule

Separation of concerns

HTML → Structure

CSS → Presentation

Repeated inline styles

```
<h1 style="color: blue;">Hello</h1>  
<h2 style="color: blue;">Welcome</h2>  
<p style="color: blue;">Frontend</p>
```

One CSS rule

```
h1, h2, p {  
    color: blue;  
}
```

CSS Basic Syntax

A CSS rule contains:

Three important parts

- p → Selector
- color → Property
- blue → Value

Pattern

```
selector {  
    property: value;  
}
```

Example

```
p {  
    color: blue;  
    font-size: 20px;  
}
```

Understanding a CSS Declaration

color: blue; is a declaration.

- color → Property
- blue → Value

Multiple declarations

Each declaration ends with a semicolon.

```
p {  
    color: blue;  
    font-size: 20px;  
    font-weight: bold;  
}
```

Chapter

02

Your First CSS Demo

index.html

```
<h1>Hello CSS</h1>
<p>My first styled paragraph.</p>
```

style.css

```
h1 {
    color: blue;
}

p {
    color: green;
}
```

Live Demo

Open:

- index.html
- style.css

Link them together.

Three Ways to Add CSS

1. Inline CSS

```
<p style="color: red;">Hello</p>
```

2. Internal CSS

```
<style>  
  p {  
    color: red;  
  }  
</style>
```

3. External CSS

```
<link rel="stylesheet"  
      href="style.css">
```

Inline CSS

```
<h1 style="color: red;">  
  Hello  
</h1>
```

Advantages

- Quick
- Useful for testing

Disadvantages

- Difficult to maintain
- Repeated code
- Mixes HTML and CSS
- Poor scalability

Professional projects

Avoid inline CSS for normal styling.

Internal CSS

```
<head>

<style>
  h1 {
    color: blue;
  }

  p {
    color: gray;
  }
</style>

</head>
```

Useful for

- Small pages
- Quick demonstrations
- Single-page experiments

But larger projects normally use external CSS.

External CSS

HTML

```
<head>
  <link rel="stylesheet"
        href="style.css">
</head>
```

style.css

```
h1 {
    color: blue;
}

p {
    color: gray;
}
```

Why external CSS?

- Reusable
- Maintainable
- Cleaner HTML
- Shared by multiple pages
- Better organization

CSS Comments

```
/* This is a comment */  
  
p {  
    color: blue;  
}  
  
/* Main page heading */  
h1 {  
    color: navy;  
}
```

Comments are ignored by the browser.
Use them to explain your code.

CSS Selectors

A selector tells CSS:

Which HTML elements should receive these styles?

Selectors are one of the most important CSS concepts.

Examples

```
p { }
```

```
h1 { }
```

```
button { }
```

```
.profile { }
```

```
#header { }
```

Element Selector

Select by HTML element.

All matching `<p>` elements are selected.

CSS

```
p {  
    color: gray;  
}
```

Affects

```
<p>First</p>  
<p>Second</p>  
<p>Third</p>
```

Class Selector

HTML

```
<p class="important">  
    Important information  
</p>
```

CSS

```
.important {  
    color: red;  
}
```

Syntax

.class-name

A class can be reused on many elements.

ID Selector

HTML

```
<header id="main-header">
```

CSS

```
#main-header {  
    background: black;  
}
```

Syntax

#id-name

Important

An id identifies a particular element. A class is normally reusable.

Class vs ID

Class (.card)

- Reusable
- Many elements
- Common for styling

ID (#main-card)

- Intended to identify one element
- Normally unique
- Less commonly needed for styling

Example

```
<div class="card"></div>  
<div class="card"></div>  
<div class="card"></div>
```

Universal Selector

```
* {  
  margin: 0;  
  padding: 0;  
}
```

* selects all elements.

Useful for

- Resetting default spacing
- Global rules

Use it carefully because it affects everything.

Grouping Selectors

Instead of

```
h1 {  
    color: blue;  
}  
  
h2 {  
    color: blue;  
}  
  
p {  
    color: blue;  
}
```

We can write

```
h1, h2, p {  
    color: blue;  
}
```

One rule, multiple selectors.

Descendant Selector

HTML

```
<div class="card">  
  <h2>Profile</h2>  
  <p>Hello</p>  
</div>
```

CSS

```
.card p {  
  color: gray;  
}
```

Meaning:

Select `<p>` elements inside `.card`.

Child Selector

```
.card > p {  
  color: red;  
}
```

> means: Direct child

Compare

.card p

with

.card > p

This distinction becomes important in larger applications.

Attribute Selectors

HTML

```
<input type="text">  
<input type="email">  
<input type="password">
```

CSS

```
input[type="text"] {  
    border: 2px solid blue;  
}
```

We can select elements based on attributes.

Chapter

03

Selector Practice

HTML

```
<h1 class="title">Profile</h1>
```

```
<p class="description">  
  Frontend Developer  
</p>
```

```
<button id="contact">Contact</button>
```

Which selector targets:

1. The `<h1>`?
2. The description class?
3. The `#contact` button?
4. All paragraphs?

Answers

1. `h1`
2. `.description`
3. `#contact`
4. `p`

CSS Colors

CSS supports different color formats.

Named color
`color: red;`

HEX
`color: #ff0000;`

RGB
`color: rgb(255, 0, 0);`

RGBA
`color: rgba(255, 0, 0, 0.5);`

HEX Colors

Format

#RRGGBB

#FF0000

^ ^ ^

R G B

Examples

#FF0000 → Red

#00FF00 → Green

#0000FF → Blue

#000000 → Black

#FFFFFF → White

Example:

color: #3498db;

Backgrounds

```
body {  
  background-color: #f5f5f5;  
}  
  
.card {  
  background-color: white;  
}  
  
.hero {  
  background-image: url("hero.jpg");  
}
```

Three cases

- Page background
- Element background
- Background image

Text Styling

```
p {  
  color: #333;  
  font-size: 18px;  
  font-family: Arial, sans-serif;  
  font-weight: bold;  
  text-align: center;  
}
```

Common properties

- color
- font-size
- font-family
- font-weight
- font-style
- text-align
- text-decoration
- line-height

Font Size Units

```
font-size: 20px;
```

Common units

- px → fixed CSS pixel unit
- % → relative to another value
- em → relative to the element's font context
- rem → relative to the root (html) font size

For now

px = fixed-size, rem = root-relative, % = percentage

Text Alignment & Decoration

CSS

```
h1 {  
    text-align: center;  
}  
  
a {  
    text-decoration: none;  
}
```

text-align values

left, center, right, justify

text-decoration

- none → removes underline
- underline → adds underline

Borders

CSS

```
.card {  
    border: 2px solid black;  
}
```

```
border: 2px solid #333;
```

A border consists of

- width
- style
- color

Other styles

solid, dashed, dotted, double

Border Radius

CSS

```
.card {  
    border-radius: 10px;  
}  
  
.profile-image {  
    border-radius: 50%;  
}
```

Creates rounded corners.

50% makes a circle.

Demo

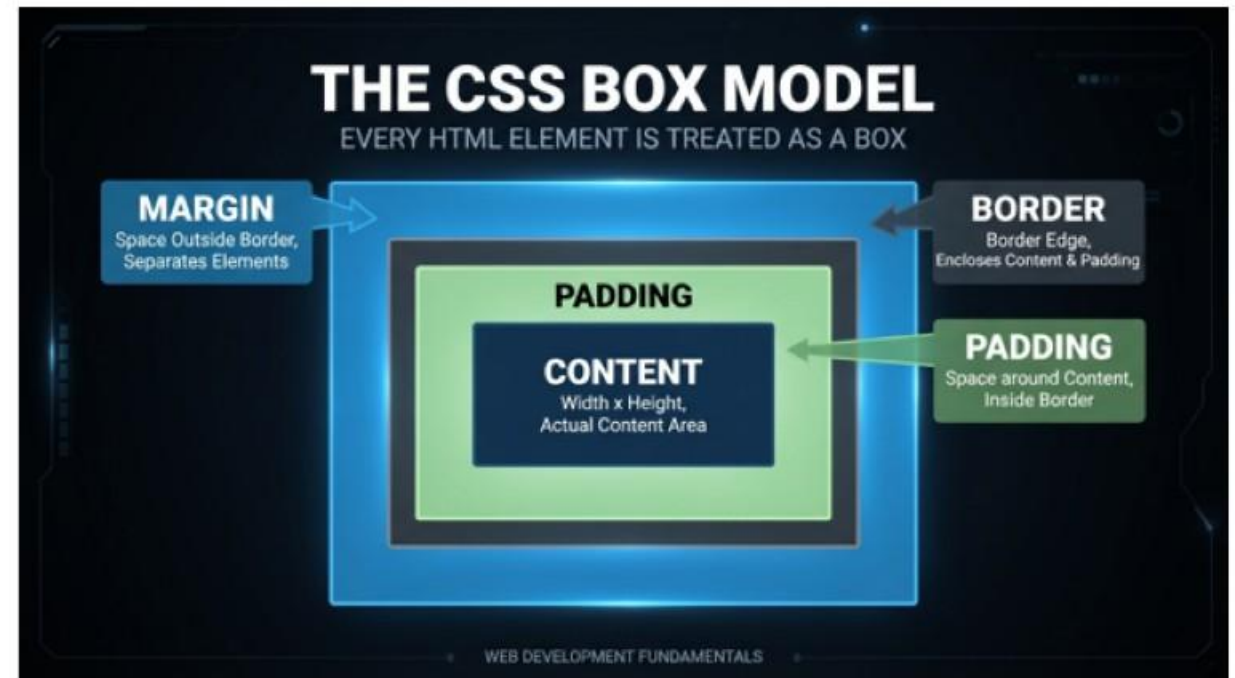
Turn a square image into a circular profile picture.

The Most Important Concept: Box Model

Every HTML element is treated as a box.

Four parts

1. Content
2. Padding
3. Border
4. Margin



Content

```
.card {  
  width: 300px;  
  height: 200px;  
}
```

The content is the actual information:

- Text
- Image
- Button
- Form
- ...

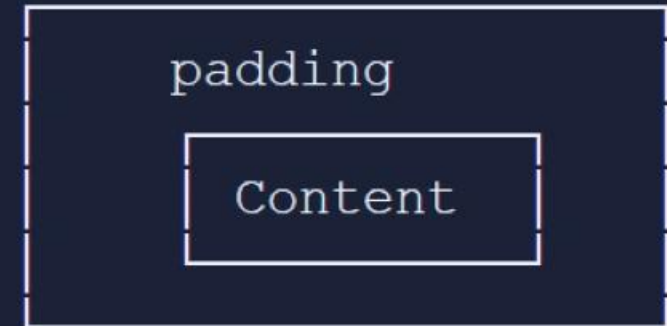
Padding

Padding is the space inside the element, between content and border.

CSS

```
.card {  
    padding: 20px;  
}
```

Visual idea



Margin

Margin is the space outside the element.

CSS

```
.card {  
  margin: 20px;  
}
```

Visual idea

Card A



margin

Card B



Padding vs Margin

Padding: space inside

```
.card {  
  padding: 20px;  
}
```

Margin: space outside

```
.card {  
  margin: 20px;  
}
```

Remember

Padding pushes the content inward.

Margin separates elements from each other.

Padding Shorthand

CSS

```
padding: 20px;
```

```
padding: 10px 20px 30px 40px;
```

padding: 20px applies to all sides.

Four values

Top, Right, Bottom, Left

Clockwise direction

Margin Shorthand

CSS

```
margin: 10px 20px;
```

```
margin: 0 auto;
```

margin: 10px 20px means:

- Top/Bottom → 10px
- Left/Right → 20px

margin: 0 auto

Often used to horizontally center a fixed-width block.

Chapter

05

Width + Padding Problem

CSS

```
.card {  
  width: 300px;  
  padding: 20px;  
  border: 5px solid black;  
}
```

Default box-sizing

```
300  
+ 40 padding  
+ 10 border  
-----  
350px
```

The actual outer width becomes 350px.

This is an important real-world CSS problem.

box-sizing

A common global rule

```
* {  
    box-sizing: border-box;  
}
```

With border-box, the declared width includes content + padding + border.

Result

The outer width of the 300px card stays 300px.

CSS Cascade

Why is CSS called Cascading Style Sheets?

Because multiple rules can apply to the same element.

Which color wins?

red

The cascade determines which declaration is applied.

```
p {  
    color: blue;  
}  
  
p {  
    color: red;  
}
```

Specificity

CSS

```
p {  
    color: blue;  
}  
  
.text {  
    color: green;  
}  
  
#main-text {  
    color: red;  
}
```

HTML

```
<p id="main-text" class="text">  
    Hello  
</p>
```

All three selectors match, but the ID selector has higher specificity.

Result

red

Specificity Concept

A useful mental model

Element

|

v

Class / Attribute / Pseudo-class

|

v

ID

Generally

ID > Class > Element

Do not solve every CSS problem by adding !important.

!important

```
p {  
    color: red !important;  
}
```

It can override normal cascade rules. But avoid unnecessary use.

Why?

- Makes CSS harder to debug
- Creates specificity problems
- Makes future overrides harder

Professional rule

Understand the cascade instead of fighting it.

CSS Inheritance

CSS

```
body {  
    color: #333;  
}
```

HTML

```
<body>  
    <p>Hello</p>  
</body>
```

Some properties can be inherited.
The paragraph may receive the
inherited text color.

display

The display property controls how an element participates in layout.

This is the bridge between basic CSS and layout.

Common values

```
display: block;  
display: inline;  
display: inline-block;  
display: none;  
display: flex;
```

Block Elements

Examples: `<div>`, `<p>`, `<h1>`, `<section>`

Block elements normally start on a new line.

Element 1

Element 2

Inline Elements

Examples: `<span>`, `<a>`,
`<strong>`

Inline elements normally stay in
the same line.

They occupy only the space
needed by their content.

```
Hello world and link
```

Inline-Block

```
.card {  
  display: inline-block;  
}
```

Combines useful characteristics:

- Appears inline
- Can have width
- Can have height
- Can have padding and margin

Useful before students learn modern layout systems.

display: none

```
.menu {  
    display: none;  
}
```

The element is removed from the layout.

Compare with visibility: hidden

- display:none → not displayed, no layout space
- visibility:hidden → invisible but space remains

Pseudo-Classes

```
button:hover {  
    background-color: blue;  
}
```

Pseudo-classes describe a state. The style applies when the mouse is over the button.

Common pseudo-classes

:hover, :focus, :active, :checked, :first-child, :last-child

:hover**HTML**

```
<button>Contact Me</button>
```

CSS

```
button {  
    background: black;  
    color: white;  
}  
  
button:hover {  
    background: blue;  
}
```

Live Demo

Move the mouse over the button.

CSS can react to user interaction without JavaScript.

:focus

```
input:focus {  
  border-color: blue;  
  outline: none;  
}
```

Useful for forms. When the user clicks or tabs into the input, the focus style appears.

Important

Focus styles are important for keyboard accessibility. Do not blindly remove focus indicators.

Styling the Lecture 1 Form

Take the registration form from Lecture 1 and style it.

Live coding exercise

Students style their own form.

After CSS

Registration

Name

[_____]

Email

[_____]

Password

[_____]

[Register]

Introduction to Flexbox

Modern CSS layout becomes much easier with Flexbox.

Flexbox is designed for arranging elements along an axis.

```
.container {  
    display: flex;  
}
```

Flex Container & Flex Items

HTML

```
<div class="container">  
  <div>One</div>  
  <div>Two</div>  
  <div>Three</div>  
</div>
```

CSS

```
.container {  
  display: flex;  
}
```

The parent becomes the Flex Container.

The children become Flex Items.

Main Axis

By default:

`flex-direction: row;`

Items are arranged horizontally.

Main Axis →

1

2

3

flex-direction

```
Horizontal:  1  2  3  
.container {  
  display: flex;  
  flex-direction: row;  
}
```

```
Vertical:  1 / 2 / 3  
.container {  
  display: flex;  
  flex-direction: column;  
}
```

justify-content

Controls alignment along the main axis.

Common values

- flex-start
- center
- flex-end
- space-between
- space-around
- space-evenly

```
.container {  
  display: flex;  
  justify-content: center;  
}
```

Chapter

09

justify-content Demo

CSS

```
.container {  
  display: flex;  
  justify-content: space-between;  
}
```

Result



```
[1] [2] [3]
```

Students should see the spacing change live.

align-items

```
.container {  
  display: flex;  
  align-items: center;  
}
```

Controls alignment along the cross axis.

Common values

stretch, flex-start, center, flex-end

Key concept

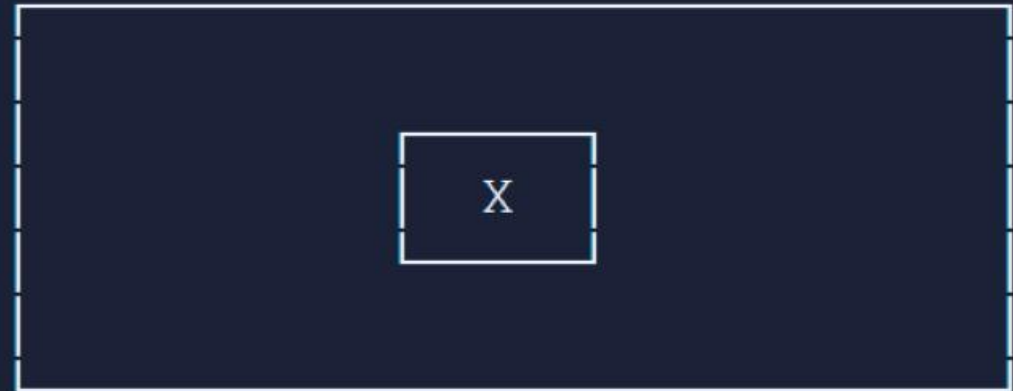
- justify-content → main axis
- align-items → cross axis

Centering with Flexbox

CSS

```
.container {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
}
```

Result



Instead of manually adding margins, use gap.
gap is extremely useful for modern layouts.

CSS

```
.container {  
  display: flex;  
  gap: 20px;  
}
```

Result

```
[ Card ] 20px [ Card ] 20px [ Card ]
```

Flexbox Navigation Demo

HTML

```
<nav>
  <a href="#">Home</a>
  <a href="#">About</a>
  <a href="#">Projects</a>
  <a href="#">Contact</a>
</nav>
```

CSS

```
nav {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

Goal

Build a simple professional navigation bar.

Flexbox Cards Demo

HTML

```
<div class="cards">

  <div class="card">
    <h3>HTML</h3>
    <p>Structure</p>
  </div>

  <div class="card">
    <h3>CSS</h3>
    <p>Presentation</p>
  </div>

  <div class="card">
    <h3>JavaScript</h3>
    <p>Behavior</p>
  </div>

</div>
```

CSS

```
.cards {
  display: flex;
  gap: 20px;
}
```

Flex Wrapping

What happens when there isn't enough horizontal space?
Items can move to another line.
This introduces the idea of responsive layouts.

CSS

```
.cards {  
  display: flex;  
  flex-wrap: wrap;  
}
```

Result

```
[Card] [Card] [Card]  
  
[Card] [Card]
```

Flex Item Growth

```
CSS
.card {
  flex: 1;
}
```

```
Result
[ Card 1 ][ Card 2 ][ Card 3 ]
```

Available space can be distributed among flex items.

CSS Debugging

Something looks wrong? Don't randomly change CSS.

Use Browser DevTools

Open with:

- Right Click → Inspect
- or press F12

```
HTML structure
  |
  v
Selected element
  |
  v
Styles
  |
  v
Box Model
```

DevTools Elements Panel

Students should identify the HTML structure, the selected element, its styles and the box model.

DevTools allows us to

- Inspect elements
- See applied CSS
- See overridden CSS
- Modify CSS live
- Inspect dimensions
- Test styles

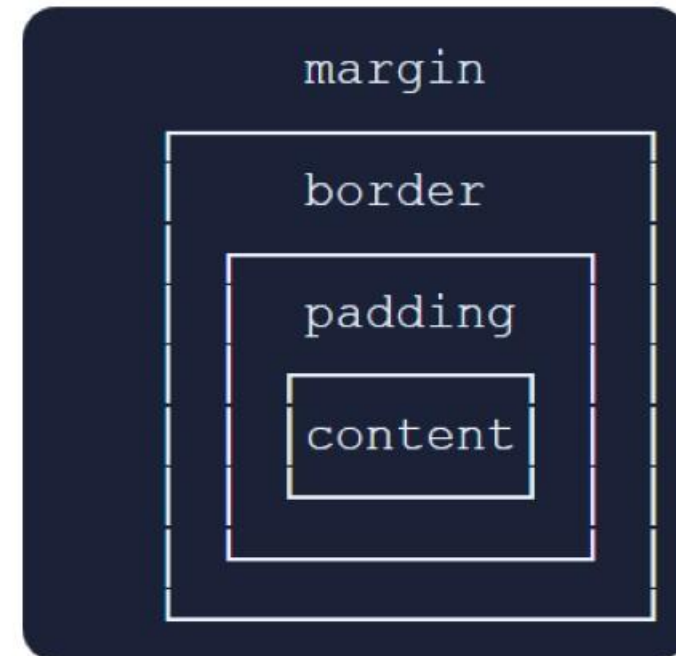
The Box Model in DevTools

When selecting an element, DevTools shows the box model diagram.

Live Demo

Select a card and change:

- margin
- padding
- width, height
- border



Common CSS Mistakes

1. Wrong CSS file (file is style.css)

```
<link rel="stylesheet" href="styles.css">
```

2. Missing semicolon

```
p {  
  color: red  
  font-size: 20px;  
}
```

3. Wrong selector

```
.card-title {  
  color: red;  
}  
  
<h2 class="title">
```

CSS Workflow

A professional frontend workflow:

1. Build HTML structure
2. Give important elements classes
3. Create CSS
4. Style typography/colors
5. Apply spacing
6. Build layout
7. Test interaction states
8. Inspect with DevTools
9. Refactor CSS

Mini Project: Transform the Lecture 1 Profile

Students already have an HTML Profile.

Now transform it into a styled page with navigation, hero, skills and a contact button.

Home	About	Skills	Contact
------	-------	--------	---------

[Profile Image]

Mohamed Saied
Frontend Developer

[Contact Me]

Skills

[HTML] [CSS] [JavaScript]

Project Step 1: Semantic HTML

```
<header>
  <nav>
    ...
  </nav>
</header>

<main>
  <section>
    ...
  </section>

  <section>
    ...
  </section>
</main>

<footer>
  ...
</footer>
```

Principle

CSS should style a good HTML structure.

CSS does not replace semantic HTML.

Project Step 2: Global CSS

```
* {  
    box-sizing: border-box;  
}  
  
body {  
    margin: 0;  
    font-family: Arial, sans-serif;  
    color: #333;  
}
```

Explain each line.

Project Step 3: Navigation

Layout

```
nav {  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
  padding: 20px 40px;  
}
```

Links

```
nav a {  
  text-decoration: none;  
}  
  
nav a:hover {  
  text-decoration: underline;  
}
```

Project Step 4: Profile Section

Section

```
.profile {  
  text-align: center;  
  padding: 60px 20px;  
}
```

Profile image

```
.profile img {  
  width: 150px;  
  height: 150px;  
  border-radius: 50%;  
}
```

Project Step 5: Skills

HTML

```
<div class="skills">  
  
  <div class="skill">HTML</div>  
  <div class="skill">CSS</div>  
  <div class="skill">JavaScript</div>  
  
</div>
```

CSS

```
.skills {  
  display: flex;  
  justify-content: center;  
  gap: 20px;  
}
```

Project Step 6: Skill Cards

```
.skill {  
  padding: 20px;  
  border: 1px solid #ddd;  
  border-radius: 10px;  
  background: white;  
}  
  
.skill:hover {  
  transform: translateY(-3px);  
}
```

This gives students their first introduction to the idea that CSS can create interaction effects.

Project Step 7: Contact Button

```
.contact-btn {  
  display: inline-block;  
  padding: 12px 24px;  
  border-radius: 8px;  
  text-decoration: none;  
  background: #222;  
  color: white;  
}  
  
.contact-btn:hover {  
  background: #444;  
}
```

CSS Architecture

As projects become larger, don't write random CSS everywhere.

Think in components:

- Navbar, Hero, Card
- Button, Form, Footer

Prepares students for

Bootstrap, Angular, React, component-based UI

```
.card { }  
  
.card-title { }  
  
.card-description { }  
  
.card-button { }
```

What We Have Learned

CSS Fundamentals

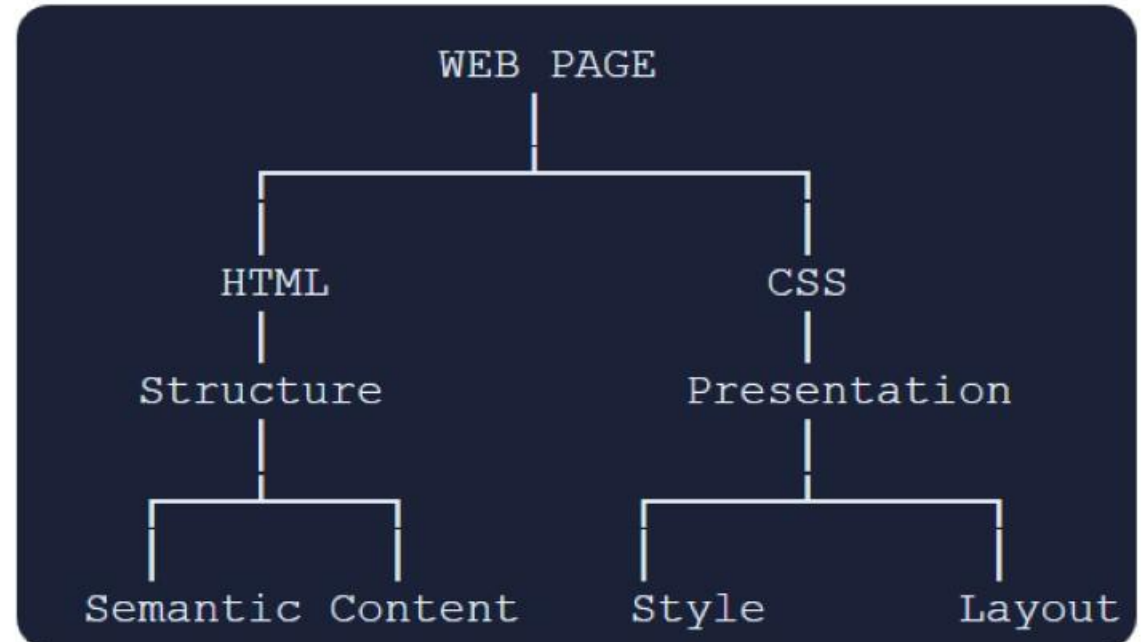
- CSS syntax
- External CSS
- Selectors, Classes, IDs
- Colors, Typography
- Borders, Backgrounds
- Box Model, Margin, Padding

Continued

- box-sizing
- Cascade, Specificity
- Inheritance
- Display
- Pseudo-classes
- Flexbox
- DevTools

HTML + CSS Mental Model

Students should leave with this model of a web page.



Important Rules to Remember

1. HTML describes what something is.
2. CSS describes how it looks.
3. Use classes for reusable styling.
4. Understand the Box Model.
5. Understand the Cascade before using !important.
6. Use Flexbox for one-dimensional layouts.
7. Use DevTools instead of guessing.

Quick Quiz 1

What does CSS stand for?

- A. Computer Style Sheets
- B. Cascading Style Sheets
- C. Creative Style Syntax
- D. Colorful Style Sheets

Answer: B

Quick Quiz 2

Which selector selects a class?

- A. #card
- B. .card
- C. card
- D. *card

Answer: B

Quick Quiz 3

Which property creates space inside an element?

- A. margin
- B. padding
- C. border
- D. spacing

Answer: B

Quick Quiz 4

Which property enables Flexbox?

- A. `position: flex;`
- B. `layout: flex;`
- C. `display: flex;`
- D. `flex: display;`

Answer: C

Quick Quiz 5

What does this do?

- A. Changes the button when clicked permanently
- B. Changes the button when the pointer is over it
- C. Hides the button
- D. Changes the HTML

Answer: B

```
button:hover {  
    background: blue;  
}
```

Final Challenge: Build Your Own Profile Card

Profile Image

Student Name

Frontend Student

HTML CSS JavaScript

[Contact]

Requirements

- External CSS
- Class selectors
- Colors, Typography
- Padding, Margin
- Border, Border radius
- Flexbox
- :hover

Challenge: No Copy/Paste

Students should build it themselves.

Goal

Don't memorize CSS properties.

Understand why you are using them.

HTML → Structure



CSS → Typography



Colors



Box Model



Layout



Interaction

Lecture 2 Summary

Part 1

HTML



CSS Syntax



Selectors



Colors & Typography



Box Model

Part 2

Display



Pseudo-classes



Flexbox



DevTools



Real Project

Chapter

12

Next Lecture: CSS Layout & Responsive Design

Lecture 3

CSS Layout & Responsive Design

Coming next

- Advanced Flexbox
- CSS Grid
- Responsive Design
- Media Queries
- Bootstrap5
- Script +DOM



Thank You