What the Browser Draws

HTML for what things are, CSS for how they look

E190BO · AI-Amplified Prototyping

Five big ideas

1. **HTML is a tree.** The browser turns your file into the DOM, and the DOM is what's on screen.
2. **Elements say what things are:** a heading, a list, a form, a button. Keyboards and screen readers depend on it.
3. **A CSS rule picks elements, then styles them.**
4. **When two rules disagree, the more specific one wins,** whatever their order in the file.
5. **Everything is a box,** and a row of boxes can wrap to fit a phone.

1 · HTML is a tree

A whole page

html-css-1-tree/index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Idea board</title>
</head>
<body>
  <h1>Idea board</h1>
  <ul>
    <li>Automatic plant waterer, 2 votes</li>
    <li>Sticker vending machine, 3 votes</li>
    <li>Bike lock that texts you, 0 votes</li>
  </ul>
</body>
</html>
```

The browser turns it into a tree

```
html
├── head
│   ├── meta
│   ├── meta
│   └── title  "Idea board"
└── body
    ├── h1  "Idea board"
    └── ul
        ├── li  "Automatic plant…"
        ├── li  "Sticker vending…"
        └── li  "Bike lock that…"
```

- Elements nest, so a page is a **tree**: every element has one parent.
- The browser reads the file once and builds the tree in memory. That tree is the **DOM** (Document Object Model).
- **The DOM is what's on screen, not the file.** Change the DOM and the page changes.

▶ LIVE Open it, and change the tree

1. Open `index.html` in Chrome, straight from the folder.
2. Right-click the first idea → **Inspect**. The **Elements** panel is the tree.
3. Double-click `2 votes` in the Elements panel, type `5 votes`, Enter. The page says 5.
4. **View page source** (Ctrl-U, or Cmd-Option-U on a Mac): the file still says 2.
5. Reload: the DOM is rebuilt from the file, and it's 2 again.

2 · Elements say what things are

Headings, a form, and real buttons

html-css-2-semantic/index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Idea board</title>
</head>
<body>
  <h1>Idea board</h1>

  <h2>Add an idea</h2>
  <form>
    <label for="title">Title</label>
    <input id="title" name="title" required>
    <button>Add idea</button>
  </form>

  <h2>Ideas</h2>
  <ul>
    <li>Automatic plant waterer, 2 votes <button type="button">Vote</button></li>
    <li>Sticker vending machine, 3 votes <button type="button">Vote</button></li>
    <li>Bike lock that texts you, 0 votes <button type="button">Vote</button></li>
  </ul>
</body>
</html>
```

What each element is

Element	What it is
`<h1>`, `<h2>`	a heading: the page's title, then a section's
`<form>`	inputs sent together; by default, to a new page
`<label for="title">`	the name of the input with `id="title"`
`<input>`	a box to type in
`<button>`	something to press
`<ul>`, `<li>`	a list, and one item in it

Use the element that says what a thing *is*, and the browser, the keyboard and screen readers know what to do with it. A `<div>` says nothing about what it is.

3 · A CSS rule picks elements, then styles them

Give CSS something to grab

In the page's `<head>`, one line pulls in the stylesheet:

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

and each idea gets classes for its parts:

```
<li class="idea">
  <span class="idea-title">Automatic plant waterer</span>
  <span class="votes">2 votes <button type="button">Vote</button></span>
</li>
```

demos/html-css-3-selectors/index.html

A stylesheet: selectors and declarations

html-css-3-selectors/style.css

```
body {                          /* an element: every <body> */
  font-family: system-ui, sans-serif;
  color: #1f2937;
}

h1 {
  color: #c2410c;
}

.votes {                        /* a class: every class="votes" */
  color: #6b7280;
}

#title {                        /* an id: the one id="title" */
  border: 2px solid #c2410c;
}
```

- A **rule** is a **selector** and a block of **declarations**, `property: value;`.

The selectors you'll use

Selector	Picks	Example
`button`	every element of that type	every button on the page
`.votes`	every element with that class	each idea's vote count
`#title`	the one element with that id	the title input
`.idea button`	a `button` *somewhere inside* a `.idea`	each Vote button, not Add idea
`button:disabled`	a button, but only while it's disabled	a button you've switched off mid-request
`[role="status"]`	anything with that attribute	a status message

▶ LIVE Find a rule from the page

1. Open `index.html` in Chrome.
2. Inspect `2 votes`. **Elements → Styles** shows the `.votes` rule, with its file and line.
3. Further down, *Inherited from body*: the font comes from `body`.
4. Click the colour value in `.votes` and pick another: the page changes. Reload: it's back.

4 · When rules disagree, the more specific wins

Two rules, one Vote button

```css
.idea button {                  /* a class, then an element inside it */
  color: #c2410c;
  background: white;
  border: 1px solid #c2410c;
}

button {                        /* every button, Add and Vote alike */
  color: white;
  background: #c2410c;
  border: none;
  padding: 0.4rem 0.9rem;
}
```

- Both rules match a Vote button, and they disagree about `color`, `background` and `border`.
- Only `button` matches Add idea.
- `button` comes **later** in the file. Which one wins for a Vote button?

demos/html-css-4-specificity/style.css, the last two rules

Specificity first; order only breaks a tie

Selector has…	Example	Outranks
an id	`#title`	any number of classes
a class, attribute or pseudo-class	`.idea button`, `[role="status"]`, `:disabled`	any number of element names
only element names	`button`	— (lowest)

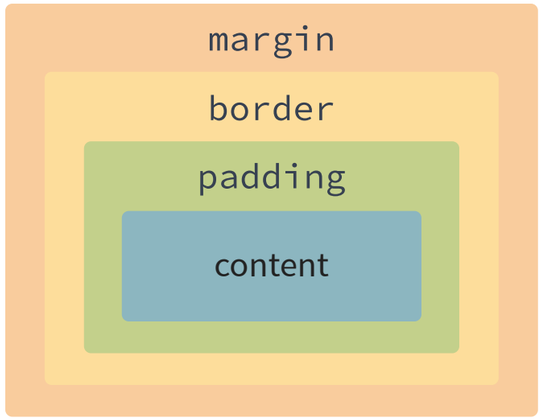
- `.idea button` has a class, and `button` doesn't, so **`.idea button` wins**, even though it comes first.
- **Order** decides only between rules that are equally specific: then the later one wins.
- A value **inherited** from a parent loses to *any* rule that matches the element itself.
- Above all of these: a `style="…"` attribute on the element, then `!important`, which is usually a sign you're fighting yourself.

▶ LIVE Watch the loser get crossed out

1. Open `index.html` in Chrome: Add idea is solid; each Vote button is outlined.
2. Inspect a Vote button. **Styles**: `.idea button` on top; in `button`, `color`, `background` and `border` are ~~crossed out~~, and `padding` isn't.
3. In the editor, move `.idea button` below `button`, save, and reload. **Nothing changes.**

5 · Boxes, and rows that wrap

Every element is a box



- **padding** is space inside the border; **margin** is space outside it.
- `max-width`, not `width`: on a phone, the column is as wide as the phone.

```
body {
  font-family: system-ui, sans-serif;
  color: #1f2937;
  max-width: 42rem;            /* a column that stops growing */
  margin: 0 auto;              /* auto left and right: centred */
  padding: 1rem;
}
```

demos/html-css-5-layout/style.css, the first rule

Cards that wrap instead of overflowing

```css
.ideas {
  display: flex;                  /* children in a row */
  flex-wrap: wrap;                /* ...that wraps when it runs out of room */
  gap: 1rem;
  padding: 0;
  list-style: none;
}

.idea {
  flex: 1 0 10rem;                /* at least 10rem wide; share the spare room */
  padding: 1rem;
  border: 1px solid #e5e7eb;
}
```

- `display: flex` puts the cards in a row, `gap` apart; each is at least 10rem wide, and they share the room left over.
- `flex-wrap: wrap`: a card that doesn't fit starts a new row. Three across on a laptop, one on a phone, and **no media query**.

demos/html-css-5-layout/style.css, the last two rules; the list is now `<ul class="ideas">`

▶ LIVE Phone width

1. Open `index.html` in Chrome: a centred column, three cards across.
2. Turn on the **device toolbar** (Ctrl-Shift-M, or Cmd-Shift-M on a Mac), and drag the width down slowly to **375**. Three across, then two, then one; nothing scrolls sideways.
3. Inspect the list; in **Styles**, untick `flex-wrap` on `.ideas`: the cards stay in one row and **run off the right edge**. Tick it again.
4. Delete the `viewport` line from `index.html`, save, reload at 375: a shrunken desktop page. Put it back.

What’s yours, and Lab 6

Lab 6 asks for	Chapter
Semantic, accessible HTML: one `<h1>`, headings in order, a label on every control, real `<button>`s	1, 2
Works at phone width: nothing overflows sideways at 375px	5
Check-in, **CSS:** “Two rules are fighting over this element. Which wins, and why?”	4

The tests in Lab 6 read your HTML for the first row. The second and third are you, with the device toolbar and the Styles pane open.

Next: this page, served by your API, with real ideas in it.