Web Coding Camp — Day 1
HTML: Structure & Content

David Pouliot
Southern Oregon University

Welcome to Web Camp

This week you build **one website** — your idea, live on the internet by Friday.

- **Day 1** — HTML: the structure and content
- **Day 2** — CSS: colours, fonts, and spacing
- **Day 3** — Layout: nav bars and columns
- **Day 4** — JavaScript: make it interactive
- **Day 5** — Publish it and show it off

No experience needed. If you can type and save a file, you can do this.

About Me

- 6 years teaching CS at SOU
- Research Interests:
  - Applied Cryptography, Searchable Encryption
  - Symbolic Execution
  - Practical Security Solutions
- Hobbies are furniture building and whitewater kayaking

About You

- Your Name
- Something about you (Optional, choose zero or more)
  - Dream job
  - Interests
  - Favorite food
  - Hobbies
  - ?

Today

- What a webpage really is
- Tags and elements — the building blocks
- Headings and paragraphs
- Images and links
- Lists (bulleted and numbered)
- Nesting and the page skeleton
- A first taste of CSS — adding colour

Goal for today: your theme's page skeleton — a heading, an image, a paragraph, and a list — open in a browser, with a splash of colour.

Setup

The whole workflow: edit, save, look

You'll write code in **VS Code** and preview it with **Live Server**.

1. Open **VS Code**.
2. Extensions panel (`Ctrl+Shift+X`) → search **Live Server** → Install.
3. **File** → **Open Folder** on a new folder, e.g. `my-site`.
4. Make a file `index.html`.
5. Right-click it → **Open with Live Server**.

Leave the browser tab open. Every time you save (`Ctrl+S`) it refreshes. Edit → save → look. That's the loop.

Pick your theme — now

One theme, all week. Every theme uses the same tags — only your words and pictures change.

- **Fan page** — game, band, team, show
- **Recipe** — a dish you love
- **Review site** — rate 5 movies
- **Fake business** — food truck, band
- **Event** — a party or tournament
- **Tool** — a tip calculator, quiz

Can't decide? Start a fan page — everyone's a fan of *something*.

What Is HTML?

Every website is just text

A webpage is a plain text file the browser reads and *draws*.

- **HTML** — the content and structure (today)
- **CSS** — how it looks (Day 2)
- **JavaScript** — what it does (Day 4)

Right-click any website → **View Page Source**. That's the HTML that built it. Every site on the internet starts here.

Tags come in pairs

Most tags have an opening tag and a closing tag (the one with the slash). Your content goes between them.

```
<h1>This is a big heading</h1>
<p>This is a paragraph of text.</p>
```

- <h1> opens it, </h1> closes it.
- Tag + content + closing tag = an element.

Tags go inside other tags: nesting

Putting tags *inside* tags is how you build structure.

```
<ul>
  <li>First item</li>
  <li>Second item</li>
</ul>
```

- Indent the inside tags so you can see the structure.
- Close them in the reverse order you opened them.

The Page Skeleton

Every page starts the same way

The <head> holds settings (like the tab title). The <body> holds everything you actually see.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My Page</title>
</head>
<body>
  <!-- your visible content goes here -->
</body>
</html>
```

Headings and text

```
<h1>Biggest heading</h1>     <!-- one h1 per page -->
<h2>Section heading</h2>
<h3>Smaller heading</h3>
<p>A paragraph of normal text.</p>
<strong>bold</strong> and <em>italic</em>
```

- <h1> down to <h6> — big to small.
- Use one <h1>: the page's main title.

Images, Links & Lists

Images

`<img>` has no closing tag. `src` is the file name; `alt` describes it (shown if the image can't load).

```
<img src="hero.jpg" alt="A screenshot from the game">
```

Put your image file in the *same folder* as your HTML, then just use its name in `src`. No image yet? Any `.jpg` or `.png` works.

Links

`href` is where the link goes when clicked.

```
1 <a href="https://developer.mozilla.org/">Click here</a>
```

- Link to another site: a full `https://...` address.
- Link to your own second page: just the file name, e.g. `about.html`.

Lists: bulleted and numbered

```
<ul>                    <!-- ul = unordered (bullets) -->
  <li>Apples</li>
  <li>Oranges</li>
</ul>

<ol>                    <!-- ol = ordered (numbered) -->
  <li>First step</li>
  <li>Second step</li>
</ol>
```

Each item is an `<li>`. Bullets or numbers appear automatically.

Project: Page Skeleton

Build your skeleton (worked example)

Four pieces every theme shares: a heading, an image, an intro paragraph, a list.

```
<body>
  <h1>The Ultimate Zelda Fan Page</h1>
  <img src="hero.jpg" alt="Screenshot from the game">
  <p>The Legend of Zelda is my favorite series. Here's why.</p>

  <h2>My Top 3 Games</h2>
  <ul>
    <li>Breath of the Wild</li>
    <li>Ocarina of Time</li>
  </ul>
</body>
```

Your theme fits the same shape

Theme	`<h1>`	`<img>`	`<p>`	`<ul>`
Recipe	Dish name	Photo of dish	Why you love it	Ingredients
Review	"My Top 5"	Poster of #1	What it ranks	The items
Business	Business name	Logo / photo	What you offer	Services
Event	Event name	Venue photo	The pitch	Schedule

Same tags for everyone — only the words change. Copy the example, then swap in *your* content.

A First Taste of CSS

HTML is content — CSS is style

The quick way to try CSS: a `<style>` block in your `<head>`.

```
<head>
  <title>My Page</title>
  <style>
    body { background: #f4f4f8; }
    h1   { color: purple; }
  </style>
</head>
```

Each rule picks a tag, then sets what to change. That's all CSS is.

Colour your page

Three properties get you a long way:

```
body {
  background: #f4f4f8;   /* page background */
  color: #222;           /* text colour */
  font-family: sans-serif;
}
```

Pick colours with a colour picker: htmlcolorcodes.com/color-picker — copy the `#hex` into your CSS. **Tomorrow:** classes, ids, and real layout.

Save Your Work: Git & GitHub

Why version control?

A **version control system** saves snapshots of your project as you work — so your code is never one bad edit away from being lost.

- **Undo** — jump back to any earlier snapshot that worked
- **Backup** — your files live online, not just on one computer
- **History** — see what changed, and when

Git is the tool that takes the snapshots. **GitHub** is the website that stores them online. Almost every programmer uses both.

Start on GitHub: create, then clone

Step 1 — create the repo. Make a free account at `github.com`, click **New repository**, and name it (e.g. `my-site`).

Step 2 — clone it. Copy the repo's URL, then clone to pull a copy down onto your computer:

```
git clone https://github.com/you/my-site.git
```

GitHub **requires two-factor authentication (2FA)**. Set it up (an authenticator app or SMS) when you make the account, or you can't push.

Then work locally: change, add, commit, push

Step 3 — go in and make changes. Open the folder and edit your files:

```
cd my-site                          # step into your project folder
# ...edit index.html, add images...
```

Step 4 — save a snapshot, then upload it.

```
git add .                           # gather up all your changes
git commit -m "My first web page"   # save a snapshot
git push                            # upload it to GitHub
```

Repeat steps 3–4 every time you get something working.

Wrap-Up

Stretch goals

Finished the skeleton? Keep going:

- Add a table (games with ratings, a menu with prices)
- Add a second page and link the two together
- Embed a YouTube video with an `<iframe>`
- Use a `<details>` / `<summary>` collapsible section
- Look up 3 new tags and use them

Every HTML tag, in one place: w3schools.com/tags — browse the full list, click any tag for an example you can try.