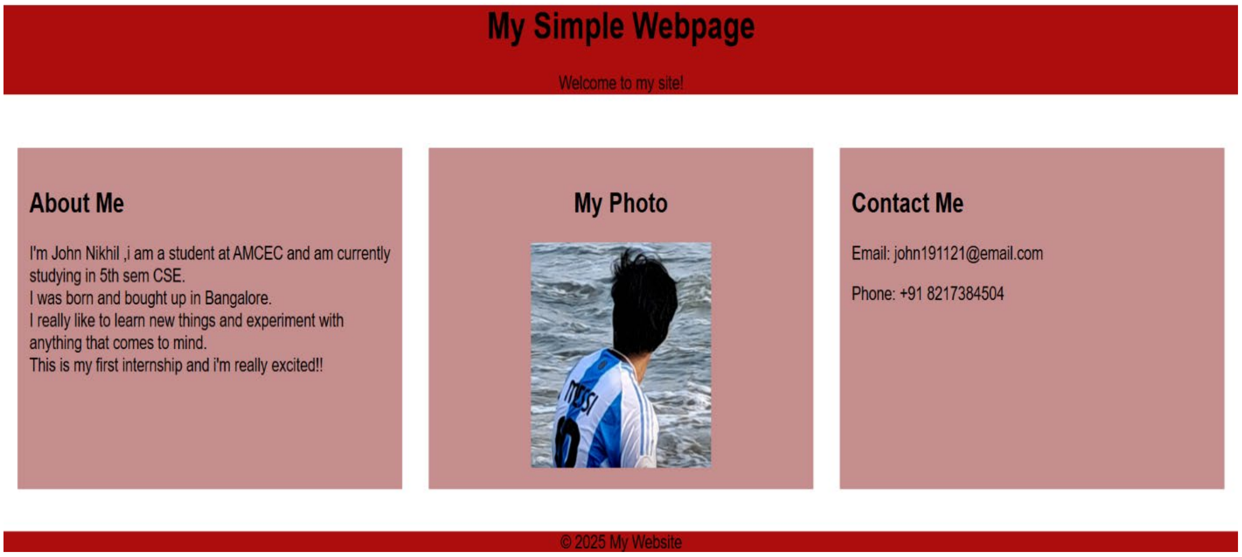
AeroAspire-SDE Intern
JOHN NIKHIL G

Week 1 – Day 2 (Sep23)

Task/Assignment :

Build basic HTML page: About / Photo / Contact sections, Style the sections; header/nav/footer;layout using Flexbox or Grid;

Flexbox Style



My Simple Webpage

Welcome to my site!

About Me

I'm John Nikhil ,i am a student at AMCEC and am currently studying in 5th sem CSE.
I was born and bought up in Bangalore.
I really like to learn new things and experiment with anything that comes to mind.
This is my first internship and i'm really excited!!

My Photo

Contact Me

Email: john191121@email.com

Phone: +91 8217384504

© 2025 My Website

Layout :

- **Flexbox:**
  - Works in one dimension at a time either a row or a column.
  - Used for managing the layout of the website

- **Grid :**
    - Works in **two dimensions** — rows and columns simultaneously.
    - For creating complex layouts like entire web page

<u>Questions/Reflection :</u>

1. <div> vs <section>

- <div>: Generic container, no required attributes, used for styling or grouping.
- <section>: Semantic container, represents a meaningful block (About, Services, etc.),supports global and event attributes.

---

2. Why semantics matter

- Adds meaning to code → easier for screen readers and search engines.
- Improves accessibility → better for users with assistive tech.
- Search engines understand content structure.
- Cleaner code → easier for developers to maintain.

3.What is the flow from writing HTML → rendering by browser?

- Browser downloads HTML.
- Builds DOM (structure).
- Loads CSS → creates CSSOM (styles).
- Combines DOM + CSSOM → render tree.
- Calculates layout (sizes, positions).
- Paints final output on screen.

---

4. How semantics help accessibility + SEO

- Clear structure for screen readers → smoother navigation.
- Headings/sections/articles highlight importance.
- Search engines index content more accurately.
- Users + bots both understand the page better.

---

5. Browser parsing (HTML + CSS)

- HTML → DOM tree (structure).
- CSS → CSSOM tree (styles).
- Combine both → render tree.
- Compute layout → element sizes/positions.
- Paint → draw text, images, colors on screen.

---

6. Flexbox resizing behavior

- Auto-adjusts items when container changes size.
- Uses flex-grow, flex-shrink, flex-basis for scaling.
- Supports wrapping for smaller screens.
- Alignment (justify-content, align-items) keeps layout balanced.

---

7. CSS box model

- Content → main text/image area.
- Padding → space inside around content.
- Border → surrounds padding.
- Margin → space outside between elements.
- Total size = content + padding + border + margin**.**

---

8. CSS specificity order

1. Inline styles → highest priority.
2. IDs → stronger than classes.
3. Classes, attributes, pseudo-classes.
4. Elements, pseudo-elements → lowest.

- If same weight → last rule in file applies**.**

---

9. Responsive layout approach

- Use relative units (%, em, rem, vw/vh).
- Make images/videos fluid (max-width: 100%).
- Apply media queries for different screen sizes.
- Mobile-first → scale up for larger screens.
- Flexbox/Grid → create adaptive layouts.