📄 ADVANCED HTML5 & CSS3 – THEORY ANSWERS

Topic: Advanced HTML5 & CSS3
Name: Sejal Singh
Submission Date: 27 February 2026

SECTION A — ADVANCED HTML THEORY

---

1. Explain Semantic vs Non-Semantic HTML Elements with Layout Examples

Semantic HTML elements clearly define the purpose and meaning of the content they contain. These elements provide structural meaning to browsers, search engines, and assistive technologies.

Examples of semantic elements include:
`<header>`, `<nav>`, `<main>`, `<section>`, `<article>`, `<aside>`, and `<footer>`.

Non-semantic elements such as `<div>` and `<span>` do not describe content meaning. They are generic containers used mainly for styling or grouping content.

Why Semantic HTML Matters

- Improves accessibility for screen readers
- Enhances SEO by defining content hierarchy
- Improves maintainability
- Reduces reliance on ARIA roles

Example Layout

```
<header>
 <nav>Navigation Links</nav>
</header>


<main>
 <section>
   <article>
     <h2>Blog Title</h2>
     <p>Blog content...</p>
   </article>
 </section>
</main>


<footer>
 <p>© 2026 Company Name</p>
</footer>
```

---

2. Design a Fully Semantic Blog Page Structure

A semantic blog structure organizes content logically and improves document hierarchy.

Structure:

- `<header>` → Website branding and navigation
- `<main>` → Primary content container

- `<article>` → Individual blog post
- `<section>` → Subtopics within article
- `<aside>` → Related posts or categories
- `<footer>` → Copyright or author info

This improves SEO indexing and allows screen readers to navigate regions effectively.

---

3. Explain ARIA Roles. When Should ARIA Be Avoided?

ARIA (Accessible Rich Internet Applications) enhances accessibility when native HTML elements are insufficient to describe interactive components.

Example:

<nav aria-label="Main Navigation">

When to Use ARIA

- Custom components like tabs, sliders, modals
- Dynamic UI updates
- Complex widgets

When to Avoid ARIA

- When semantic HTML already provides meaning
- Avoid redundant roles like `<button role="button">`

Golden Rule:
Use native HTML first. ARIA only when necessary.

---

4. Create an Accessible Navigation Structure Using ARIA

Accessible navigation must:

- Use semantic `<nav>`
- Include descriptive `aria-label`
- Maintain keyboard navigation
- Provide focus styles

Example:

```
<nav aria-label="Primary Navigation">
 <ul>
   <li><a href="#home" aria-current="page">Home</a></li>
   <li><a href="#pricing">Pricing</a></li>
   <li><a href="#contact">Contact</a></li>
 </ul>
</nav>
```

---

5. Explain aria-label, aria-labelledby, aria-describedby

These attributes improve accessibility.

- `aria-label` → Provides hidden label
- `aria-labelledby` → References visible label
- `aria-describedby` → Adds additional description

Example:

```
<button aria-label="Close menu">X</button>
```

Used in icon buttons, modals, and complex UI components.

---

6. Create a Complex Form using fieldset, legend, datalist, pattern validation

Grouping improves accessibility.

```
<form>
 <fieldset>
   <legend>User Details</legend>

   <label for="email">Email</label>
   <input type="email" id="email" required>

   <label for="country">Country</label>
   <input list="countries" id="country">

   <datalist id="countries">
     <option value="India">
     <option value="USA">
   </datalist>

   <label for="username">Username</label>
   <input type="text" pattern="[A-Za-z]{3,}" required>
 </fieldset>
```

```
</form>
```

Pattern validation ensures correct format without JavaScript.

---

7. Explain Structured Data and Schema Markup

Structured data uses Schema.org vocabulary to help search engines understand content context.

Example:

```
<script type="application/ld+json">

{

 "@context":"https://schema.org",

 "@type":"Article",

 "headline":"Advanced CSS Techniques"

}

</script>
```

Benefits:

- Rich snippets
- Higher click-through rates
- Improved search visibility

---

8. Create FAQ Layout using details and summary

```
<details>

 <summary>What is HTML?</summary>

 <p>HTML is the standard markup language for web pages.</p>
```

```
</details>
```

Advantages:

- No JavaScript required
- Keyboard accessible
- Native browser behavior

---

9. Explain Meta Tags for SEO and Performance

Important meta tags:

```
<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta name="description" content="Advanced HTML and CSS Course">
```

They improve:

- SEO ranking
- Mobile responsiveness
- Proper character encoding

---

10. Build Responsive Images using picture, srcset, sizes

```
<picture>

 <source srcset="large.jpg" media="(min-width: 768px)">

 <img src="small.jpg" alt="Dashboard Preview">

</picture>
```

Benefits:

- Optimized performance
- Reduced bandwidth usage
- Improved LCP score

---

11. Explain defer vs async

Both prevent render-blocking.

- `async` → Executes immediately after download (order not guaranteed)
- `defer` → Executes after HTML parsing (order preserved)

Defer is recommended for structured applications.

---

12. Create Accessible Video Embed with Captions

```
<video controls>
 <source src="video.mp4" type="video/mp4">
 <track kind="captions" src="captions.vtt">
</video>
```

Captions improve accessibility for hearing-impaired users.

---

13. Explain How HTML Impacts Performance Metrics

HTML structure affects:

- Largest Contentful Paint (LCP)
- First Contentful Paint (FCP)

- SEO ranking
- Accessibility score

Well-structured semantic HTML improves rendering performance and crawlability.

---

14. Create Multi-Step Form Structure (HTML Only)

```
<form>
 <section>
   <h2>Step 1: Personal Info</h2>
   <input type="text" required>
 </section>


 <section>
   <h2>Step 2: Account Details</h2>
   <input type="password" required>
 </section>
</form>
```

Using CSS (like radio buttons or :target), steps can be toggled without JavaScript.

---

15. Build Semantic Pricing Table Layout

```
<section aria-labelledby="pricing-heading">
 <h2 id="pricing-heading">Pricing Plans</h2>


 <article>
```

```
  <h3>Basic Plan</h3>

  <p>$9/month</p>

  <ul>

    <li>10 Projects</li>

    <li>Email Support</li>

  </ul>

 </article>

</section>
```

Semantic layout improves SEO and accessibility compared to table-based pricing.

SECTION B — ADVANCED CSS THEORY

---

16. Explain CSS Specificity Hierarchy

Specificity determines which CSS rule applies when multiple rules target the same element.

Hierarchy:
Inline > ID > Class > Element

---

17. Explain Stacking Context

Stacking context controls how elements overlap along the z-axis.

Triggered by:

- position with z-index
- opacity < 1

- transform
- isolation

---

18. Compare All Positioning Types

- static → default
- relative → relative to itself
- absolute → relative to nearest positioned parent
- fixed → relative to viewport
- sticky → toggles between relative and fixed

---

19. Explain Inheritance and Cascade

Inheritance passes certain properties from parent to child.

Cascade determines final style based on:

- Origin
- Specificity
- Importance
- Source order

---

20. Deep Explanation of Box Model

Every element includes:

- Content
- Padding
- Border
- Margin

Using:

box-sizing: border-box;

Simplifies layout calculations.

---

21. Explain Logical Properties

Logical properties adapt layout for different writing modes.

Example:

margin-inline-start: 20px;

padding-block: 10px;

Improves internationalization support.

---

22. Compare em, rem, vh, vw, ch, clamp()

- em → relative to parent
- rem → relative to root
- vh/vw → viewport
- ch → character width
- clamp() → fluid responsive sizing

Example:

```
font-size: clamp(1rem, 2vw, 2rem);
```

---

23. Explain CSS Reset Strategies

CSS reset removes browser inconsistencies.

Example:

```
* {
 margin: 0;
 padding: 0;
 box-sizing: border-box;
}
```

Ensures consistent layout across browsers.

---

24. Explain CSS Containment

CSS Containment improves performance by limiting layout recalculations.

```
.card {
 contain: layout paint;
}
```

---

25. Explain CSS Custom Properties

```
:root {
 --primary-color: #6366f1;
}
```

Improves theme control and maintainability.

---

26. Explain BEM Methodology

BEM stands for Block, Element, Modifier.

Example:

```
.card {}

.card__title {}

.card--active {}
```

Improves scalability and clarity.

---

27. Explain prefers-reduced-motion

```
@media (prefers-reduced-motion: reduce) {

 * {

   animation: none;

 }

}
```

Respects users who disable animations.

---

28. Compare Container Queries and Media Queries

Media queries respond to viewport size.
Container queries respond to parent container size.

Container queries enable modular responsive components.

---

29. Explain will-change and isolation

`will-change` informs the browser of upcoming changes for optimization.

```
.box {
 will-change: transform;
}
```

`isolation: isolate;` creates a new stacking context.

---

30. Explain Critical CSS

Critical CSS loads essential styles first to improve initial render time and reduce render-blocking resources.