Session 3: Interview Questions

1. What are the new features provided by HTML 5?

HTML5 introduced a wide array of new features aimed at enhancing the capabilities of web pages and applications. Here are some of the key new features:

1. New Semantic Elements

HTML5 introduced several new semantic elements that help structure the content more meaningfully:

- <header>: Represents the header section of a document or section.
- <footer>: Represents the footer section of a document or section.
- <article>: Represents a self-contained composition that can be independently distributed or reused.
- <section>: Represents a generic section of a document.
- <nav>: Represents a section of a page that links to other pages or to parts within the page.
- <aside>: Represents content aside from the content it is placed in (like a sidebar).

2. New Form Elements and Attributes

HTML5 introduced new input types and attributes to improve forms:

- New input types: email, url, number, range, date, datetime-local, month, week, time, search, color.
- New attributes: placeholder, required, autofocus, pattern, min, max, step, list.

3. **Multimedia Elements**

HTML5 made it easier to embed media content:

- <audio>: For embedding audio content.
- <video>: For embedding video content.
- <track>: For specifying text tracks for media elements like <video> and <audio>.

4. **Graphics and Effects**

HTML5 added new elements and APIs for creating graphics and visual effects:

- <canvas>: For drawing graphics via scripting (usually JavaScript).
- <svg>: For Scalable Vector Graphics.
- WebGL: For 3D rendering.

5. APIs and Other Technologies

HTML5 comes with a variety of new APIs that enhance the capabilities of web applications:

- Geolocation API: Allows web applications to obtain the geographical location of the user.
- Web Storage API: Provides a way to store data on the client-side, including localStorage and sessionStorage.
- Web Workers API: Allows background scripts to run in parallel with the main page.
- WebSocket API: For full-duplex communication channels over a single TCP connection.
- Server-Sent Events (SSE): For receiving automatic updates from a server via an HTTP connection.
- Drag and Drop API: For drag-and-drop functionality.

6. Enhanced Accessibility

HTML5 includes several features that improve accessibility:

- New elements like <figure> and <figcaption> to associate images with their captions.
- The <mark> element to highlight parts of the text.

These features collectively enhance the functionality, performance, and user experience of modern web applications, making HTML5 a significant improvement over its predecessors.

2. What is the difference between Elements, Tags and Attributes?

1**. Elements**

An HTML element is the basic building block of an HTML document. It consists of a start tag, content, and an end tag. An element can be used to structure the content, apply formatting, embed images, create links, and much more.

Eg:

```
<p>This is a paragraph.</p>
```

In this example:

- <p> is the start tag.
- This is a paragraph. is the content.
- </p> is the end tag.
- The entire <p>This is a paragraph.</p> is the paragraph element.

2. Tags

Tags are the HTML code that tells the browser how to interpret the content. Tags are enclosed in angle brackets (< and >). Most tags come in pairs: an opening tag and a closing tag.

Eg:

```
<h1>Title</h1>
```

In this example:

- <h1> is the opening tag.
- </h1> is the closing tag.

Some tags are self-closing and do not have a closing tag. These are also known as void elements.

Example of a self-closing tag:

```
<img src="image.jpg" alt="Description">
```

In this example, <img> is a self-closing tag.

3. Attributes

Attributes provide additional information about HTML elements. They are always included in the opening tag and are made up of a name and a value pair. Attributes modify the default behavior of elements or provide additional properties.

3. What is the purpose of the <DOCTYPE> declaration in HTML?

   a. **Specifies the Document Type and Version of HTML:**

The <!DOCTYPE> declaration is used to inform the web browser about the version of HTML that the document is written in. This helps the browser to interpret and render the content correctly.

   b. **Triggers Standards Mode:**

The <!DOCTYPE> declaration triggers standards mode (also known as strict mode) in web browsers. In standards mode, browsers render the content according to the HTML and CSS specifications, ensuring consistent behavior across different browsers.

   c. **Avoids Quirks Mode:**

Without a proper <!DOCTYPE> declaration, browsers might render the page in quirks mode. In quirks mode, browsers emulate the behavior of older browsers to support legacy websites, which can lead to inconsistent rendering and unexpected behaviors.

4. What are semantic elements in HTML and why do we use them?

Semantic elements in HTML are tags that clearly describe their meaning in a way that both browsers and developers can understand.

Why Use Semantic Elements?

1. Improved Accessibility:

Semantic element usage helps accessibility, and using non-semantic elements reduces accessibility. For example, screen readers can use semantic tags to provide better navigation options for users with disabilities.

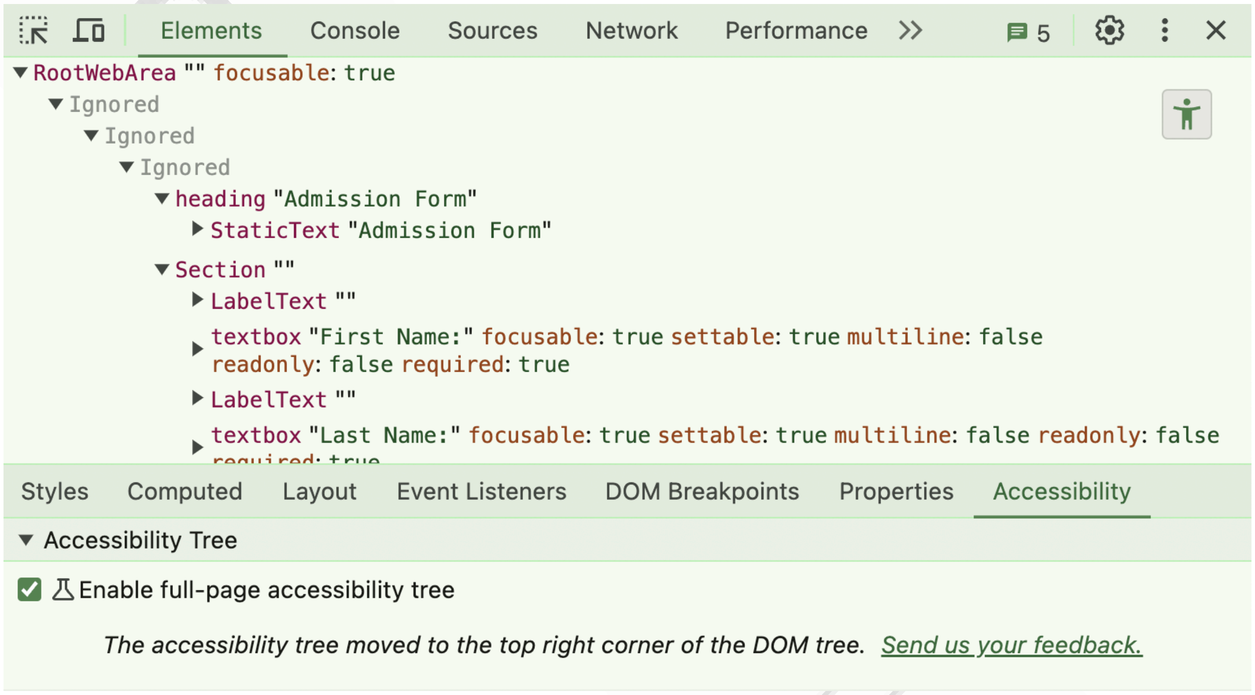
Accessibility object model (AOM)

As the browser parses the content received, it builds the document object model (DOM) and the CSS object model (CSSOM). It then also builds an accessibility tree. Assistive devices, such as screen readers, use the AOM to parse and interpret content. The DOM is a tree of all the nodes in the document. The AOM is like a semantic version of the DOM.

For testing Accessibility of a website: use Lighthouse in devtools

For checking the AOM tree:

Open dev tools -> Elements Tab -> Accessibility -> check the enable full-page tree -> click on the man icon on the top right



```
RootWebArea "" focusable: true
  Ignored
    Ignored
      Ignored
        heading "Admission Form"
          StaticText "Admission Form"
        Section ""
          LabelText ""
          textbox "First Name:" focusable: true settable: true multiline: false
          readonly: false required: true
          LabelText ""
          textbox "Last Name:" focusable: true settable: true multiline: false readonly: false
```

Accessibility Tree

✓ Enable full-page accessibility tree

The accessibility tree moved to the top right corner of the DOM tree. *Send us your feedback.*

2. Better SEO:

Search engines use the structure of a web page to determine the importance and context of its content. Semantic elements help search engines understand the hierarchy and relevance of the content, which can lead to improved search rankings.

3. Enhanced Readability:

Semantic HTML provides a clearer structure for developers, making the code easier to read and maintain. This clarity can speed up development time and reduce errors.

5. What's the difference between `div` and `section` tag? Which one should we use?

   The <div> and <section> tags are both used to define sections of content within an HTML document, but they serve different purposes and have distinct semantic meanings.

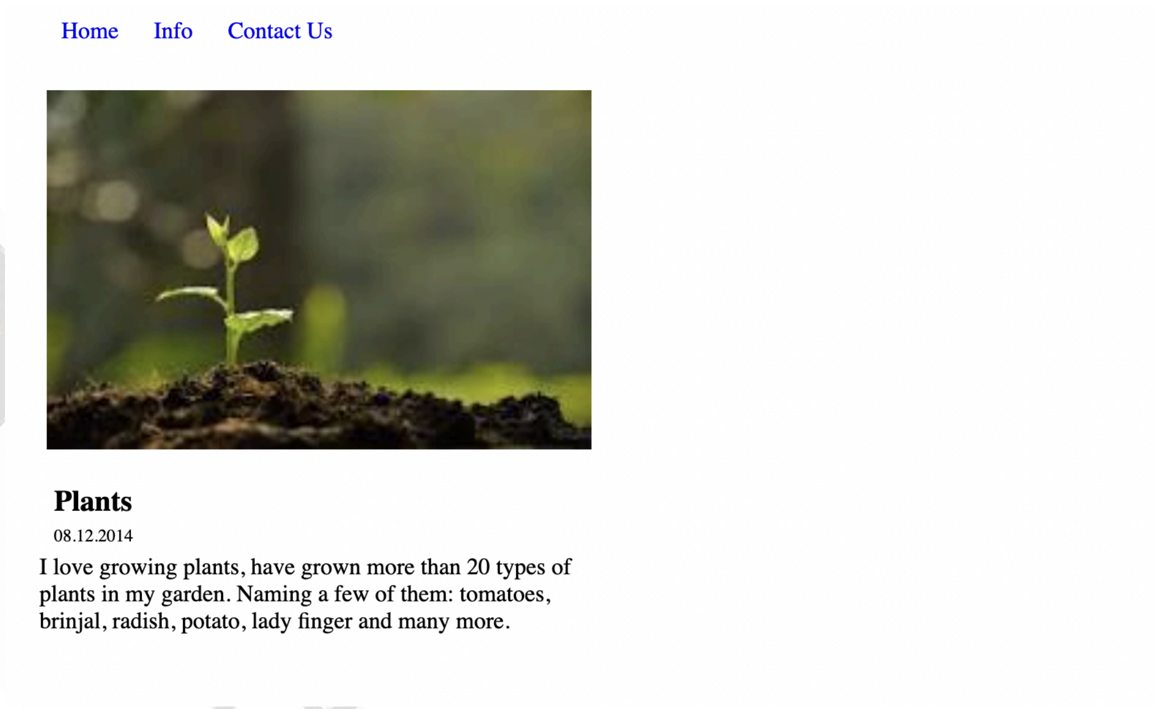
   <div> Element:
   Definition: The <div> (short for "division") element is a generic container for flow content, which does not have any semantic meaning. It is primarily used for styling and layout purposes.
   Usage: Use the <div> element when you need a container solely for styling or layout purposes, without conveying any specific semantic meaning. It is commonly used with CSS to apply styles or with JavaScript to manipulate groups of elements.

   <section> Element:
   Definition: The <section> element represents a standalone section of content that is thematically related. It is a semantic element, meaning it provides meaningful information about the content it contains.
   Usage: Use the <section> element when you want to enhance the semantic structure of the document, which can improve accessibility and SEO.

6. Give examples of a few semantic tags.

   a. <header>:
      - The <header> HTML element represents introductory content, typically a group of introductory or navigational aids. It may contain some heading elements but also a logo, a search form, an author name, and other elements.
      - The <header> element can define a global site header, described as a banner in the accessibility tree.
      - It is generally located at the top of the page.

      Eg: In the example below you can see three ways in which header has been used. Firstly it has been used to wrap navigation links, secondly it is used to wrap the logo and thirdly it is used to wrap heading inside an article.



Home Info Contact Us

Plants

08.12.2014

I love growing plants, have grown more than 20 types of plants in my garden. Naming a few of them: tomatoes, brinjal, radish, potato, lady finger and many more.

HTML file : index.html

```
<header>
  <nav class="navbar">
   <a href="#home">Home</a>
   <a href="#info">Info</a>
   <a href="#contact">Contact Us</a>
 </nav>
</header>

<header>
 <img src = "./assets/plant-img.png"/>
</header>

<article>
 <header>
   <h1>Plants</h1>
   <h6>08.12.2014</h6>
 </header>
 <p>I love growing plants, have grown more than 20 types of plants in my garden. Naming a few of them: tomatoes, brinjal, radish, potato, lady finger and many more.</p>
</article>
```

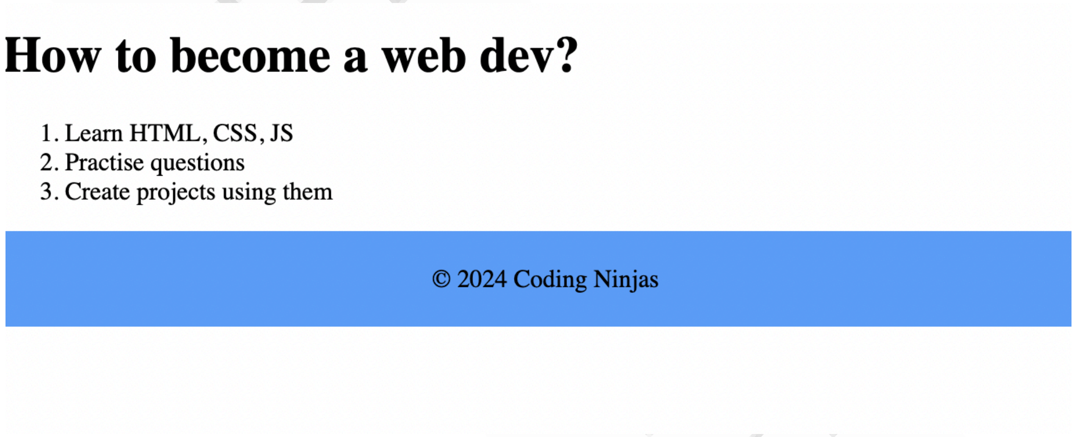
CSS file: style.css

```
*{
  margin: 5px 10px;
}

.navbar{
  height: 40px;


}

.navbar a{
  text-decoration: none;
}

img{
  width: 40%
}



article {
  width: 40%;
   padding: 5px;
}

h1{
  font-size: 20px;
  font-weight: bold;
}
```

```
h6{
  font-size: 12px;
  margin: 5px 0px;
}
```

b. <footer>:
- The <footer> HTML element represents a footer for its nearest ancestor sectioning content or sectioning root element.
- A <footer> typically contains information about the author of the section, copyright data or links to related documents.

Eg 1: Here footer is being added for a section inside a content



How to become a web dev?

1. Learn HTML, CSS, JS
2. Practise questions
3. Create projects using them

© 2024 Coding Ninjas

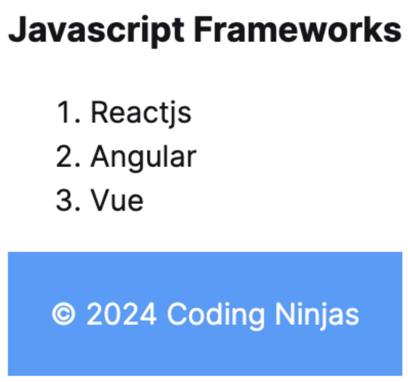
HTML file:

```
<article>
  <h1>How to become a web dev?</h1>
  <ol>
    <li>Learn HTML, CSS, JS</li>
    <li>Practise questions</li>
    <li>Create projects using them</li>
  </ol>
  <footer>
    <p>(c) 2024 Coding Ninjas</p>
  </footer>
</article>
```

CSS file:

```
footer {
  display: flex;
  justify-content: center;
  padding: 5px;
  background-color: #45a1ff;
  color: #fff;
}
```

Eg 2: Here a footer is being added for the root element.


Javascript Frameworks

1. Reactjs
2. Angular
3. Vue

© 2024 Coding Ninjas

HTML file:

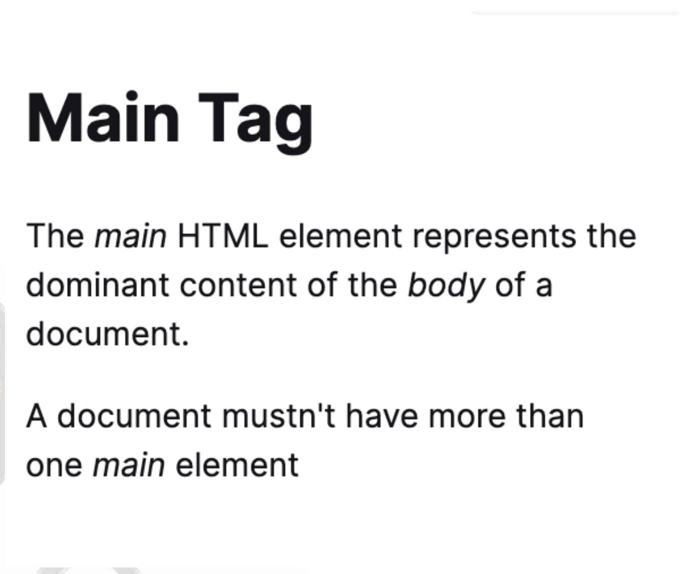
```
<body>
  <h3>Javascript Frameworks</h3>
  <ol>
    <li>Reactjs</li>
    <li>Angular</li>
    <li>Vue</li>
  </ol>
  <footer>
    <p>(c) 2024 Coding Ninjas</p>
  </footer>
</body>
```

CSS file:

```
footer {
  display: flex;
  justify-content: center;
  padding: 5px;
  background-color: #45a1ff;
  color: #fff;
}
```

c. <main>:
   - The <main> HTML element represents the dominant content of the <body> of a document.
   - A document mustn't have more than one <main> element .

   Eg:



Main Tag

The *main* HTML element represents the dominant content of the *body* of a document.

A document mustn't have more than one *main* element

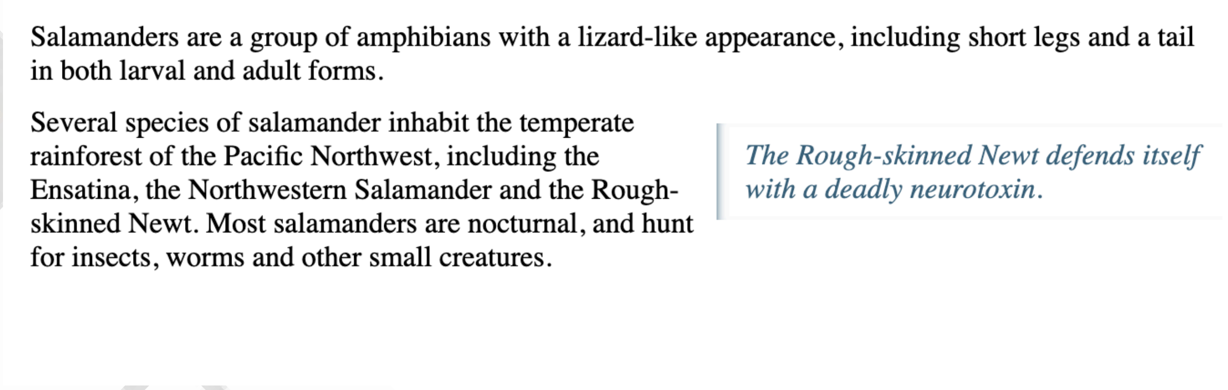
Html file:

```
<main>
  <section>
    <p>
      The <em>main</em> HTML element
      represents the dominant content of the
      <em>body</em> of a document.
    </p>
    <p>
      A document mustn't have more than one
      <em>main</em> element
    </p>
  </section>
</main>
```

d. <aside>:
   - The <aside> HTML element represents a portion of a document whose content is only indirectly related to the document's main content.

- Asides are frequently presented as sidebars or call-out boxes.
- They won't create any separate block on the UI, the UI would look similar to cases if you would use any div or section tag as well.

Eg:



Salamanders are a group of amphibians with a lizard-like appearance, including short legs and a tail in both larval and adult forms.

Several species of salamander inhabit the temperate rainforest of the Pacific Northwest, including the Ensatina, the Northwestern Salamander and the Rough-skinned Newt. Most salamanders are nocturnal, and hunt for insects, worms and other small creatures.

The Rough-skinned Newt defends itself with a deadly neurotoxin.

HTML File:

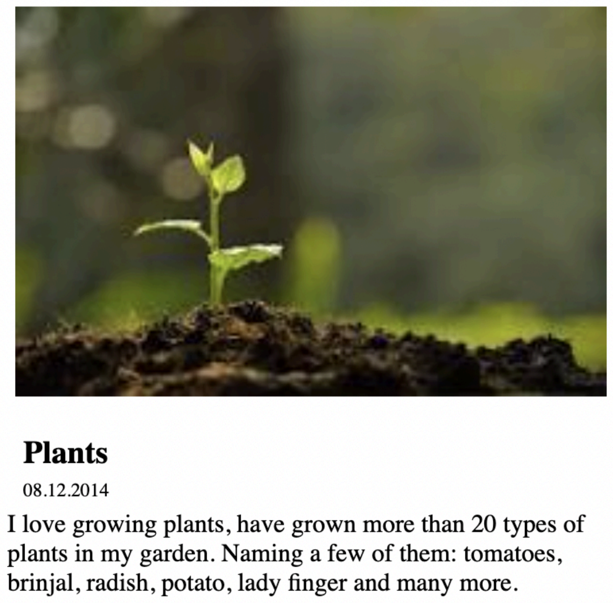
```
<p>
  Salamanders are a group of amphibians with a lizard-like appearance, including short legs and a tail in both larval
  and adult forms.
</p>

<aside>
  <p>The Rough-skinned Newt defends itself with a deadly neurotoxin.</p>
</aside>

<p>
  Several species of salamander inhabit the temperate rainforest of the Pacific Northwest, including the Ensatina, the
  Northwestern Salamander and the Rough-skinned Newt. Most salamanders are nocturnal, and hunt for insects, worms and
  other small creatures.
</p>
```

CSS File:

```
*{
  margin: 10px;
}
aside {
  width: 40%;
  padding-left: 0.5rem;
  margin-left: 0.5rem;
  float: right;
  box-shadow: inset 5px 0 5px -5px #29627e;
  font-style: italic;
  color: #29627e;
}

aside > p {
  margin: 0.5rem;
}
```

e. <article>:
 - The <article> HTML element represents a self-contained composition in a document.
 - Examples include: a forum post, a magazine or newspaper article, or a blog entry, a product card, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.

 Eg:



Plants

08.12.2014

I love growing plants, have grown more than 20 types of plants in my garden. Naming a few of them: tomatoes, brinjal, radish, potato, lady finger and many more.

HTML file: index.html

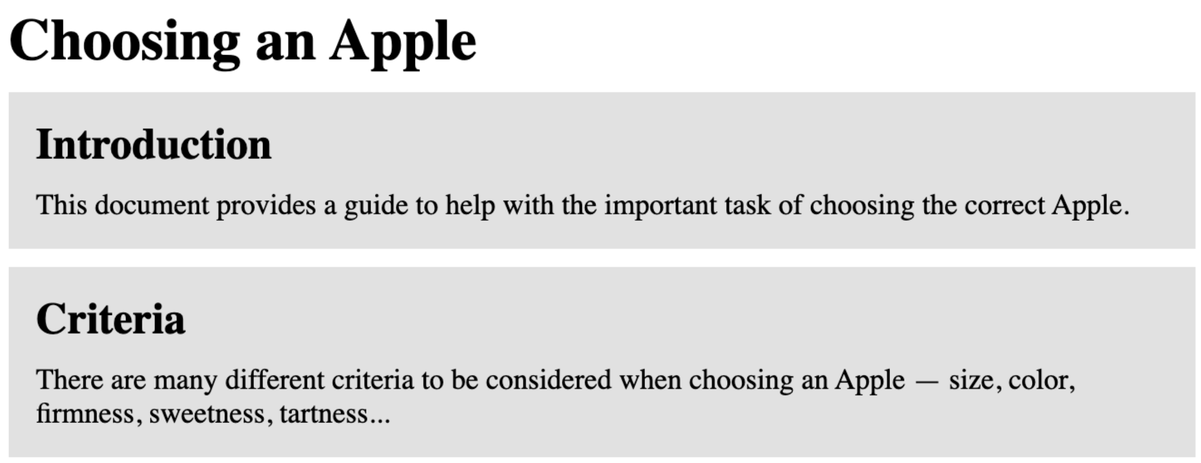
```
<article>
<section>
 <img src = "./assets/plant-img.png"/>
</section>

 <header>
   <h1>Plants</h1>
   <h6>08.12.2014</h6>
 </header>
 <p>I love growing plants, have grown more than 20 types of plants in my garden. Naming a
few of them: tomatoes, brinjal, radish, potato, lady finger and many more.</p>
</article>
```

CSS file: style.css

```
img{
  width: 40%
}



article {
  width: 40%;
   padding: 5px;
}

h1{
  font-size: 20px;
  font-weight: bold;
}

h6{
  font-size: 12px;
  margin: 5px 0px;
}
```

7. <section>:
   The <section> tag defines a section in a document. The div tag also does the same thing, just the difference is that it’s non-semantic.

   Eg:



Choosing an Apple

Introduction

This document provides a guide to help with the important task of choosing the correct Apple.

Criteria

There are many different criteria to be considered when choosing an Apple — size, color, firmness, sweetness, tartness...

HTML file:

```
<h1>Choosing an Apple</h1>
<section>
  <h2>Introduction</h2>
  <p>This document provides a guide to help with the important task of choosing the correct
Apple.</p>
</section>

<section>
  <h2>Criteria</h2>
  <p>
    There are many different criteria to be considered when choosing an Apple -- size, color,
firmness, sweetness,
    tartness...
  </p>
</section>
```

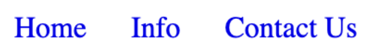
CSS File:

```
section{
  background-color: #e1e1e1;
  padding: 5px;
}
```

F. <nav>:

- It represents the section of the document which consists of all the navigation links.
- It’s generally present at the top of the page.
- Common examples of navigation sections are menus, tables of contents, and indexes.

Eg:



Home Info Contact Us

HTML File:

```
<nav class="navbar">
  <a href="#home">Home </a>
  <a href="#info">Info</a>
  <a href="#contact">Contact Us</a>
</nav>
```

8. What is the importance of the <meta> tag in HTML?

The <meta> tag in HTML plays a crucial role in providing metadata about an HTML document. Metadata is data about data, and it helps browsers, search engines, and other services understand the content and settings of a web page. Here are the key aspects and importance of the <meta> tag:

Importance of the <meta> Tag

a. Character Encoding:

The <meta> tag can specify the character encoding for the HTML document, ensuring that text is rendered correctly.

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

This declaration helps avoid issues with displaying special characters and ensures that the document is interpreted correctly across different browsers and platforms.

b. Viewport Settings:

The <meta> tag can define how a web page should be displayed on different devices, particularly mobile devices.

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

This setting makes the web page responsive, meaning it adjusts its layout based on the screen size and resolution, enhancing the user experience on mobile devices.

c. SEO (Search Engine Optimization):

<meta> tags can provide information to search engines about the content of the page, helping improve the page's visibility and ranking in search results.

```
<meta name="description" content="A brief description of the page content">
<meta name="keywords" content="keyword1, keyword2, keyword3">
```

The description tag provides a summary of the page, often displayed in search engine results. The keywords tag lists relevant keywords, although its importance has diminished over time.

d. Author and Copyright Information:

<meta> tags can specify the author of the document and provide copyright information.

```
<meta name="author" content="Author Name">
<meta name="copyright" content="(c) 2024 Example Corp.">
```

e. Refresh and Redirect:

The <meta> tag can be used to refresh the page or redirect to another URL after a specified time.

```
<meta http-equiv="refresh" content="30">
<meta http-equiv="refresh" content="5; url=https://www.example.com">
```

The first example refreshes the page every 30 seconds, while the second example redirects the user to a different URL after 5 seconds.

f. Setting HTTP Headers:

The <meta> tag can simulate HTTP headers, controlling aspects like content type, cache control, and page expiration.

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="Tue, 01 Jan 2025 00:00:00 GMT">
```

Example of `meta` tags in HTML document

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta name="description" content="This is an example of a meta description tag.">
   <meta name="keywords" content="HTML, meta tags, example">
   <meta name="author" content="John Doe">
   <meta property="og:title" content="Example Page">
   <meta property="og:description" content="This is an example page with meta
tags.">
   <meta property="og:image" content="https://www.example.com/image.jpg">
```

```
    <meta property="og:url" content="https://www.example.com/page.html">
    <title>Example Page with Meta Tags</title>
</head>
<body>
    <h1>Welcome to the Example Page</h1>
    <p>This page demonstrates the use of meta tags in HTML.</p>
</body>
</html>
```

9. What is the difference between HTML and XHTML?

HTML and XHTML are both markup languages used to create web pages, but they have some key differences:

- HTML (HyperText Markup Language) is more lenient with syntax rules. Browsers can often correct errors in HTML code.
- XHTML (Extensible HyperText Markup Language) is stricter with syntax rules and is an XML-based language. Every tag must be properly closed, nested correctly, and written in lowercase.

10. How can we add SEO using HTML tags?

To enhance SEO using HTML tags, implement the following key practices:

a. <title> Tag

Purpose: Defines the title of the web page, displayed in search engine results and browser tabs.
Best Practices:

- Keep it concise (under 60 characters).
- Include relevant keywords.
- Make it descriptive and compelling.

b. <meta name="description"> Tag

Purpose: Provides a summary of the web page content, often shown in search engine results.
Best Practices:

- Keep it between 150-160 characters.
- Include relevant keywords naturally.
- Write a compelling and accurate summary.

c. Header Tags (<h1>, <h2>, etc.)

Purpose: Define headings and subheadings, structuring content for readability and search engines.

Best Practices:

- Use one <h1> tag per page for the main heading.
- Use <h2> to <h6> for subheadings in hierarchical order.
- Include relevant keywords.

d. <a> Tag (Anchor Tag)

Purpose: Creates hyperlinks, crucial for SEO by establishing connections between pages.

Best Practices:

- Use descriptive anchor text with relevant keywords.
- Use the title attribute for additional context.
- Link to authoritative and relevant pages.

e. <img> Tag

Purpose: Embeds images into the page, with the alt attribute important for accessibility and SEO.

Best Practices:

- Use descriptive alt text that includes keywords.
- Use the title attribute to provide additional information.
- Optimize image file size and format for faster loading.

f. <meta name="keywords"> Tag

Purpose: Lists relevant keywords for search engines.

Best Practices:

- Include relevant keywords.

- Avoid keyword stuffing.

g. Semantic HTML5 Tags (<header>, <footer>, <section>, <article>, <aside>)

Purpose: Structure content logically, aiding readability and SEO.

Best Practices:

- Use these elements to clearly define different parts of your content.
- Ensure each section has relevant headings and content.

h. <nav> Tag

Purpose: Defines navigation links, helping users and search engines understand the site structure.

Best Practices:

- Use for primary navigation links.
- Ensure the navigation is clear and logical.

i. <strong> and <em> Tags

Purpose: Emphasize text, with <strong> indicating strong importance and <em> indicating emphasis.

Best Practices:

a. Use to highlight important keywords.
b. Avoid overusing to prevent keyword stuffing.

11. What is the difference between the ‘id’ attribute and the ‘class’ attribute of HTML elements?

The id and class attributes in HTML are both used to apply styles and manipulate elements with CSS and JavaScript, but they serve different purposes and have different characteristics:

id Attribute

- Uniqueness: The id attribute must be unique within a document. No two elements can have the same id.
- Usage: It's used to identify a single, unique element.

- Selector: In CSS, an id is selected with a # followed by the id value. In JavaScript, it can be accessed using document.getElementById("id").
- Specificity: An id has a higher specificity in CSS compared to a class, meaning it will override styles defined by a class.

class Attribute

- Non-uniqueness: The class attribute is not unique. Multiple elements can share the same class.
- Usage: It's used to identify one or more elements that belong to a group or share common styles.
- Selector: In CSS, a class is selected with a . followed by the class name. In JavaScript, it can be accessed using document.getElementsByClassName("class") or document.querySelectorAll(".class").
- Specificity: A class has lower specificity compared to an id, but it's more flexible for applying styles to multiple elements.

12. What are void elements in HTML?

Void elements in HTML, also known as self-closing or empty elements, are elements that do not have any content and do not need a closing tag. They are typically used to insert an element that doesn't require any content within it.

Eg: <br>, <col>, <embed>, <hr>, <img>, <input>, <link>, <meta>

13. What is the difference between <strong>, <b> tags and <em>, <i> tags?

<strong> and <em>:

a. Carry semantic meaning.
b. Convey importance and emphasis respectively.
c. Useful for accessibility and SEO.

<b> and <i>:

d. Used for visual styling without semantic meaning.
e. Purely presentational.
f. Not SEO friendly

14. Can we display a web page inside a web page or Is nesting of webpages possible?

Yes, it is possible to display a web page inside another web page. This is commonly done using the <iframe> element in HTML. The <iframe> element allows you to embed another HTML document within the current document, effectively nesting web pages.
<iframe> Element
The <iframe> (inline frame) element is used to embed another HTML document within the current HTML document. Here's how it works:

Eg:

```
<!DOCTYPE html>
<html>
<head>
    <title>Parent Page</title>
</head>
<body>
    <h1>Welcome to the Parent Page</h1>
    <iframe src="https://www.example.com" width="600" height="400">
        Your browser does not support iframes.
    </iframe>
</body>
</html>
```

15. Create a Simple Flex Container

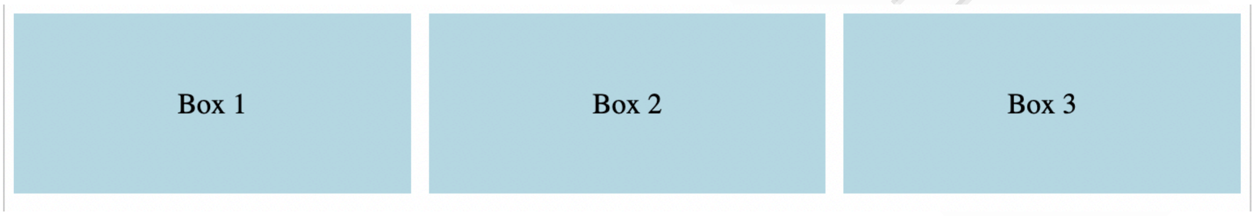
Question: Create a container with three boxes side by side, using Flexbox. Each box should be of equal width and height.

Answer:

```
<div class="container">
   <div class="box">Box 1</div>
   <div class="box">Box 2</div>
   <div class="box">Box 3</div>
</div>
```

```
.container {
   display: flex;
   justify-content: center;
   align-items: center;
}
.box {
   flex: 1;
   height: 100px;
   background-color: lightblue;
   margin: 5px;
}
```

Output:



16. Align Items Center

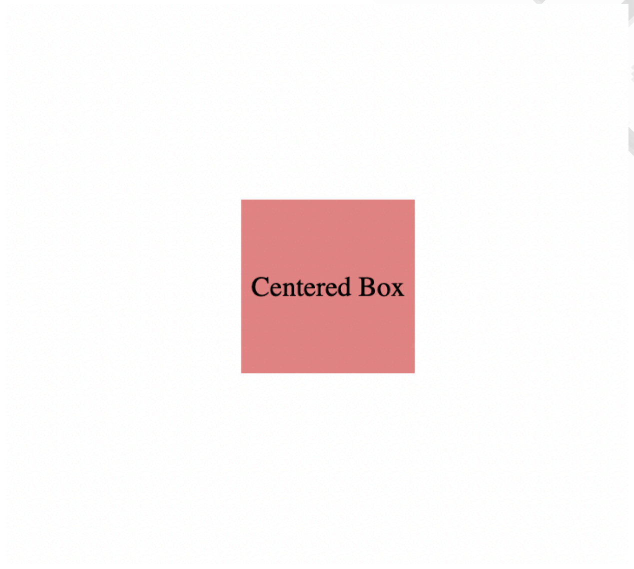
Question: Using Flexbox, center an element both horizontally and vertically within a container.

Answer:

```
<div class="container">
   <div class="box">Centered Box</div>
</div>
```

```
.container {
   display: flex;
   justify-content: center;
   align-items: center;
   height: 100vh;
}
.box {
   width: 100px;
   height: 100px;
   background-color: lightcoral;
}
```

Output:



17. Responsive Flexbox Layout

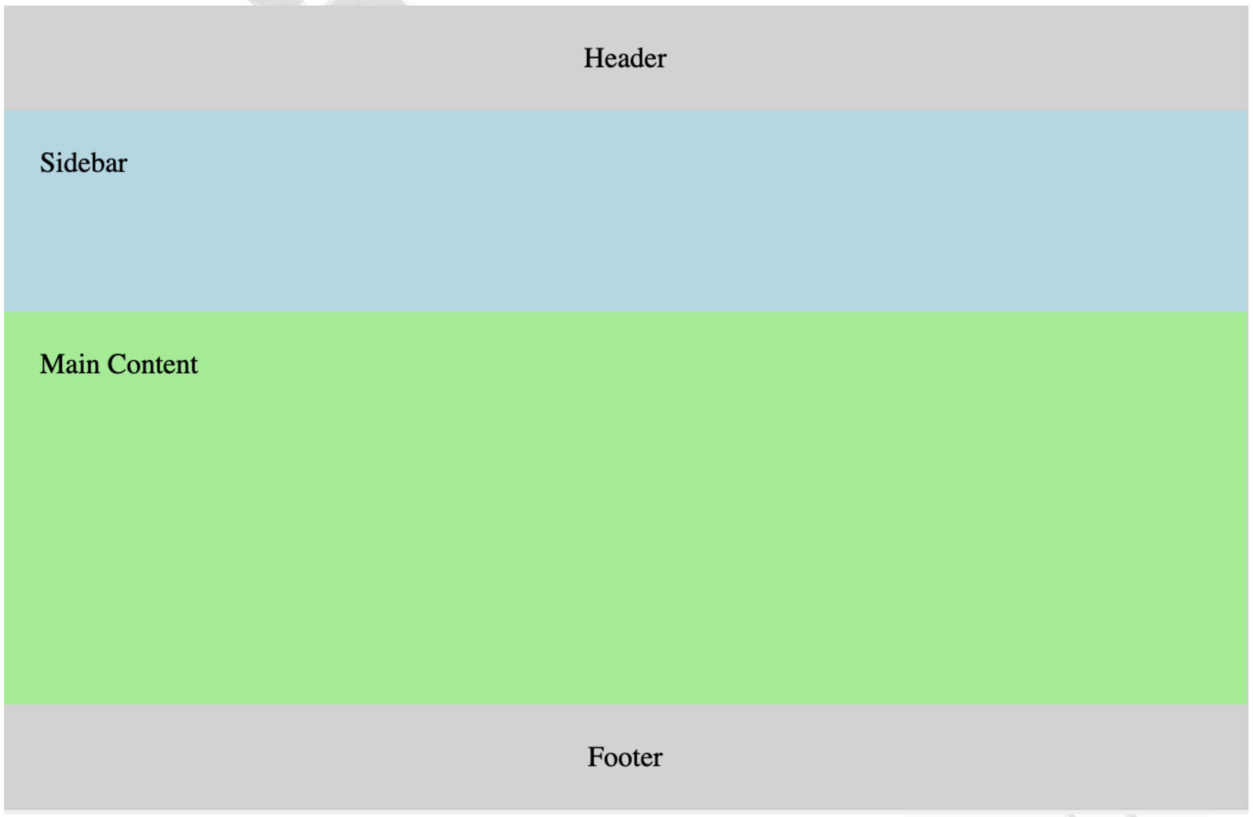
Question: Create a responsive layout with a header, sidebar, main content, and footer using Flexbox. Ensure that the sidebar and main content are side by side on larger screens and stacked on smaller screens.

Answer:

```
<div class="container">
    <header class="header">Header</header>
    <div class="content">
        <aside class="sidebar">Sidebar</aside>
        <main class="main">Main Content</main>
    </div>
    <footer class="footer">Footer</footer>
</div>
```

```
.container {
    display: flex;
    flex-direction: column;
    min-height: 100vh;
}
.header, .footer {
    background-color: lightgray;
    text-align: center;
    padding: 20px;
}
.content {
    display: flex;
    flex: 1;
}
.sidebar {
    flex: 1;
    background-color: lightblue;
    padding: 20px;
}
.main {
    flex: 3;
    background-color: lightgreen;
    padding: 20px;
}
@media (max-width: 768px) {
    .content {
        flex-direction: column;
    }
}
```

Output:



18. Flexbox Order Property

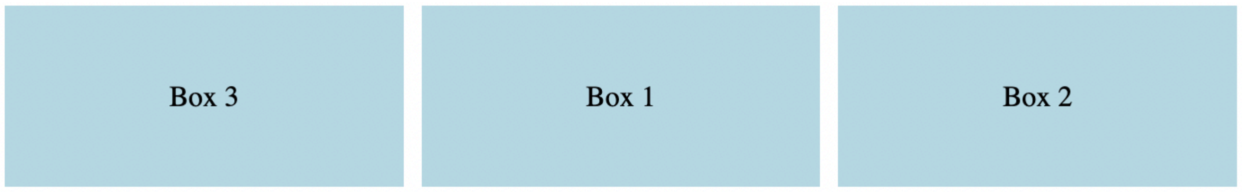
Question: Use Flexbox to create a container with three items. Change the order of the items so that the third item appears first.

Answer:

```
<div class="container">
   <div class="box box1">Box 1</div>
   <div class="box box2">Box 2</div>
   <div class="box box3">Box 3</div>
</div>
```

```
.container {
   display: flex;
   justify-content: center;
   align-items: center;
}
.box {
   flex: 1;
   height: 100px;
   background-color: lightblue;
   margin: 5px;
  }
.box1 {
   order: 2;
}
.box2 {
   order: 3;
}
.box3 {
   order: 1;
}
```

Output:



19. Creating a Navigation Bar with Space Between Links

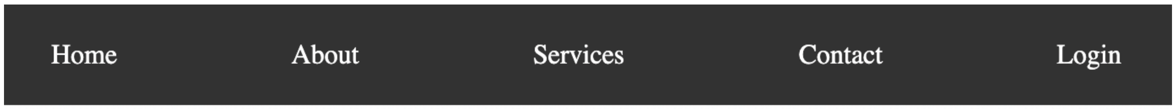
Question: Create a navigation bar with five links. Use Flexbox to space the links evenly across the width of the navigation bar.

Answer:

```
<nav class="navbar">
   <a href="#">Home</a>
   <a href="#">About</a>
   <a href="#">Services</a>
   <a href="#">Contact</a>
   <a href="#">Login</a>
</nav>
```

```
.navbar {
   display: flex;
   justify-content: space-between;
   background-color: #333;
   padding: 10px;
}
.navbar a {
   color: white;
   text-decoration: none;
   padding: 10px 20px;
}
```

Output:


Home About Services Contact Login

20. Equal Height Columns

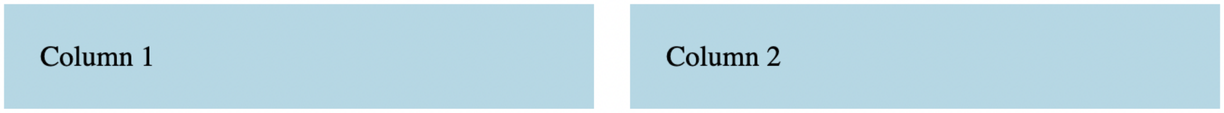
Question: Create two columns of equal height using Flexbox. Each column should have some content.

Answer:

```
<div class="container">
    <div class="column">Column 1</div>
    <div class="column">Column 2</div>
</div>
```

```
.container {
    display: flex;
}
.column {
    flex: 1;
    padding: 20px;
    background-color: lightblue;
    margin: 10px;
}
```

Output:


Column 1 Column 2

21. Flexbox with Nested Flex Containers

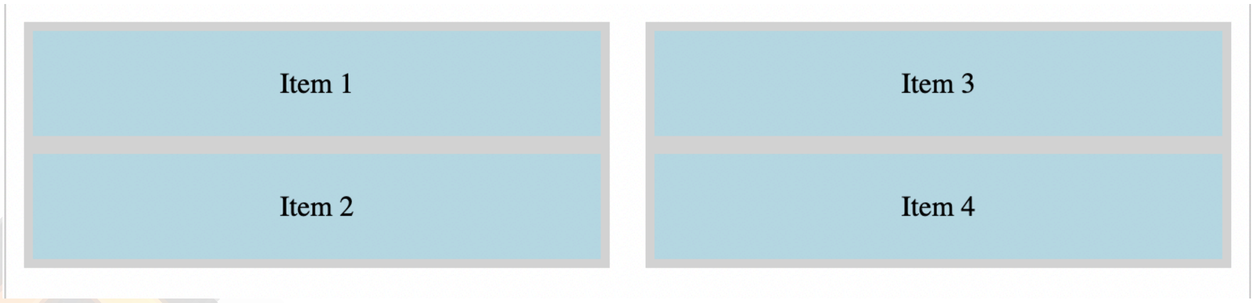
Question: Create a flex container with two nested containers inside. The outer container should arrange the nested containers in a row. Each nested container should arrange its children in a column.

Answer:

```
<div class="outer-container">
    <div class="inner-container">
        <div class="item">Item 1</div>
        <div class="item">Item 2</div>
    </div>
    <div class="inner-container">
        <div class="item">Item 3</div>
        <div class="item">Item 4</div>
    </div>
</div>
```

```
.outer-container {
    display: flex;
}
.inner-container {
    display: flex;
    flex-direction: column;
    flex: 1;
    margin: 10px;
    background-color: lightgray;
}
.item {
    background-color: lightblue;
    margin: 5px;
    padding: 20px;
    text-align: center;
}
```

Output:



```
Item 1                Item 3

Item 2                Item 4
```