Mohammad Bayajeed
bayajeedbd@gmail.com

Q1: What is axis in CSS Flexbox? Explain with an example.

Answer:

Two types of axis in Flexbox

1. Main Axis (Horizontal axis)
2. Cross Axis (Vertical axis)

Main Axis: The main axis is the primary axis along which flex items are placed. By default, this axis runs horizontally from left to right

- **Flex Direction**:
  row (default): The main axis runs horizontally (left-to-right).
  row-reverse: The main axis runs horizontally but in reverse order (right-to-left).
  column: The main axis runs vertically (top-to-bottom).
  column-reverse: The main axis runs vertically but in reverse order (bottom-to-top).

- **Properties Affecting the Main Axis**:
  ✓ **justify-content:** Aligns and distributes flex items along the main axis.
  Options include:
  flex-start: Aligns items to the start of the main axis.
  flex-end: Aligns items to the end of the main axis.
  center: Centers items along the main axis.
  space-between: Distributes items with equal space between them.
  space-around: Distributes items with equal space around them.

Cross Axis: The cross axis is perpendicular to the main axis. If the main axis is horizontal, the cross axis is vertical, and if the main axis is vertical, the cross axis is horizontal.

- **Properties Affecting the Cross Axis**:
  ✓ **align-items**: Aligns flex items along the cross axis.
  Options include:
  flex-start: Aligns items to the start of the cross axis.
  flex-end: Aligns items to the end of the cross axis.
  center: Centers items along the cross axis.
  baseline: Aligns items along their baselines.
  stretch: Stretches items to fill the container along the cross axis.

  ✓ **align-content**: Aligns flex lines (if there are multiple lines) along the cross axis. This property only has an effect if there are multiple lines of items (due to wrapping).
  Options include:
  flex-start: Aligns lines to the start of the cross axis.
  flex-end: Aligns lines to the end of the cross axis.
  center: Centers lines along the cross axis.
  space-between: Distributes lines with equal space between them.
  space-around: Distributes lines with equal space around them.
  stretch: Stretches lines to fill the container along the cross axis.

  ✓ **align-self**: Allows the alignment of a single flex item along the cross axis, overriding the container align-items setting.

Mohammad Bayajeed
bayajeedbd@gmail.com

Q2: Can I use multiple CSS Background images in a div? Explain with an example.

Answer:

Yes, we can use multiple background images in a div using CSS.
background-image: url('image1.jpg'), url('image2.png');

Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Multiple Background Images</title>
    <style>
        .multi-background {
            width: 400px;
            height: 400px;
            background-image: url('image1.jpg'), url('image2.png');
            background-position: center, top left;
            background-size: cover, 50px 50px;
            background-repeat: no-repeat, repeat;
            border: 1px solid black;
        }
    </style>
</head>
<body>
    <div class="multi-background"></div>
</body>
</html>
```

Main Things is

background-image: url(img_flwr.gif), url(paper.gif);
background-position: center, left top;
background-repeat: no-repeat, repeat;

- **background-image**: This property defines multiple background images.
- **background-position**: Specifies the position of each background image.
- **background-size**: Specifies the size of the background images.
- **background-repeat**: Controls whether the background images repeat. The first image doesn't repeat (no-repeat), while the second repeats across the div (repeat).

Mohammad Bayajeed
bayajeedbd@gmail.com

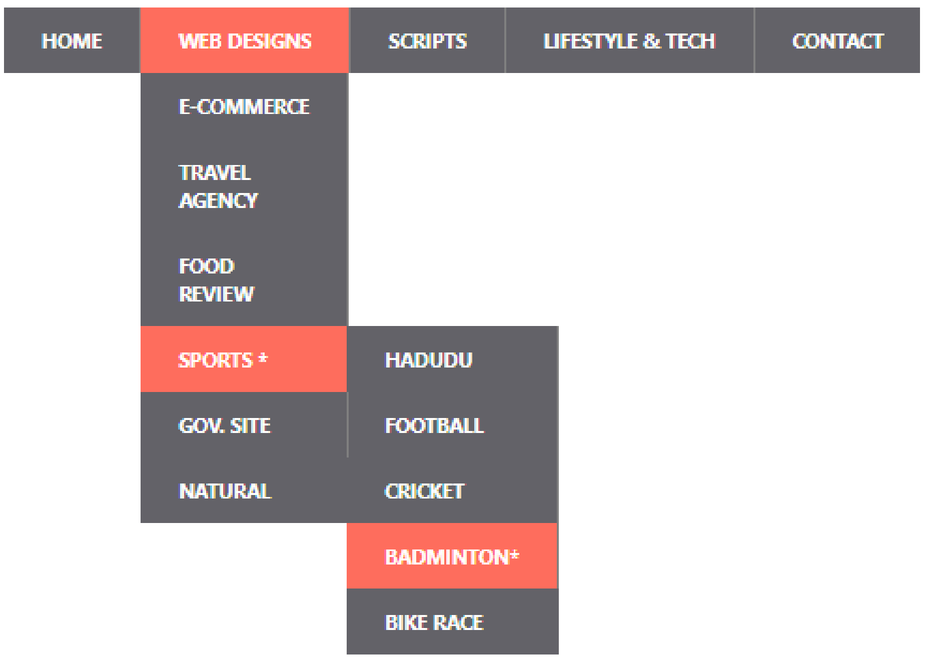
Q3: Code the following section:

Answer:

DropDn Manu View: **https://66e737e4af8a91418bcd2588--incomparable-mandazi-f19a16.netlify.app/**

Code link : https://github.com/bayajeedwithgo/Menu.git

Output:



Mohammad Bayajeed
bayajeedbd@gmail.com

Q4: Explain CSS Specificity

Answer:

CSS specificity determines which style rules apply to an element when there are conflicting rules. It's essentially a measure of how specific a selector is in targeting an element. Specificity is calculated based on four categories:

1. **Inline styles** (styles applied directly to the HTML element via the style attribute) have the highest specificity.
2. **ID selectors** (#header).
3. **Class selectors**, **attribute selectors**, and **pseudo-classes** (.main, [type="text"], :hover).
4. **Element selectors** and **pseudo-elements** (div, h1, :: before).

Specificity Calculation:

Selector property flowing things **(IICT)**

1. Inline styles (1000).
2. ID selectors (0100).
3. Class, attribute, pseudo-class selectors (0010).
4. Tag/Element and pseudo-elements (0001).

Example:

```css
/* Example 1 */
h1 {
    color: blue; /* specificity (0, 0, 0, 1) */
}

.main h1 {
    color: red;  /* specificity (0, 0, 1, 1) */
}

#header h1 {
    color: green;  /* specificity (0, 1, 0, 1) */
}
```

Mohammad Bayajeed
bayajeedbd@gmail.com

Q5: Explain CSS Box Model

Answer:

The CSS Box Model is a fundamental concept in web design and layout. It describes how the elements on a web page are structured and spaced. The CSS box model is essentially a box that wraps around every HTML element.

We use two types of Box sizing

```css
box-sizing: border-box;
box-sizing: content-box;
```

It consists of: **content**, **padding**, **borders** and **margins**.

1. **Content**: This is the actual content inside the box, such as text, images, or other elements. It is the innermost part of the box.
2. **Padding**: Padding is the space between the content and the border. It clears an area around the content, making sure the content does not touch the border directly. Padding is inside the border.
3. **Border**: The border wraps around the padding and the content. It can have various styles, colors, and thicknesses. The border is placed between the padding and the margin.
4. **Margin**: The margin is the outermost layer, creating space between the border of the element and adjacent elements. Unlike padding, margins are outside the border and can also collapse with the margins of neighboring elements, especially in block-level elements.

Q6: Code the following section: CSS Timeline

Answer:

View link: https://66e848b769ac3f0749da5f98--jade-torte-246fa2.netlify.app/

Q7: Code the following section:

Answer:

View link: https://66e852d4a4f9bc1f0d0ca481--bespoke-daffodil-edbac7.netlify.app/

Mohammad Bayajeed
bayajeedbd@gmail.com

Q8: What are pseudo selectors? According to you explain 5 important pseudo selectors.

Answer:

Pseudo selectors are used to style specific states of elements that cannot be targeted using regular selectors. They help in enhancing user experience by targeting certain conditions or specific portions of an HTML element.

There are two main types of pseudo-selectors:

1. **Pseudo-classes** (:): Target elements based on conditions like hover, focus, or position (:hover, :nth-child()).
2. **Pseudo-elements** (::): Style parts of an element, such as before or after content (::before, ::after).

Here are 5 important pseudo-selectors with explanations:

1. **:hover**
   Usage: This selector is applied when a user hovers over an element with the mouse.
   Importance: It's commonly used in creating interactive effects like changing the color of buttons or links when hovered.

Example:

```css
button:hover {
  background-color: blue;
  color: white;
}
```

2. **:nth-child()**
   Usage: This pseudo-selector selects elements based on their position among siblings. we can specify a number or a formula (odd or even).
   Importance: It provides granular control over styling specific elements in a list or grid.

Explain:

```css
li:nth-child(2) {
    color: red;
}
li:nth-child(even){
    background-color: gray;
}
```

3. **:focus**
   Usage: Targets elements that are focused, typically form inputs or links that are selected by keyboard or clicked.
   Importance: It improves accessibility by indicating which element is active, helping users navigate.

Example:

```css
input:focus {
    border-color: green;
    outline: none;
```

```
  }
```

4. **:first-child**
   Usage: Selects the first child of an element.
   Importance: Useful for styling the first item in a list, menu, or section differently.

Example:

```
p:first-child {
    font-weight: bold;
  }
```

5. **:before**
   Usage: This pseudo-element allows you to insert content before an element without modifying the HTML.
   Importance: It is widely used for adding decorative elements or icons before text or other content.

Example:

```
h1:before {
    content: "★ ";
    color: gold;
}
```

Q9: What are media queries? Min-width or max-width which will be more helpful on Mobile First Approach? Explain.

Answer:

Media queries are CSS techniques used to apply different styles based on the characteristics of the user's device, such as screen size, resolution, or orientation. They help create responsive designs that adapt to various devices like smartphones, tablets, and desktops.

In a **Mobile First** approach, **min-width** is more helpful. This approach starts by designing for the smallest screen size (mobile) and then gradually adding styles for larger screens (tablets, desktops) as needed.

When use First mobile approach we will use Min-width, its much better design helping for me and desktop first approach we use **Max-width**

Min-width vs. Max-width

Min-width: Specifies styles for screen widths that are *larger* than the defined value. In a mobile-first design, this is often used because you start by defining the base styles for smaller screens and then add media queries to adjust the layout for larger devices (tablets, desktops).

Example:

```
@media screen and (min-width: 500px) {}
```

Anything > 500px

Max-width: Specifies styles for screen widths that are *smaller* than the defined value. This is generally used when you're focusing on designing for larger screens first (desktop-first approach) and then adding media queries to adjust for smaller devices.
Example:

```
@media screen and (max-width: 500px) {}
```

Anything < 500px

In a **Mobile-First Approach**, using min-width is more helpful because:

- **Default Styles for Mobile**: We write the default styles assuming the user is on a mobile device. This ensures the page loads faster for mobile users, without needing extra conditions to target smaller screens.
- **Progressive Enhancement**: We apply media queries with min-width to add enhancements for larger screens, such as tablets and desktops. This keeps the design simple and focused on essential features for mobile devices, while gracefully adapting to larger displays.

Q10: What is float property and why do we use clear property with float?
Answer:

The **float** property in CSS is used to position an element to the left or right within its container, allowing other elements to wrap around it.

The float property can take these values:

```
h2{
   /* The element floats to the left of its container, and content wraps
around it on the right. */
   float: left;
   /* The element floats to the right of its container, and content wraps
around it on the left. */
   float: right;
   /*  (default): The element does not float. */
   float: none;
   /* Inherits the float value from its parent. */
   float: inherit;
}
```

Mohammad Bayajeed
bayajeedbd@gmail.com

The clear property is used to prevent overlapping of floated elements by forcing elements to move below the floated ones.

```css
h2{
    /* Clears content that is floated to the left, ensuring the element starts below any left-floated elements. */
    clear: left;
    /* Clears content that is floated to the right. */
    clear: right;
    /* Clears both left- and right-floated content, ensuring the element starts below all floated */
    clear: both;
    /* No clearing is applied. */
    clear: none;
}
```

Project: https://66e85dc662ed92332eebb478--dynamic-vacherin-4128f6.netlify.app/