Assignment No. 1

Q1] What is responsive web page design & how to use it?

-> Responsive web design

- Responsive web design is used to make your web page look appropriate, good and well placedon all device (desktop, tablet, smartphone, etc.).
- Responsive web design uses HTML & CSS to resize, hide, shrink, enlarge or move the content.
- It makes the content look good on any screen.
- Responsive web design is not a program or a Javascript.

* To use responsive web design:

- Plan Your Design:
  Sketch out how you want your website to look on different devices like desktop, tablets & smartphones.

- Use Flexible Layout:
  Instead of fixed sizes, design your website using flexible grids that can adjust based on the screen size.

- Media Queries:
  Use CSS media queries to apply different style based on the devices screen size, ensuring optimal viewing & usability.

Scanned with OKEN Scanner

+ Test Across Devices

Test your website on various device to make sure it looks & works well on each one.

Q.2] How to check web page is responsive or not?

-> - Open the web page in a browser on your computer.
- Resize the browser window by dragging its edges!
Make the window smaller by dragging its edges.
- If the content adjust & fits neatly as you make the window smaller, it's responsive.
- If the content gets cut off, overlaps, or become difficult to read, it's not responsive.
- You can also try this on different devices like smartphones or tablets to see how the page behaves.

Q.3] What are the benefits of responsive web page design?

->

- Better User Experience

User can easily access & navigate your website on any device, making their experience smooth & enjoyable.

- Improved SEO

Search engines prefer responsive websites, boosting your site's visibility and ranking in search result.

- Cost-Effectiveness

You only need to maintain one website that works on all devices, saving you time & money compared to managing multiple versions.

- Faster Page Loading

Responsive site loads quickly on all devices, reducing bounce rates & keeping users engaged.

- Easier Management

With one website to update, managing content & making changes is simpler & more efficient.

- Increased Conversion Rates

A better user experience, improved SEO, & faster page loading times can all contribute to higher conversion rates.

Responsive design helps ensure that visitors have a positive experience on your website, increasing the likelihood that they will convert into customers or take other desired actions.

Scanned with OKEN Scanner

Q. ⊙ What is Flexbox & how does it work?

-> - CSS3 Flexible boxes also known as CSS Flexbox, is a new layout mode in CSS3.
- It is used for building responsive & flexible web layouts.
- It provides a more efficient way to arrange, align, and distribute space among elements within a container, regardless of their size or order.

* How Flexbox works?

- Flex Container

The flex container specifies the properties of the parent. It is declared by setting the display property of an element to either 'flex' or 'inline-flex'.

- Flex item

Any direct children of the flex container become flex items. These items can be any HTML element like divs, spans, paragraph etc.

Flex items can be horizontally aligned, vertically aligned, or both, depending on how you configure the flex container.

- Flex Direction

The flex-direction property is used to set the direction of the flexible items inside the flex container.
- Its default value is row (left-to-right, top-to-bottom).

The other possible values are:
- row-reverse
- column
- column-reverse

These values allowing you to create horizontal or vertical layouts as needed.

- Flex-wrap
- The flex-wrap property specifies whether the flex items should wrap or not, in the case of not enough space for them on one flex line.
- Its possible values are
- nowrap: It is the default value. The flexible items will not wrap.
- wrap: It specifies that the flexible items will wrap, if necessary.
- wrap-reverse: It specifies that the flexible items will wrap, if necessary in reverse order.

- Flexbox Justify-Content
- The justify-content property is used to define the alignment along the main axis.
- It sets the Flexbox container items horizontally when the items don't use all the available space on the main axis.

Scanned with OKEN Scanner

- Its possible values are
- flex-start
It is the default value. It sets the items at the begining of the container
- flex-end
It sets the items at the end of the container.
- center
It sets the items at the center of the container.
- space-between
It sets the items with space between the lines.
- space-around
It sets the items with space before, between, and after the lines.

- Flexbox align-items
- The flexbox align-items property is used to set the flexible container's item's vertically align when the items do not use all available space on the cross axis.
- Its possible values are
- stretch -
It is the default value. It specifies that items are stretched to fit the container.
- flex-start
It sets the items at the top of the container.
- flex-end
It sets the items at the bottom of the container.
- center
Sets the items at the center of the container (vertically).
- baseline
Sets the items at the baseline of the container.

- Flexbox 'align-content'
- The align-content property is used to modify the behavior of the flex-wrap property.
- It is just like align items, but it aligns flex lines instead of flex items.

Q.5] How to you reverse the order of flex items in row?
-> To reverse the order of flex items in a row using CSS Flexbox, you can use the 'flex-direction' property set to 'row-reverse'. Here's how you can do it:

- CSS File
```
.container{
  display: flex;
  flex-direction: row-reverse;
}
```
In this example, '.container' is the class of the container element that holds the

flex items. Setting 'flex-direction' to 'row-reverse' will reverse the order of the flex items along the main axis (horizontal axis in the case of 'row').

Here's a basic HTML structure to illustrate its usage:

- html file

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

In this example, the flex item with content "1", "2", & "3" will be displayed in reverse order horizontally within the container.

Q.5] Explain the difference between flex grow, flex shrink & flex basis with example?

=> 'flex-grow', 'flex-shrink', & 'flex-basis' are three properties used to control the sizing & behavior of flex items within a flex container.

1. Flex-grow

The flex-grow property determines how much the item will grow relative to the rest of the flexible items inside the same container if there is extra space available.

- It accepts a unitless number, where a value of 0 means the item won't grow, while a value greater than 0 specifies the proportion it will grow compared to other flex items.

- Example

```
.flex-item{
    flex-grow: 1 /* This item can grow to fill
}                available space */
```

2. Flex-shrink

- The flex-shrink property specifies how the item will shrink relative to the rest of the flexible items inside the same container.
- It accepts a unitless number, where 0 means the item won't shrink, while a value greater than 0 indicates the factor by which the item should shrink relative to others.

- Example

```
.flex-item{
    flex-shrink: 1; /* Each flex item will shrink
}                   at an equal rate */
```

3. Flex-basis

- The flex-basis property specifies the initial length of a flexible item.
- It is along with the main axis before free space is distributed according to the

'flex-grow', and 'flex-shrink' properties.
- It can be set as a length, a percentage, or the keyword 'auto'.

- Example:

```
.flex-item{          */Initial width of flex
  flex-basis: 100px;   items */
}
```

We can use the shorthand property 'flex' to set 'flex-grow', 'flex-shrink', and 'flex basis' in one line. Here's the syntax:

- Syntax:

```
flex: [flex-grow] [flex-shrink] [flex-basis]
```

Q.7] How can we create a responsive navigation menu using flex box?

-> To create a responsive navigation menu using Flexbox, we can use the following approach:

- Html code

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

- CSS code

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

nav{
  background-color: #333;
  padding: 1vh;
  display: flex;
  flex-wrap: wrap;
}

a{
  flex-grow: 1;
  flex-shrink: 1;
  flex-basis: auto;
  text-align: center;
  color: #fff;
  text-decoration: none;
  padding: 1vh;
}
```

Q.8] Explain the concept of flexbox model & its component?

-> The Flexbox (Flexible Box) layout model in CSS is designed to provide a more efficient way to layout, align, & distribute space among items within a container, even when their size is unknown or dynamic.

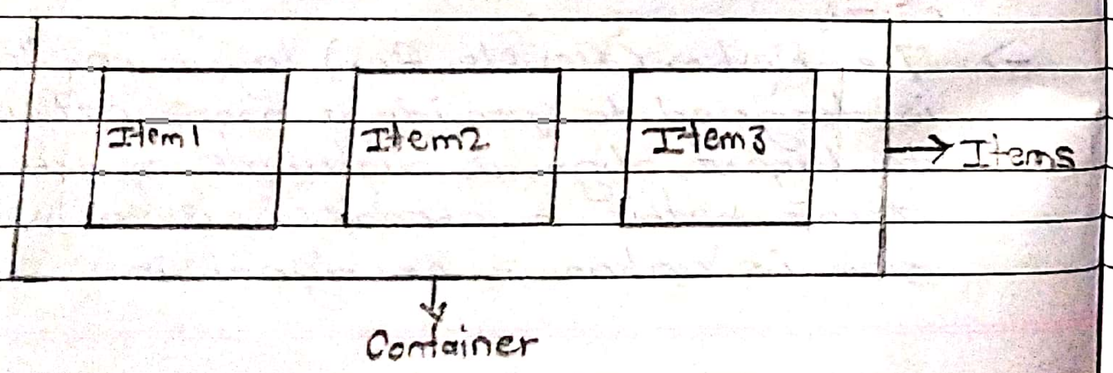
Scanned with OKEN Scanner

- Flexbox introduces a set of properties that allow you to control the behavior of the container & its items.

- Component of Flexbox model

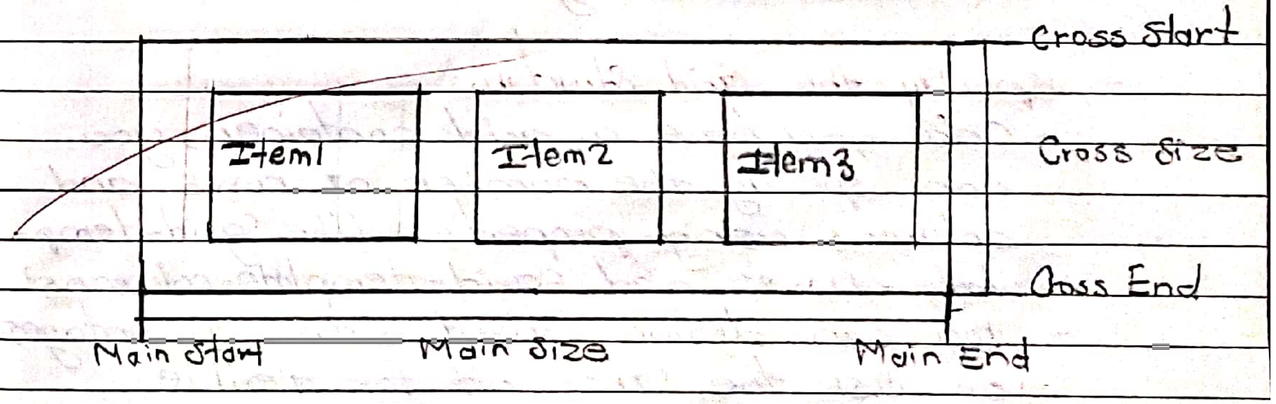
1. Flex container ('display: flex;'):
   - The parent elements that contains the flex items. is called a flexcontainer
   - By setting the 'display' property of an element to 'flex', you create a flex container.

2. Flex Items:
   - The children of flex container.
     The item inside the container "parent" are flex items.
   - There are individual flex elements inside the flex container that are laid out using the Flexbox model.
   - By default, flex items are laid out in a row.
   - We can control the size, order, & alignment of flex items using Flexbox properties.



3. Flex Axes
   - Main Axis
     - By Default, the main axis runs from left to right.
     - Main Start: The start of the main axis is called Main Start.
     - Main Size: The length bet^n Main Start & Main End is called Main Size.
     - Main End: The endpoint is called Main End.

   - Cross Axis
     - By Default, the cross axis runs from top to bottom.
     - Cross Start: The start of the Cross axis is called Cross Start.
     - Cross Size: The length bet^n Cross Start & Cross End is called Cross Size.
     - Cross End: The endpoint is called cross End.



Q.10] What is CSS grid layout & how its works?
-> The CSS grid layout module offers a grid-based layout system, with rows & columns,

Scanned with OKEN Scanner

making it easier to design web pages without having to use floats and positioning.

CSS grid layout is a powerful layout system that allows you to design web pages using a two-dimensional grid.

With CSS Grid, you can divide a webpage into rows & columns & then place elements within those rows & columns.

* CSS Grid works
- Define a grid container
  - We start by defining a grid container using the 'display: grid' property in CSS.
  - This turns an HTML element into a grid container, which acts as the parent element for the grid items.

- Specify the Grid Structure
  - Once you have a grid container, you can specify the number of rows and columns using properties like 'grid-template-rows' and 'grid-template-columns'.
  - You can define fixed sizes, percentages, or use the 'fr' unit for flexible sizing.

- Place grid items
  - After defining the grid structure, you can place grid items (HTML elements) within the grid container.
  - You can use properties like 'grid-row' & 'grid-column' to specify where each item should be placed in the grid.

- Control Spacing & alignment
  - CSS Grid provides various properties for controlling the spacing & alignment of grid items, such as 'grid-gap', 'justify-items', and 'align-items'.

Q. III List out justify content property & explain any two property with example?

→ Justify content property list
- flex-start
- flex-end
- center
- space-between
- space-around
- space-evenly

- 'justify-content: center;':
  It is used to align items are centered along the line. (main axis of the flex container).

- Example:

```
.container{
    display: flex;
    justify-content: center;
}
```

In this example, all the items within the container will be centered along the main axis of the container.

- 'justify-content: space-between':

This property evenly distributes the items within the container with the first item placed at the start & the last item at the end; and equal space between each item. For Example.

```
.container {
  display: flex;
  justify-content: space-between;
}
```

Q.12] What is a align-self property in flexbox & how it is use?

→ The align-self property specifies the alignment in the block direction for the selected items inside a flexbox or grid container.

It makes possible to override the align-items value for specific flex items.

Here's how you use align-self:

```
.grid-container {
  display: grid;
}
.grid item {
  align-self: center;
}
```

In this example, the .grid-item will be vertically centered within its grid cell, regardless of the alignment set for the .grid container.

You can use other values such as start, end, stretch or baseline for diff" alignment options.

Date 11/08/24.