Disclaimer

The information and content provided in this presentation are for educational purposes only. Delta Tech Info and its affiliates do not endorse or promote any specific products or services mentioned in this presentation. The viewers are advised to exercise their own judgment and discretion while applying the knowledge gained from this presentation.

Delta Tech Info

Delta Tech Info is a platform dedicated to spreading digital literacy among the Pakistani youth. Our aim is to equip individuals with the knowledge and skills necessary to navigate the digital world effectively. Through our courses and educational resources, we strive to empower the youth to harness the power of technology and enhance their opportunities in today's digital age."

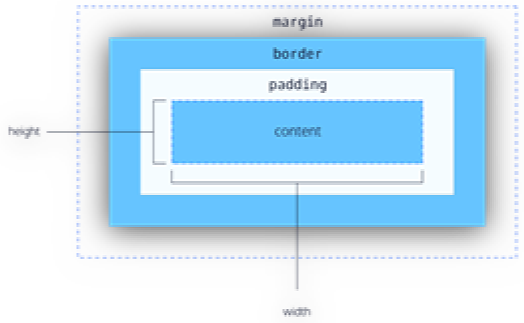
Lecture 3

- Understanding the CSS Box Model (margin, padding, border, content)
- CSS Flex and Flexbox Display
- Creating columns and grids with CSS
- Introduction to responsive design principles
- Media queries for different screen sizes and devices
- Adapting layouts using CSS for better user experience on mobile and desktop

Understanding the CSS Box Model (margin, padding, border, content)

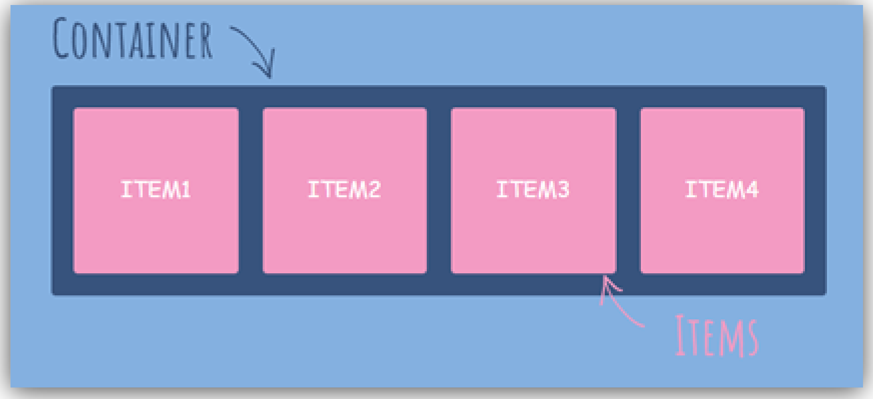
The CSS box model is a fundamental concept that describes how elements are structured and spaced within a web page. It consists of four main components: content, padding, border, and margin. Understanding the box model is essential for controlling the layout and spacing of elements using CSS.

1. **Content**: The content area represents the actual content, such as text or images, displayed within an element. Its size is defined by the width and height properties.
2. **Padding:** Padding is the space between the content and the element's border. It provides internal spacing within the element.
3. **Border:** The border is a line that surrounds the padding and content of an element. It separates the content from the margin and can be customized in terms of style, color, and thickness.
4. **Margin:** The margin is the space between an element's border and adjacent elements. It creates space around the element and helps control the overall spacing between elements on a page.



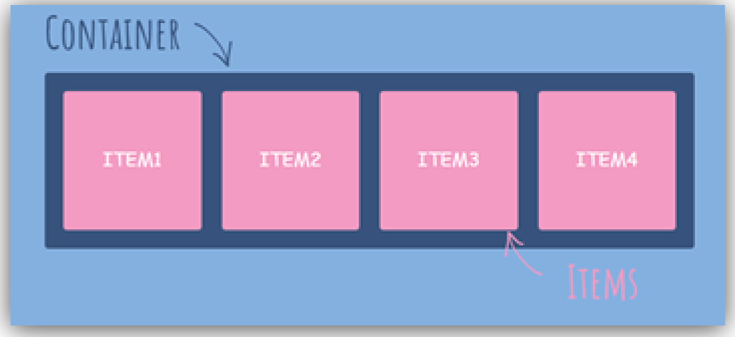
CSS Flex and Flexbox Display

CSS Flexbox is a layout module that provides a flexible way to arrange and align elements within a container. It introduces the concept of flex containers and flex items, allowing for responsive and dynamic layouts.



CSS Flex and Flexbox Display Main Components

1. **Flex Container:** A flex container is an element that serves as the parent container for a group of flex items. To create a flex container, you need to apply the display: flex; or display: inline-flex; property to the container element. This activates the flexbox behavior for its children.
2. **Flex Items:** Flex items are the child elements of a flex container. They can be any HTML element within the container. Flex items are laid out along a flex container's main axis and can be configured to stretch, shrink, or grow based on available space.
3. **Main Axis and Cross Axis:** Flexbox introduces two axes: the main axis and the cross axis. The main axis runs horizontally or vertically, depending on the flex-direction property of the flex container. The cross axis is perpendicular to the main axis.
4. **Flex Direction:** The flex-direction property determines the direction in which flex items are placed within the flex container. It can be set to row (default), row-reverse, column, or column-reverse.

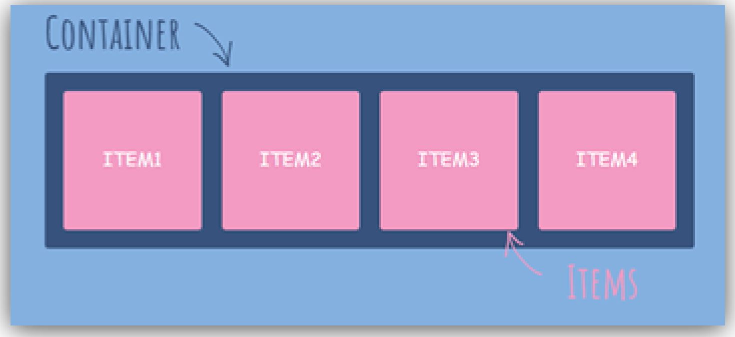


CSS Flex and Flexbox Display Main Components

Flex Wrapping: By default, flex items are laid out in a single line. However, the flex-wrap property allows items to wrap onto multiple lines if there isn't enough space. It can be set to nowrap (default), wrap, or wrap-reverse.

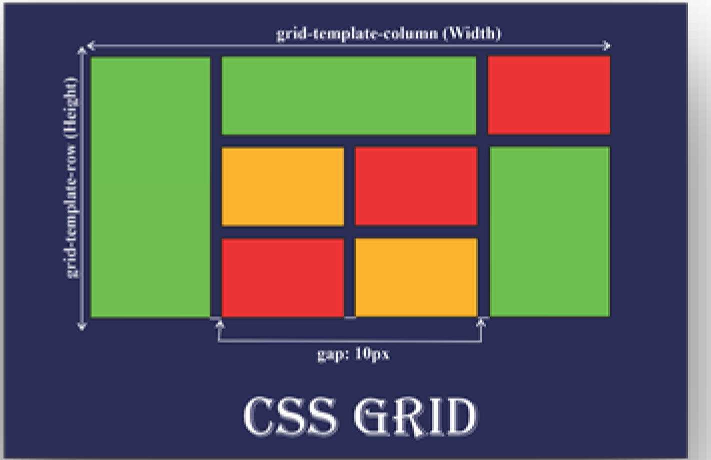
Justify Content: The justify-content property defines how flex items are distributed along the main axis of the flex container. It controls the alignment and spacing between items. Values include flex-start, flex-end, center, space-between, space-around, and space-evenly.

Align Items and Align Content: The align-items property controls the alignment of flex items along the cross axis. It can be set to flex-start, flex-end, center, baseline, or stretch. The align-content property is similar but applies to multiple lines of flex items. It can be used when wrapping occurs.



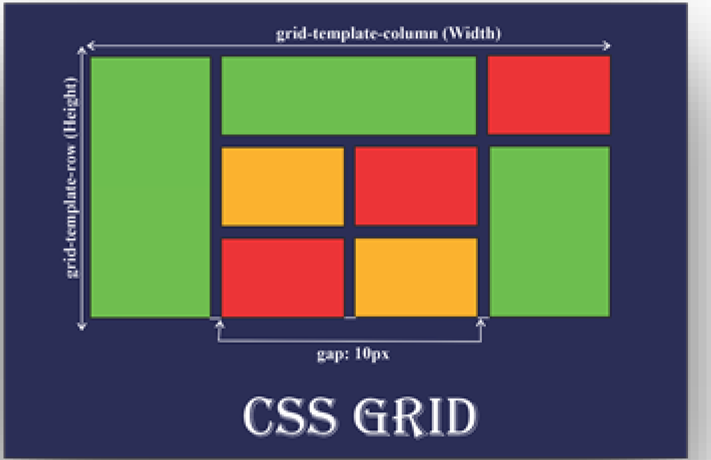
Creating columns and grids with CSS

CSS Grid is a powerful layout module that allows you to create grid-based layouts with columns and rows. It provides a flexible way to arrange and align elements within a container.



Creating columns and grids with CSS

1. **Grid Container:** To create a grid layout, you need to define a grid container. This is achieved by applying the display: grid; property to the container element.
2. **Define Columns:** Use the grid-template-columns property on the grid container to define the size and number of columns in the grid. You can specify the width of each column using values such as fixed lengths (pixels, percentages) or flexible units (fr).
3. **Create Grid Items:** Place elements within the grid container to create grid items. These can be any HTML elements. By default, grid items will flow into the grid's cells in the order they appear in the HTML markup.
4. **Adjust Column Placement and Spanning:** To control the placement and spanning of grid items across columns, you can use the grid-column-start, grid-column-end, and grid-column properties. These allow you to specify the start and end positions of a grid item within the grid columns.
5. **Alignment and Spacing:** CSS Grid provides properties for aligning and spacing grid items within columns. You can use justify-items to align items horizontally within the column, and align-items to align items vertically. Additionally, you can set

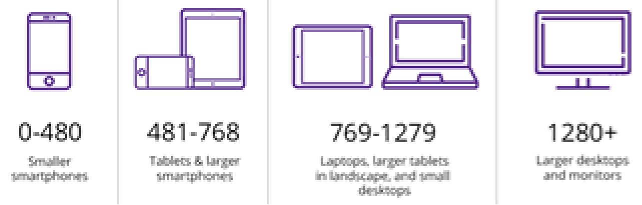


Introduction to responsive design principles

Responsive design is an approach to web design aimed at creating websites that provide an optimal viewing and user experience across a wide range of devices and screen sizes. It involves adapting the layout, content, and functionality of a website to ensure it is usable and visually appealing on various devices, including desktops, laptops, tablets, and smartphones.

Media queries for different screen sizes and devices

Media queries are an essential component of responsive web design, as they allow you to apply different styles and layout adjustments based on the characteristics of the device or screen size. By using media queries, you can create a customized user experience for various devices and ensure that your website looks and functions well across different screen sizes. Here are some common media queries used for different screen sizes and devices:


0-480
Smaller smartphones
481-768
Tablets & larger smartphones
769-1279
Laptops, larger tablets in landscape, and small desktops
1280+
Larger desktops and monitors

Media queries for different screen sizes and devices

Mobile Devices

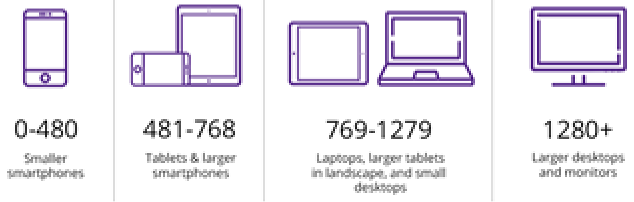
CSS Code

```
 @media (max-width: 767px) {
  /* Styles for small screens (e.g., smartphones) */
}
```

Tablets:

CSS Code

```
@media (min-width: 768px) and (max-width: 1023px)
{
  /* Styles for medium-sized screens (e.g., tablets) */
}
```

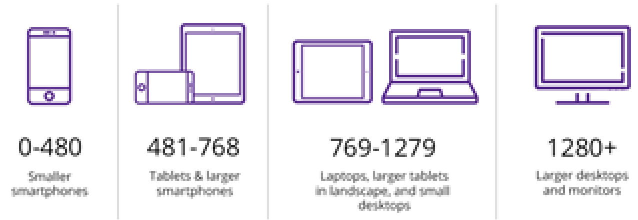

0-480
Smaller smartphones
481-768
Tablets & larger smartphones
769-1279
Laptops, larger tablets in landscape, and small desktops
1280+
Larger desktops and monitors

Media queries for different screen sizes and devices

Laptops and Desktops:

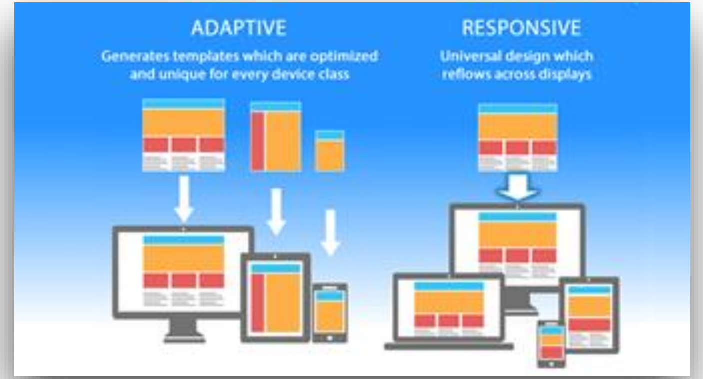
CSS Code

```
@media (min-width: 1024px) {
 /* Styles for large screens (e.g., laptops and desktops) */
}
```


0-480
Smaller smartphones
481-768
Tablets & larger smartphones
769-1279
Laptops, larger tablets in landscape, and small desktops
1280+
Larger desktops and monitors

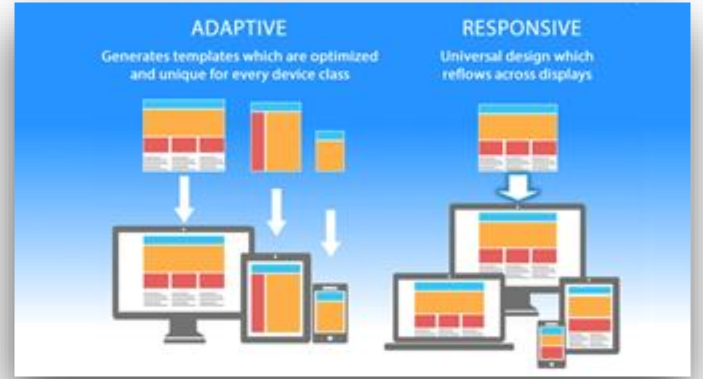
Adapting layouts using CSS for better user experience on mobile and desktop

To adapt layouts using CSS for a better user experience on both mobile and desktop devices, you can utilize various CSS techniques and properties.


ADAPTIVE
Generates templates which are optimized and unique for every device class
RESPONSIVE
Universal design which reflows across displays

Adapting layouts using CSS for better user experience on mobile and desktop

1. **Responsive Layouts:** Use CSS media queries to adjust the layout based on different screen sizes. Modify the positioning, sizing, and arrangement of elements to provide an optimal experience. Consider employing a mobile-first approach, where you design for smaller screens first and gradually enhance the layout for larger screens.
2. **Flexible Units and Grids:** Use relative units like percentages or em instead of fixed pixels for widths, heights, and margins. Employ CSS Grid or Flexbox to create flexible and responsive layouts that adapt to different screen sizes.
3. **Hiding and Displaying Content:** Employ CSS techniques to hide or show specific content based on screen size. You can use the display property (display: none;) or CSS classes to control the visibility of certain elements on different devices.


ADAPTIVE
Generates templates which are optimized and unique for every device class
RESPONSIVE
Universal design which reflows across displays

Questions