CSS

Reading Material

PW SKILLS

Topics

- Understanding the concept of CSS
  - Basic CSS Syntax
  - CSS Selectors
  - Box Model
  - Display
    - Block
    - Inline
    - Inline-block
    - none
  - Colors and Backgrounds
  - Text and font property
  - Layout and Positioning
  - Responsive
  - Animation, Transition, Transform, and Filter
- Interview Questions
  - Easy
  - Medium
  - Heard
- Multiple choice questions

Basic css syntax

- A CSS ruleset consists of a selector and a declaration block.
- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.

```
selector {
    property: value;
    property: value;
}
```

Specificity: Determines which styles apply when multiple rules conflict:
- Inline styles
- ID selectors
- Class, attribute, pseudo-class selectors
- Element, pseudo-element selectors

CSS Selectors

- **CSS selectors are patterns that are used to select and style HTML elements. Here's a summary of some common CSS selectors:**

  - **Element Selector:** Selects all elements of a specific type. It targets HTML elements by their tag names.
  - **Class Selector:** Selects elements with a specific class attribute. It targets HTML elements by their class names.
  - **ID Selector:** Selects a single element with a specific id attribute. It targets HTML elements by their id attribute.
  - **Descendant Selector:** Selects an element that is a descendant of another specified element. It targets an element that is nested inside another element.
  - **Child Selector:** Selects an element that is a direct child of another specified element.
  - Adjacent Sibling Selector: Selects an element that is directly after another specified element.
- **Attribute Selector:** Selects elements based on their attribute values.

```
// Element Selector
p { /* Styles applied to all <p> elements */   }

 // Class Selector
.highlight {  /* Styles applied to all elements with
class="highlight" */
}

// ID Selector
#header { /* Styles applied to the element with id="header" */ }

// Descendant Selector
div p {  /* Styles applied to all <p> elements inside <div>
elements */
}

// Child Selector
ul > li {    /* Styles applied to all <li> elements that are
direct children of <ul> elements */
}

// Adjacent Sibling Selector
h2 + p {    /* Styles applied to <p> element that directly follows
an <h2> element */
}

// Attribute Selector
input[type="text"] {   /* Styles applied to <input> elements with
type="text" */ }
```

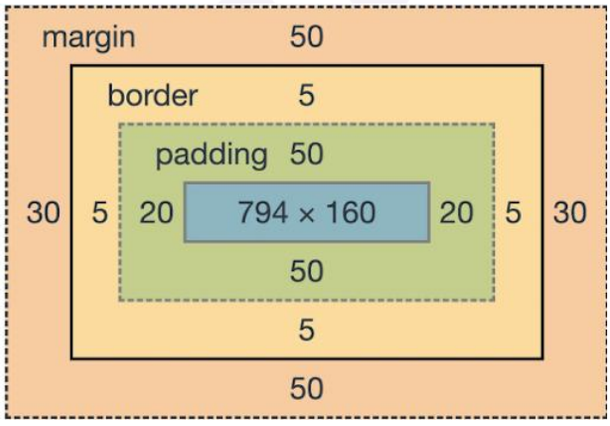
- **Pseudo-classes and Pseudo-elements:** These selectors target elements based on their state or position in the document, such as
- **:hover :** Applies styles when the user designates an element (typically by placing the mouse pointer over it).
- **:active :** Applies styles when an element is being activated by the user (such as when it's being clicked).
- **:first-child :** Selects an element that is the first child of its parent.
- **:nth-child(n) :** Selects an element that is the nth child of its parent, where n can be a number, keyword, or formula.

```
a:hover {
    /* Styles applied to <a> elements when hovered */
}
p:first-child {
    /* Styles applied to the first <p> element in its parent */
}
p::first-letter {
    /* Styles applied to the first letter of each <p> element */
}
li:nth-child(1) { font-size: larger; }
```

```
a:hover {
    /* Styles applied to <a> elements when hovered */
}
p:first-child {
    /* Styles applied to the first <p> element in its parent */
}
p::first-letter {
    /* Styles applied to the first letter of each <p> element */
}
li:nth-child(1) { font-size: larger; }
```

Box Model

The CSS box model is a container that contains multiple properties, including borders, margins, padding, and the content itself. It is used to create the design and layout of web pages, According to the CSS box model, the web browser supplies each element as a square prism.

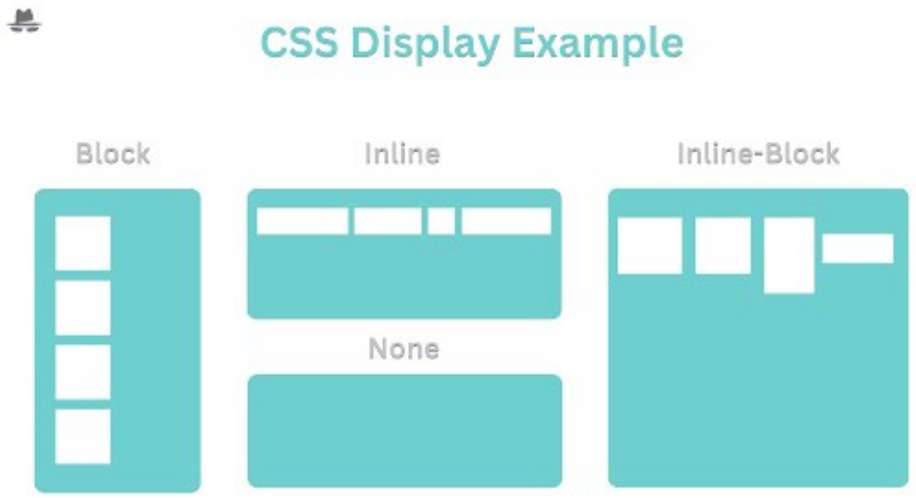


Properties of the box model

- **Content:** The content area consists of content like images, text, or other forms of media content, the height and width properties help to modify the box dimensions.
- **Padding:** The padding area is the space around the content area and within the border box. It can be applied to all sides of the box or to the specific, selected side(s) - top, right, bottom, and/or left.
- **Margin:** The margin area consists of space between the border and the margin, the margin does not possess its own background color and is completely transparent, it shows the background color of the element, like the body element.
- **Border:** The border area surrounds the padding and the content, and can be applied to all the sides of the box or to selected sides/side top, right, bottom, and left.

Display

- **Block:** Creates a block-level element, generating line breaks before and after the element in Normal Flow.
- **Inline:** Creates an inline-level element that does not generate line breaks, and starts on the same line in Normal Flow.
- **Inline-block:** The display inline-block property in CSS is used to make an element behave like an inline element while retaining some features of a block-level element. It combines characteristics of both inline and block elements.
- **none:** The display inline-block property in CSS is used to make an element invisible, and the element will not take up any space in the document.



Color and background

you can specify the color of text, backgrounds, borders, and other elements using various properties. Here's how you can set colors and backgrounds:
Text Color:
You can set the color of text using the color property. You can use color names, hexadecimal codes, RGB, RGBA, HSL, HSLA values, or other color formats.

```
p {
  color: red; /* Using color name */
}

h1 {
  color: #00ff00; /* Using hexadecimal color code */
}

span {
  color: rgb(255, 0, 255); /* Using RGB */
}

div {
  color: rgba(255, 0, 0, 0.5); /* Using RGBA with alpha
transparency */
}

a {
  color: hsl(120, 100%, 50%); /* Using HSL */}
```

Background Color: To set the background color of an element, you use the background-color property.

```
body {
  background-color: #f0f0f0; /* Setting background color of the
body */
}

.container {
  background-color: rgba(0, 0, 255, 0.2); /* Setting background
color of a container with transparency */
}
```

Background Image: Instead of a solid color, you can also use an image as the background.

```
.element {
  background-image: url('path/to/image.jpg');
  background-repeat: no-repeat; /* Specify whether and how a
background image should be repeated */
  background-size: cover; /* Cover the entire element */
}
```

Background Gradient: You can create gradient backgrounds using the background-image property with the linear-gradient() or radial-gradient() function.

```
.element {
  background-image: linear-gradient(to right, red, yellow); /*
Horizontal gradient from red to yellow */
}
```

Font and text property

Font property: Font properties specify the style, size, weight, and family of the text. Examples include

- **font-family:** Sets the font for an element.
- **font-size:** Specifies the size of the font.
- **font-weight:** Controls the thickness of the font.
- **font-style:** Defines the style of the font (normal, italic, oblique).
- **font-variant:** Controls small-caps usage.
- **font:** Shorthand for setting multiple font properties at once.

```
body {  font-family: "Arial", sans-serif; } // font-family
h1 { font-size: 24px; } // font-size
p {  font-weight: bold; } // font-weight
em { font-style: italic; } // font-style
h2 {  font-variant: small-caps; } // font-variant
.title { font: italic small-caps bold 16px/1.5 "Arial", sans-
serif; } //font
```

Text Properties: Font properties specify the style, size, weight, and family of the text. Examples include

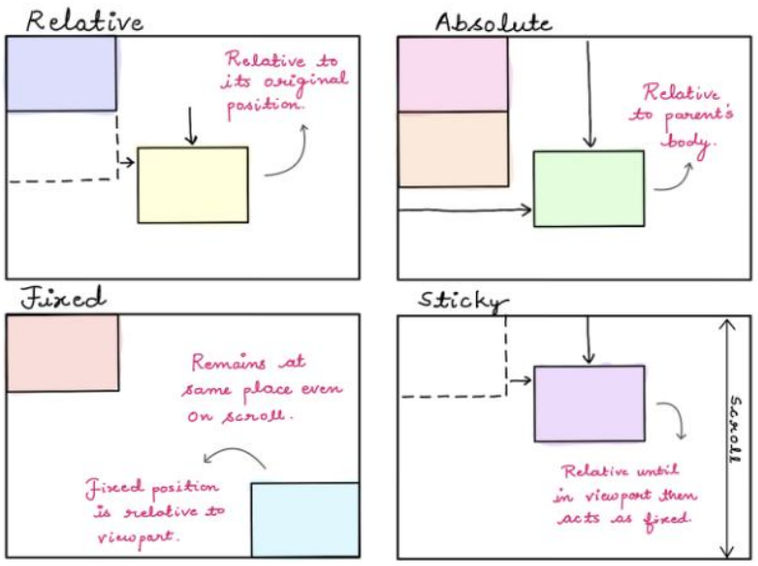
- **color:** Sets the color of the text
- **text-align:** Aligns text horizontally.
- **text-decoration:** Adds decoration to text.
- **text-indent:** Indents the first line of text.
- **line-height:** Sets the height of lines of text.
- **letter-spacing:** Adjusts space between characters.
- **word-spacing:** Adjusts space between words.
- **text-shadow:** Adds shadow to text.

```
p {  color: #333;   } // color
h1 { text-align: center;} //text-align
a { text-decoration: underline; } // text-decoration
h2 { text-transform: uppercase; } //text-transform:
p { text-indent: 30px; } // text-indent
p { line-height: 1.5; } // line-height
h1 { letter-spacing: 2px; } // letter-spacing
p {word-spacing: 4px; } //word-spacing
h1 { text-shadow: 2px 2px 4px #999; } // text-shadow
```

Layout Positioning

Position involves determining the specific placement of elements
within the layout. In web development, this is often controlled through CSS. Key positioning techniques include:

- **static:** The default position; the element is positioned according to the normal flow of the document.
- **relative:** The element is positioned relative to its normal position.
- **absolute:** The element is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- **fixed:** The element is positioned relative to the browser window.
- **sticky:** The element is treated as relative until its containing block crosses a specified threshold, then it is treated as fixed.



Flex

The Flexible Box Layout Module, which is popularly known as Flexbox, is a powerful way of designing an efficient and responsive layout in CSS
Here are some of the most important properties of Flexbox

- **Container Properties**
  - **display:** Defines a flex container (flex) or inline flex container (inline-flex).
  - **flex-direction:** Specifies the direction of flex items (row, row-reverse, column, column-reverse).
  - **flex-wrap:** Determines whether flex items wrap onto multiple lines (nowrap, wrap, wrap-reverse).
  - **flex-flow:** A shorthand for setting both flex-direction and flex-wrap.
  - **justify-content:** Aligns flex items along the main axis (start, end, center, space-between, space-around, space-evenly).
  - **align-items:** Aligns flex items along the cross axis (stretch, flex-start, flex-end, center, baseline).
  - **align-content:** Aligns flex lines within a container when there is extra space in the cross axis (stretch, start, end, center, space-between, space-around).

- **Item Properties**
  - order: Specifies the order of a flex item relative to others.
  - flex-grow: Defines the ability of a flex item to grow relative to others.
  - flex-shrink: Defines the ability of a flex item to shrink relative to others.
  - flex-basis: Specifies the initial size of a flex item before space is distributed.
  - flex: A shorthand for setting flex-grow, flex-shrink, and flex-basis.
  - align-self: Overrides the container's align-items value for individual flex items (auto, start, end, center, baseline, stretch).

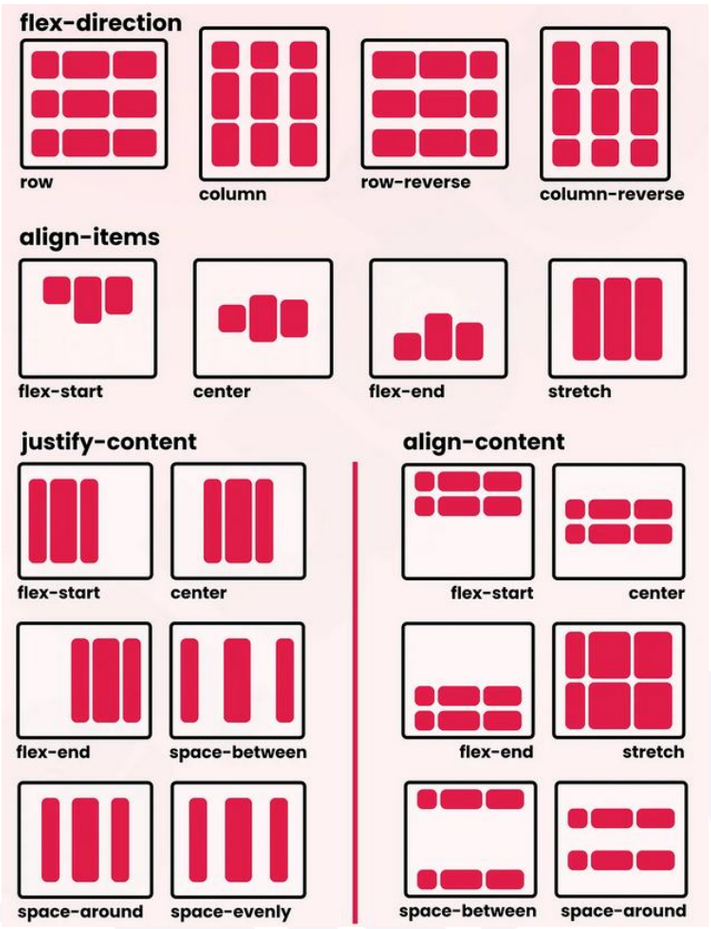
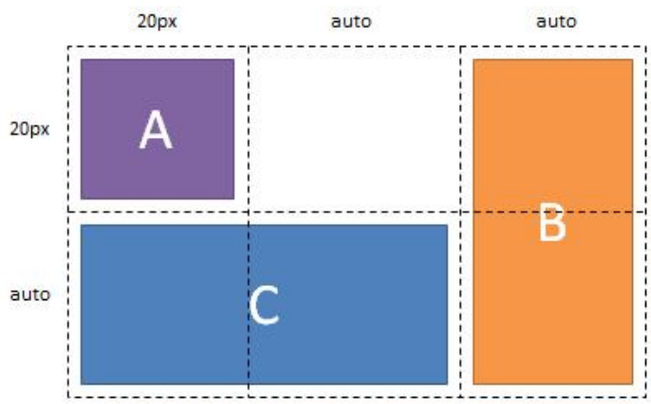


Fig. Flex overview

Grid



CSS Grid is a powerful two-dimensional layout system that allows you to create complex and flexible layouts with rows and columns, making it easier to design web pages without having to use floats and positioning. Here are some of the most important properties

- **Container Properties:**
  - **Container Definition:**
    - **display: grid;** - Establishes an element as a grid container.
    - **grid-template-columns** - Defines the size and number of columns in the grid layout.
    - **grid-template-rows** - Defines the size and number of rows in the grid layout.
    - **grid-template-areas** - Specifies the layout of grid areas using named grid areas.
    - **gap** - Sets the gap (gutter) between grid rows and columns.

  - **Implicit Grid:**
    - grid-auto-rows - Specifies the size of implicitly created rows.
    - grid-auto-columns - Specifies the size of implicitly created columns.
  - grid-auto-flow - Controls how auto-placed items are inserted into the grid.

- **Item Properties:**
  - **Positioning:**
    - **grid-column** - Specifies the start and end positions of a grid item within columns.
    - **grid-row** - Specifies the start and end positions of a grid item within rows.
    - **grid-area** - Specifies a grid item's position and size within the grid.

- **Alignment:**
  - **justify-self** - Aligns a grid item along the inline (row) axis within its grid area.
  - **align-self** - Aligns a grid item along the block (column) axis within its grid area.
  - **place-self** - A shorthand for setting both justify-self and align-self for a grid item.

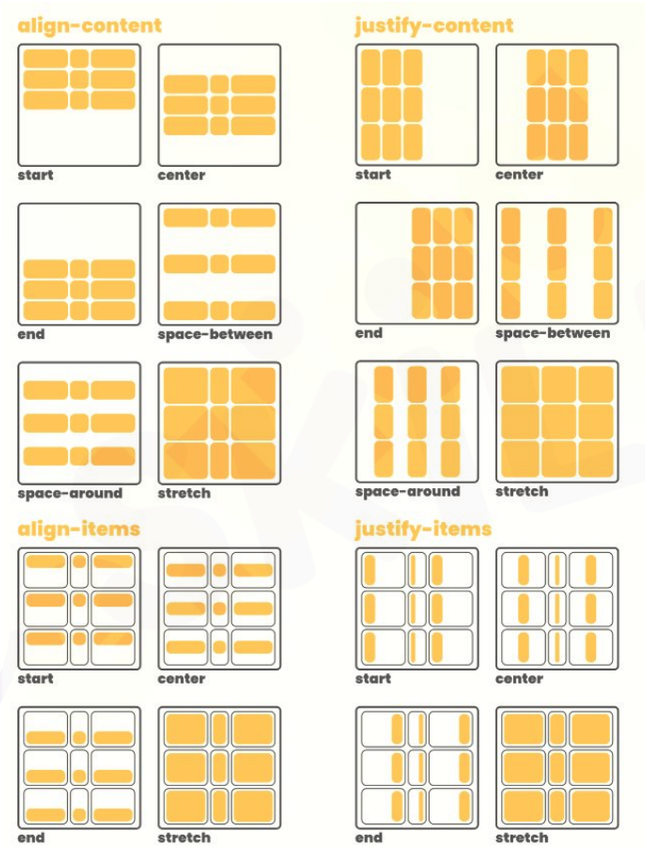
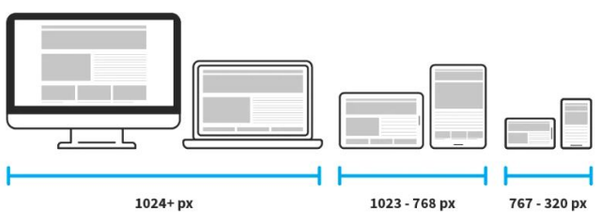


Fig. Grid overview

Responsive

Responsive Web Design (RWD) is a web design approach that ensures websites provide an optimal viewing and user experience across all devices and screen sizes, including smartphones, tablets, and desktops. RWD aims to make websites adaptable and accessible on various platforms.



- **Key Points:**
  - Adaptive Layouts: Websites automatically adjust to fit the screen size and orientation of the device.
  - Seamless Experience: Ensures websites are visually appealing and easy to navigate on any device.
- **Advantages of RWD:**
  - Improved User Experience: Consistent and user-friendly experience across devices.
  - Increased Mobile Traffic: Accommodates a larger audience using mobile devices.
  - Cost and Time Efficiency: A single codebase for all devices reduces development and maintenance costs.
  - SEO Benefits: Favoured by search engines, leading to better search rankings and visibility.

- **What is the media query in CSS?**

Media queries in CSS allow you to define different styles for different device types, screen sizes, and orientations. This is particularly useful for creating responsive web designs that adapt to different devices and screen sizes.

Animation, Transition, Transform and Filter

Animation

CSS Animations allow elements to transition between multiple styles. They can be used to create complex, multi-step animations.

Key Properties:

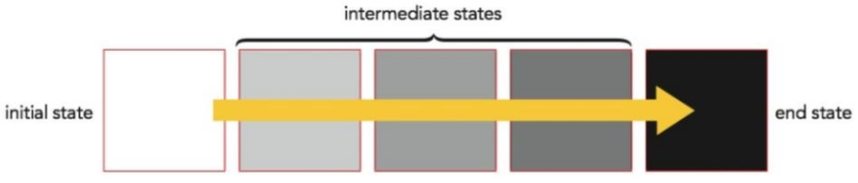
- **@keyframes:** Defines the animation's keyframes.
- **animation-name:** Name of the animation.
- **animation-duration:** Duration of the animation.
- **animation-timing-function:** Speed curve of the animation.
- **animation-delay:** Delay before the animation starts.
- **animation-iteration-count:** Number of times the animation should run.
- **animation-direction:** Direction of the animation.

Transition

CSS Transitions provide a way to make changes to CSS properties smoothly over a specified duration.

Key Properties:

- **transition-property:** Specifies the CSS property to which the transition is applied.
- **transition-duration:** Duration of the transition.
- **transition-timing-function:** Speed curve of the transition.
- **transition-delay:** Delay before the transition starts.

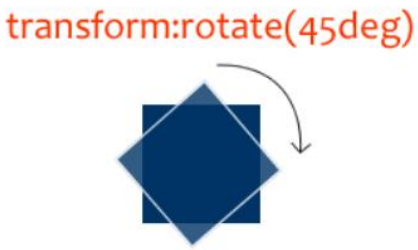


Transform

CSS Transforms allow you to move, rotate, scale, and skew elements.

Key Functions:

- **translate():** Moves an element.
- **rotate():** Rotates an element.
- **scale():** Scales an element.
- **skew():** Skews an element.



Animation, Transition, Transform and Filter

Filter

CSS Filters allow you to apply graphical effects like blur or color shifting to elements.

Key Function:

- **blur():** Applies a blur effect.
- **brightness():** Adjusts the brightness.
- **contrast():** Adjusts the contrast.
- **grayscale():** Converts the element to grayscale.
- **sepia():** Applies a sepia tone.