Andhra Pradesh State Skill Development Corporation

Skill AP
APSSDC

eSkill AP
Learn Anytime Anywhere
Andhra Pradesh State Skill Development Corporation

Web designing using ReactJs

Cascading Style Sheets

Cascading Style Sheets

Introduction

- CSS stands for Cascading Style Sheets
- Which is used to apply the beautification to the HTML document

What are we gonna discuss in the CSS concept?

- CSS Types
  - Inline
  - Internal
  - External
- CSS Selectors
  - Basic or simple selectors
    - Universal Selector
    - Type Selector
    - Class Selector
    - ID Selector
    - Attribute Selector
  - Grouping Selector
  - Combinators
    - Descendant combinator
    - Child combinator
    - General sibling combinator
    - Adjacent sibling combinator
    - Columns combinator
  - Pseudo Selector
    - Pseudo class selector
    - Pseudo element selector
- CSS Flexbox
- CSS Responsive Web Design

Syntax of CSS:

Selector {property: value;}

Kinds of CSS

CSS can be include to HTML documents in 3 ways:

- Inline - by using the style attribute inside HTML elements
- Internal - by using a <style> element in the <head> section
- External - by using a <link> element to link to an external CSS file

Inline CSS:

```
<h1 style="color:blue;"> A Blue Heading </h1>
```

Internal CSS:

We have to include css styles in style tag in head tag

```
<!DOCTYPE html>
<html>
<head>
    <title> :) Css Document (:</title>
    <style type="text/css">
        h1 {
            color: blue;
        }
    </style>
</head>
<body>
 <h1>Internal CSS</h1>
</body>
</html>
```

External CSS:

For external CSS we have to take the CSS file as separate and include that in the respective HTML file. We've to do this by using the link tag

CSS Selectors

- **Universal selector**

  Selects all elements. Optionally, it may be restricted to a specific namespace or to all namespaces.

  Syntax: * ns|* *|*

  Example: * will match all the elements of the document.

- **Type selector**

  Selects all elements that have the given node name.
  Syntax: elementname { property : value }

Example: input will match any <input> element.

body{ background: #ddd}

Here the body is the type selector, the background is the property, and #ddd (grey color) is the value

- **Class selector:**
  Selects all elements that have the given class attribute. We've denoted this with a dot(.) symbol

  Syntax: .className

  Example: .index will match any element that has a class of "index".

- **ID selector (*Identifier*):**
  Selects an element based on the value of its id attribute. There should be only one element with a given ID in a document.

  Syntax: #idName

  Example: #demo will match the element that has the ID "demo".

- **Attribute selector:**

  Selects all elements that have the given attribute.

  Syntax: [attr] [attr=value] [attr~=value] [attr|=value] [attr^=value] [attr$=value] [attr*=value]

  Example: [autoplay] will match all elements that have the autoplay attribute set (to any value).

  a[target] { background: #ddd; }

  a[target="_blank"] { background: yellow; }

- **Grouping Selector**

  The is a grouping method, it selects all the matching nodes.
  Syntax: A, B
  Example: div, span will match both and elements.

```
h1,
h2,
.heading { background: #000; color: #fff; }
```

- **Descendant combinator**
  The (space) combinator selects nodes that are descendants of the first element. The descendant selector matches all elements that are descendants of a specified element.

  Syntax: A B

  Example: div p { background-color: yellow; }

- **Child combinator**
  The > combinator selects nodes that are direct children of the first element.

  Syntax: A > B

  Example: ul > li will match all <li> elements that are nested directly inside a <ul> element.

- **General sibling combinator**
  The ~ combinator selects siblings. This means that the second element follows the first (though not necessarily immediately), and both share the same parent.

  Syntax: A ~ B

  Example: p ~ span will match all elements that follow a <p>, immediately or not.

- **Adjacent sibling combinator**
  The + combinator selects adjacent siblings. This means that the second element directly follows the first, and both share the same parent.

  Syntax: A + B

Example: h2 + p will match all <p> elements that directly follow an <h2>.

- **Column combinator**
  The || combinator selects nodes that belong to a column.

  Syntax: A || B

  Example: col || td will match all <td> elements that belong to the scope of the <col>.

- **Pseudo Selectors**
  - The pseudo allows the selection of elements based on state information that is not contained in the document tree.
    Example: a: visited will match all <a> elements that have been visited by the user.

    Pseudo-elements
    The pseudo represents entities that are not included in HTML.

    Example: p::first-line will match the first line of all **<p>** elements.

CSS FlexBox:

CSS Flexible Box Layout is a module of **CSS** that defines a CSS box model optimized for user interface design, and the layout of items in one dimension. While we working flexbox concept first you have to create a parent element with some child elements. Then we can apply flexbox layout with display: flex property. We can see child elements can align side by side.

Syntax: .parentClass { display:flex; }

Example:

For this concept, we will take the below HTML file as common for this file and we can apply flex properties one by one.

Flex.html *Style.css*

```
<!DOCTYPE html>
<html>
<head>
        <title> :) Css Document (:
    </title>

        <link rel="stylesheet"
type="text/css" href="css/style.css">

</head>

<body>
 <div class="parent">
        <div class="child">
                Child1
        </div>
        <div class="child">
                Child2
        </div>
        <div class="child">
                Child3
        </div>
        <div class="child">
                Child4
        </div>
 </div>
</body>
</html>
```

```
.parent {
        display: flex;
        background-color: #40FF00;
  }

   .child {
        width: auto;
        padding: 10px;
        margin:1%;
        color: #fff;
        background-color: #2E2EFE;
        /*#2E2EFE is hexadecimal code for
        colors*/
}
```

Output:

Flex-direction:

The flex-direction property defines in which direction the container wants to stack the flex items.It has property value row,row-reverse,column,column-reverse.

Style.css:

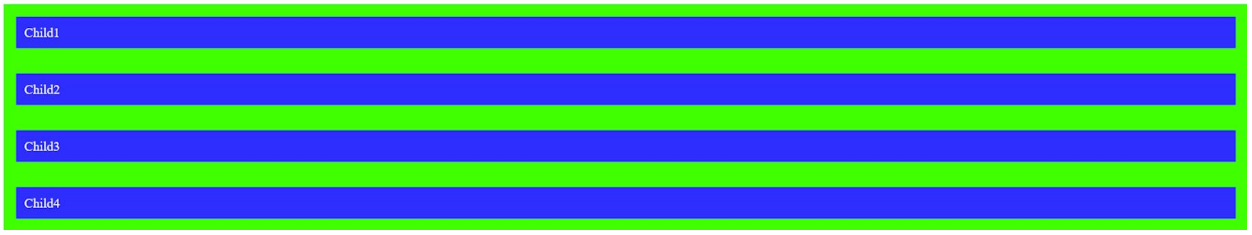
```
.parent {
        display: flex;
        background-color: #40FF00;
        flex-direction: row-reverse;
   }
```

Flex-direction:column

We can get childelements in column format

Example:

```
.parent {
        display: flex;
        background-color: #40FF00;
        flex-direction: column;
}
```



Flex-direction:column-reverse

We can get data in column reverse format

Example:

```
.parent {
        display: flex;
        background-color: #40FF00;
        flex-direction: column-reverse;
}
```

Flex-wrap:wrap:

The flex-wrap property specifies whether the flex items should wrap or not.Default is property value is nowrap.If child elements exceed it's total width we can't see those data for that we will use flex-wrap:wrap

Example:

Style.css:

```
.parent {
        display: flex;
        background-color: #40FF00;
        flex-wrap: wrap;
}
```

```
.child {
        width: auto;
        padding: 100px;
        font-size:30px;
        margin:1%;
        color: #fff;
        background-color: #2E2EFE;
}
```

Note: The child4 element we can see in the next line.

Output:

Justify-content:

This property used to align flex items with property values of the center,space-around,space-between,flex-start, and flex-end. Try to apply those properties to the parent element.

Align-items :

This property used to align flex items with different values as shown below

- Align-items:center
- Align-items:flex-start
- Align-items:flex-end
- align-items:stretch

Align-content:

This property used to align the flex lines. Let's some property values for the align-content

- Align-content:center
- Align-content:flex-start
- Align-content:flex-end
- Align-content:stretch
- Align-content:space-around
- Align-content:space-between

CSS Responsive Web Design

- Responsive Web Design make our web page look good on different devices
- Html and CSS we will make webpage as a responsive

Viewport :

The viewport is the user's visible area of a webpage and it varies with the respective device. To control viewport for different devices we will use viewport through <meta> tag

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

The above viewport we have to include in our webpage to get responsive for that webpage.

The meta tag gives the browser instructions on how to control the page's dimensions and scaling. The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device). The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

CSS Grid view:

To place elements easily on a web page grid-view will be helpful. A responsive grid-view often has 12 columns with a total width of 100%. The property box-sizing: border-box makes sure that the padding and border are included in the total width and height of the elements and we have to include them in our CSS file.

```
* {  box-sizing: border-box; }
```

Example:

Style.css:

```
* {
        box-sizing: border-box;
   }

.child1{
        width:20%;
        float: left;
        background-color: red;
        padding: 10px;
}

.child2{
        width:70%;
        float: left;
        background-color: green;
        padding: 10px;
}
```

Output:

Media queries:

- Media queries are introduced in css3
- It uses the **@media** rule to include a block of CSS properties only if a certain condition is true
- We will use different breakpoints for different devices

- We can look at the same content of a webpage with a different response on different devices by using media queries

Example:

Here breakpoint is 768px. The below CSS styles applicable up to device width is 768px if device width exceeds 768px styles in media query is not worked.

```
@media only screen and (max-width: 768px) {
        body {
                background-color: lightblue;
        }
}
```

Breakpoints for different devices:

extra-small devices	max-width:600px
small devices	min-width:600px
medium devices	max-width:768px
large devices	min-width:992px;
extra-large devices	min-width:1200px