accenture

CASCADING STYLE SHEET

We can completely change the look of your website with only a few changes in CSS code.

Accenture Interactive

AGENDA

TOPIC	SPEAKER	TIME
Overview - SCSS Basics - Style Guide Creation	Xavier	10 min
Advanced Concepts	Swathi Kailasa	30 min
Demo	Swathi Kailasa	10 min
Q&A	Swathi Kailasa	10 min

Add your questions to the Chat Window – There will be a time for Q&A at the end of the session

CSS

Overview:

Importance of using SASS and creating style guide

Accenture Interactive

Advantages of SASS

Code Reusability

Create **Variables** and **Mixins**, and use it throughout your stylesheet, make your code more maintainable and reduce the need for repetitive code.

SCSS Nesting

With SCSS, you can nest selectors inside of other selectors, making it easier to understand the relationships between elements and reducing the need for repetitive class names.

Improved Organization

With the ability to use variables and nesting, SCSS allows you to organize your styles in a more modular and logical way.

Improved Readability

SCSS makes it easier to read and understand your code by allowing you to use more human-readable syntax and by making it easier to see the relationships between selectors and styles.

Compatible With CSS

Compatibility with CSS: SCSS is fully compatible with CSS, so you can use it alongside standard CSS styles in your projects.

Browser Support

Browser support: SCSS is widely supported by modern browsers, so you can use it without worrying about compatibility issues.

Style Guide Creation

- Typography
- Border
- Colors
- Forms
- Gutters
- Breakpoints
- Grid
- Mixins
- Button
- Hide/show

A CSS style guide is important because it helps to **maintain consistency** and organization in the design of a website or application. It sets clear guidelines for how elements such as typography, colours, and spacing should be used, which can make it easier for developers to implement design changes and for **multiple team members to work on the project without introducing inconsistencies**. A style guide can also make it easier to **update the design in the future**, as all of the design elements are clearly defined in one place. Additionally, a style guide can help to improve the performance and accessibility of the website or application by ensuring that the **code is clean, organized and semantic.**

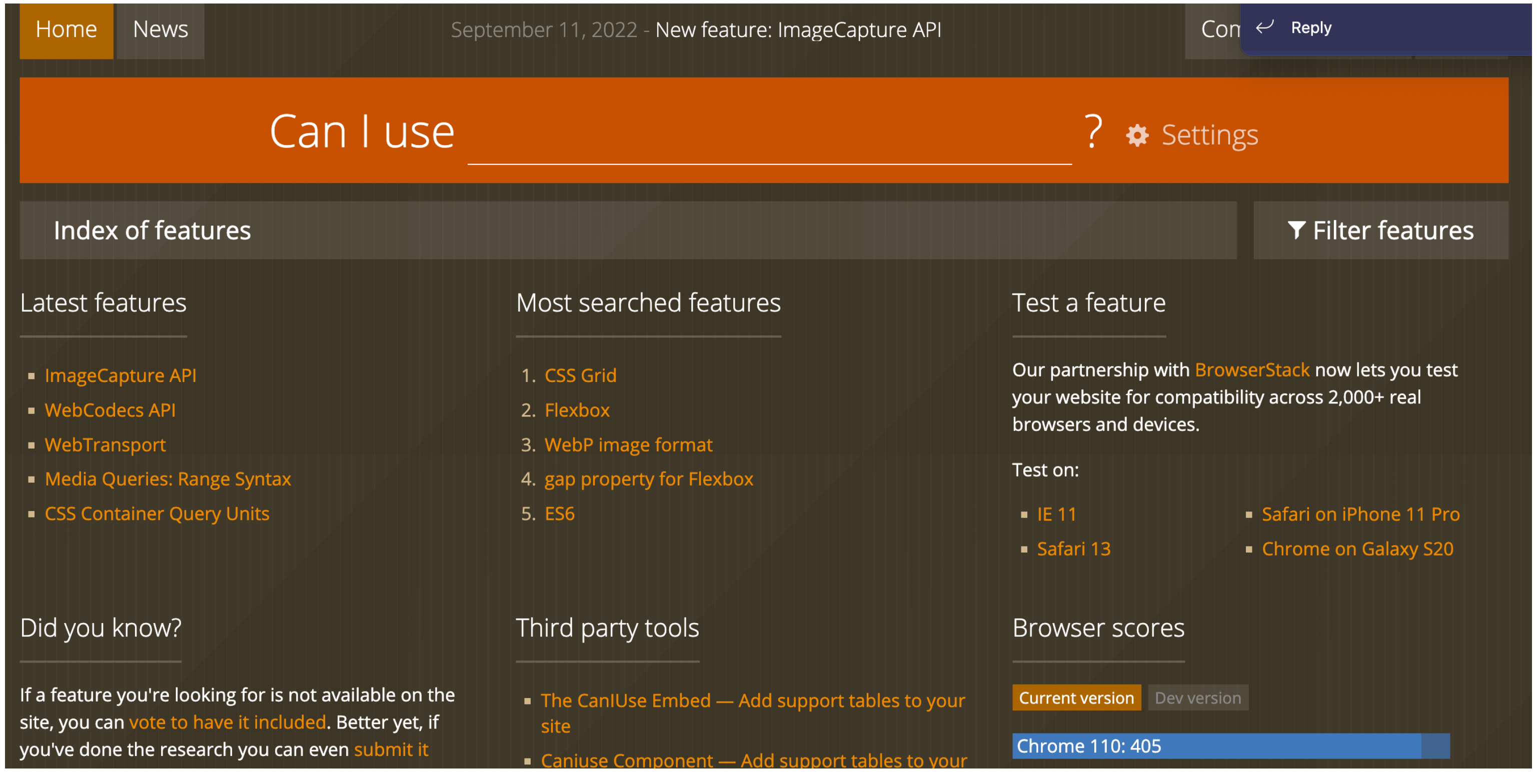
CSS / SCSS

Advanced concepts

Accenture Interactive

Check for Browser support

Before using any of the latest CSS Properties, make sure which all browser version does it support.
https://caniuse.com/



September 11, 2022 - New feature: ImageCapture API

Latest features

- ImageCapture API
- WebCodecs API
- WebTransport
- Media Queries: Range Syntax
- CSS Container Query Units

Most searched features

1. CSS Grid
2. Flexbox
3. WebP image format
4. gap property for Flexbox
5. ES6

Test a feature

Our partnership with BrowserStack now lets you test your website for compatibility across 2,000+ real browsers and devices.

Test on:

- IE 11
- Safari 13
- Safari on iPhone 11 Pro
- Chrome on Galaxy S20

Did you know?

If a feature you're looking for is not available on the site, you can vote to have it included. Better yet, if you've done the research you can even submit it

Third party tools

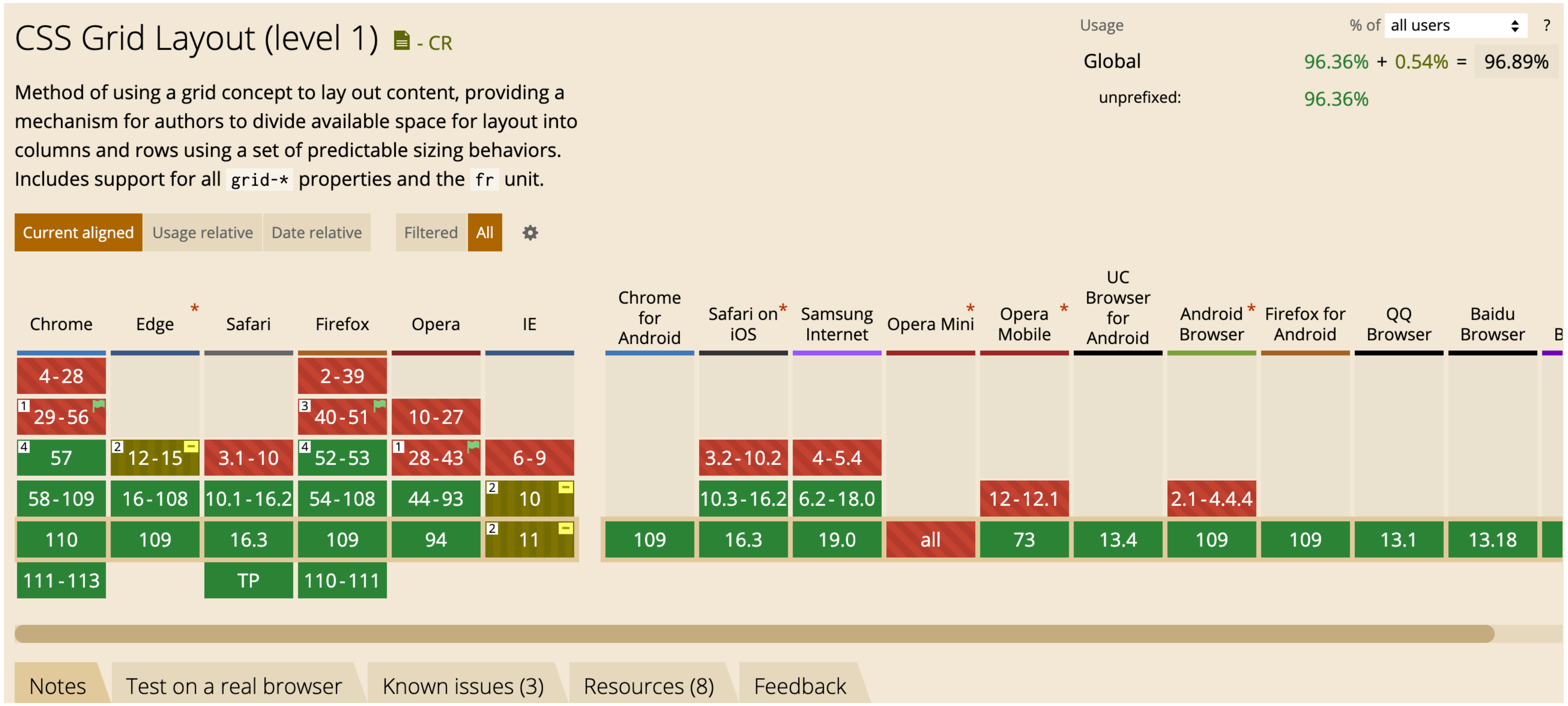
- The CanIUse Embed — Add support tables to your site
- Caniuse Component — Add support tables to your

Browser scores

Current version Dev version

Chrome 110: 405

Check for Browser support



CSS Grid Layout (level 1) - CR

Usage — % of all users ?

Global 96.36% + 0.54% = 96.89%

unprefixed: 96.36%

Method of using a grid concept to lay out content, providing a mechanism for authors to divide available space for layout into columns and rows using a set of predictable sizing behaviors. Includes support for all `grid-*` properties and the `fr` unit.

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	B
🟥 4-28			🟥 2-39													
🟥 [1] 29-56			🟥 [3] 40-51	🟥 10-27												
🟩 [4] 57	▨ [2] 12-15	🟥 3.1-10	🟩 [4] 52-53	🟥 [1] 28-43	🟥 6-9		🟥 3.2-10.2	🟥 4-5.4								
🟩 58-109	🟩 16-108	🟩 10.1-16.2	🟩 54-108	🟩 44-93	▨ [2] 10		🟩 10.3-16.2	🟩 6.2-18.0		🟥 12-12.1		🟥 2.1-4.4.4				
🟩 110	🟩 109	🟩 16.3	🟩 109	🟩 94	▨ [2] 11	🟩 109	🟩 16.3	🟩 19.0	🟥 all	🟩 73	🟩 13.4	🟩 109	🟩 109	🟩 13.1	🟩 13.18	🟩
🟩 111-113		🟩 TP	🟩 110-111													

SCSS Variables

Accenture Interactive

SCSS Creating Variables

Organization and group related variables together and use comments to separate different sections.

Good Example:

```scss
$color-red: #661100;

//Text
$color-accent: $color-red;

//Container Backgrounds
$color-bg: $color-red;

//Container Borders
$border-color: $color-red;

//Button Borders
$btn-border-color: $color-red;

//Buttons color
$color-btn-primary: $color-red;

//Buttons background
$color-bg-btn: $color-red;
```

Bad Example:

```scss
$color-red: #661100;

p {
    color: $color-red;
}

div {
    background-color: $color-red;
    border-color: $color-red;
}

Button {
    color: $color-red;
    background-color: $color-red;
    border-color: $color-red;
}
```

CSS Custom Properties

Accenture Interactive

CSS Custom Properties

CSS custom properties, are used in CSS stylesheets and are defined using the **--** syntax. They can be updated dynamically using JavaScript and offer the ability to change styles based on user interaction and other dynamic factors.

CSS Example:

```
:root { --blue: #1e90ff;} /* global variable */

body { background-color: var(--blue); }

button {
  --blue: #0000ff;  /* local variable will override global */
  background-color: var(--blue);
}
```

JS Example:

```
//Set the variable value
document.documentElement.style.setProperty('--variable-name', 'value’);
//Get the computed style
var computedStyle = window.getComputedStyle(document.documentElement);
//Get the variable value
var variableValue = computedStyle.getPropertyValue('--variable-name');
```

CSS

Block Element Modifier

Accenture Interactive

What is BEM?

BEM is a front-end naming method for organizing and naming CSS classes. The Block, Element, Modifier methodology is a popular naming convention for class names in HTML and CSS. It helps to write clean CSS by following some simple rules.

How to implement BEM?

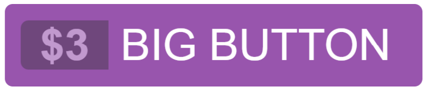
There are three main parts of BEM.

- **Block**: which holds everything (elements) inside and acts as a scope.
  Eg: .btn {/* Styles */}
- **Element**: which acts as a specific part of the component.
  Eg: .btn__price{/* Styles */}
- **Modifier**: which adds additional styles to a specific element(s).
  Eg: .btn--big {/* Styles */}
  .btn--orange {/* Styles */}

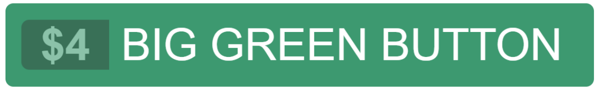
If all three are used in a name it would look something like this:
[block]__[element]--[modifier]

Below are the examples to use BEM structure

```
<a href=“# class="btn btn--purple btn--big">
  <span class="btn__price">$3</span>
  <span class="btn__text">Big button</span>
</a>
```



```
<a href=“# class="btn btn--green btn--big">
  <span class="btn__price">$4</span>
  <span class="btn__text">Big green button</span>
</a>
```



```
/* Block Component */
.btn{ }

/* Element that depends upon the block*/
.btn__price{ }

/* Modifier that changes the style of the block*/
.btn--green{ }
```

Class Naming Convention:

- Don’t use uppercase
- Aways use -(hyphen) as a separator
- Use __(double underscore) as a separator only for elements that depends upon a block
- Use --(double hyphen) as separaton only for modifier

Nesting

Accenture Interactive

SCSS Nesting

In the good example, the nesting is clear and concise, making it easy to understand the styles being applied. In the bad example, the excessive nesting makes it difficult to understand the styles being applied and can lead to performance issues.

Good Example:

```
.parent { /*Block*/

  &__element {/*Element*/
    font-size: 16px;
    color: #333;
  }

  &--modifier {/*Modifier*/
     background-color: #ccc;
  }

  &:hover { /* pseudo-elements  and pseudo-classes */
    color: #000;
   }
}
```

Bad Example:

```
.parent {

  .child {
    font-size: 16px;
    color: #333;

    .grand-child {
      font-size: 15px;

      .great-grand-child {
        font-size: 14px;
      }
    }
  }
}
```

SCSS Nesting

When a SCSS file is compiled, the nested rules are expanded into regular CSS rules as below.

Good Example:

```
.parent__element {
   font-size: 16px;
   color: #333;
 }

 .parent--modifier {
    background-color: #ccc;
  }

 .parent::hover {
    color: #000;
 }
```

Bad Example:

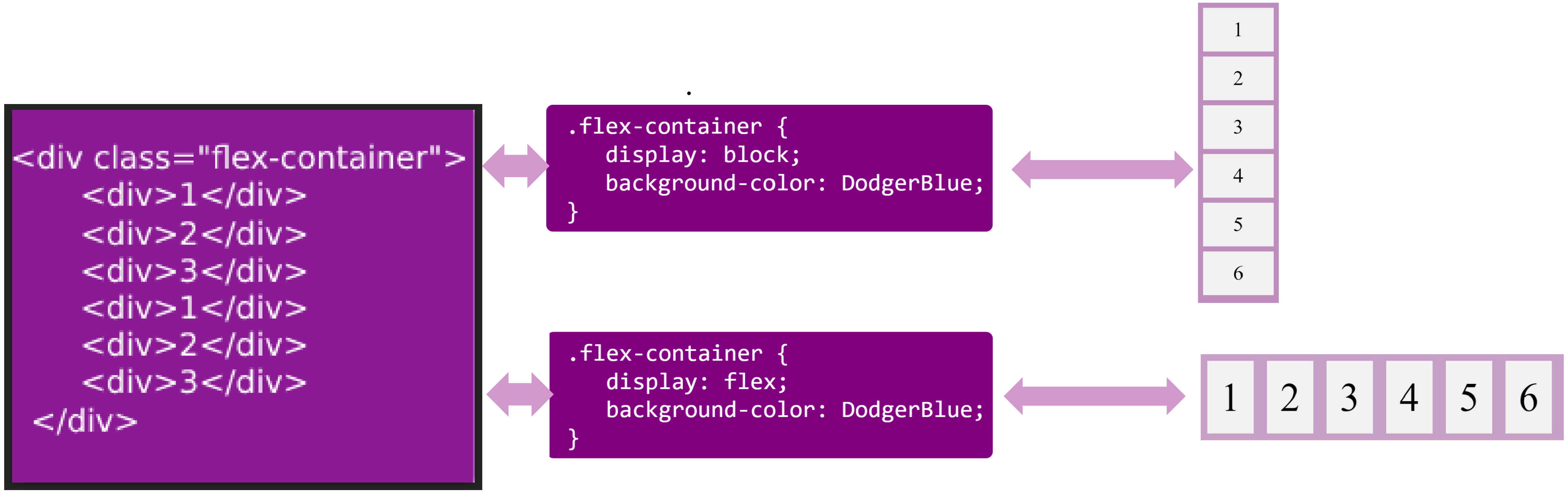
```
.parent .child {
   font-size: 16px;
   color: #333;
}
.parent .child .grand-child {
     font-size: 15px;
}
.parent .child .grand-child .great-grand-child {
       font-size: 14px;
}
```

Flexbox

Accenture Interactive

CSS Flexbox

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.



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

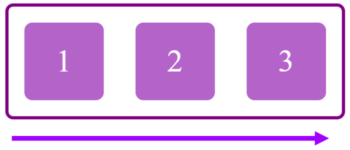
```
.flex-container {
   display: block;
   background-color: DodgerBlue;
}
```

```
.flex-container {
   display: flex;
   background-color: DodgerBlue;
}
```

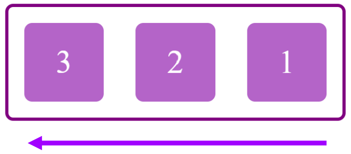
CSS FLEX-DIRECTION

The flex-direction property defines in which direction the container wants to stack the flex items.

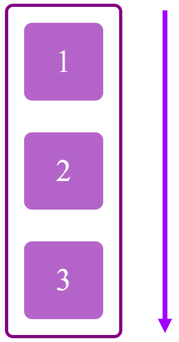
```
.flex-container {
  display: flex;
  flex-direction: row;
}
```



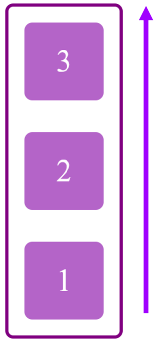
```
.flex-container {
  display: flex;
  flex-direction: row-reverse;
}
```



```
.flex-container {
  display: flex;
  flex-direction: column;
}
```



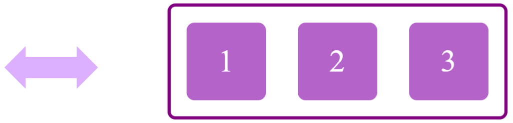
```
.flex-container {
  display: flex;
  flex-direction: column-reverse;
}
```



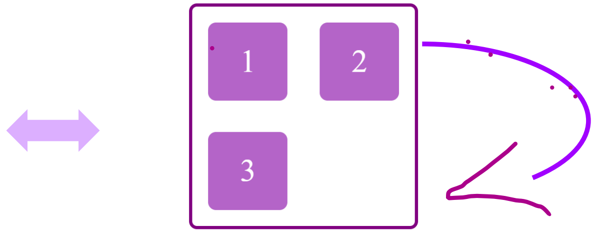
CSS FLEX-WRAP

The flex-wrap property specifies whether the flex items should wrap or not.

```
.flex-container {
  display: flex;
  flex-wrap: no-wrap;
}
```

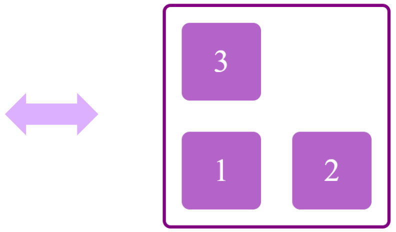


```
.flex-container {
  display: flex;
  flex-wrap: wrap;
}
```



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

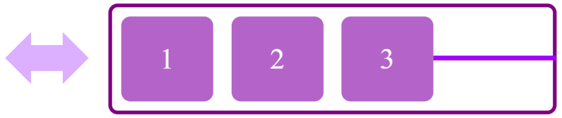
```
.flex-container {
  display: flex;
  flex-wrap: wrap-reverse;
}
```



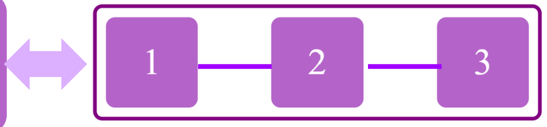
CSS JUSTIFY-CONTENT

The justify-content property is used to align the flex items horizontally:

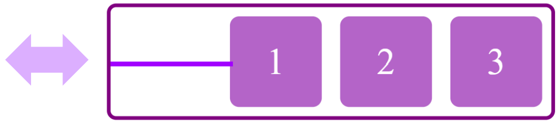
```
.flex-container {
  display: flex;
  justify-content: flex-start;
}
```



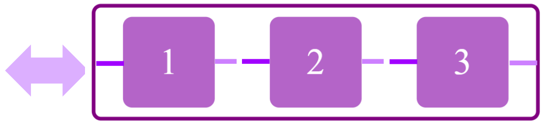
```
.flex-container {
  display: flex;
  justify-content: space-between;
}
```



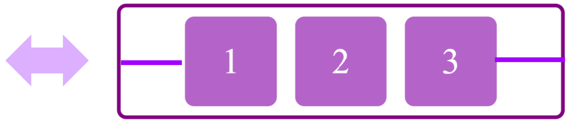
```
.flex-container {
  display: flex;
  justify-content: flex-end;
}
```



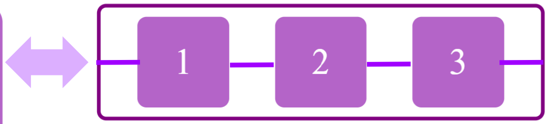
```
.flex-container {
  display: flex;
  justify-content: space-around;
}
```



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



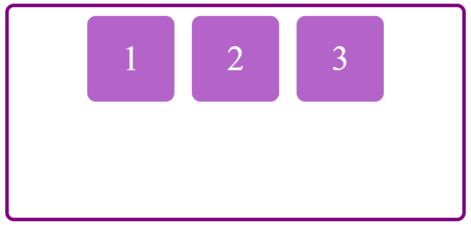
```
.flex-container {
  display: flex;
  justify-content: space-evenly;
}
```



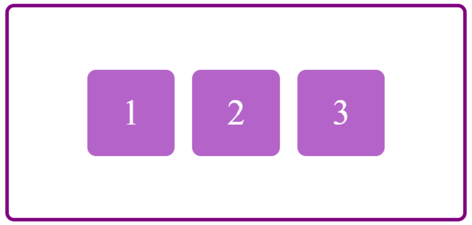
CSS ALIGN-ITEMS

The align-items property is used to align the flex items vertically.

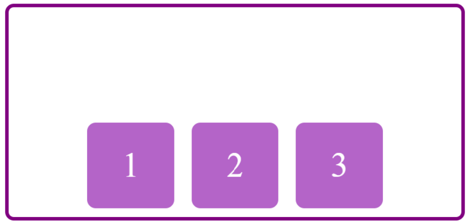
```
.flex-container {
  display: flex;
  align-items: flex-start;
}
```



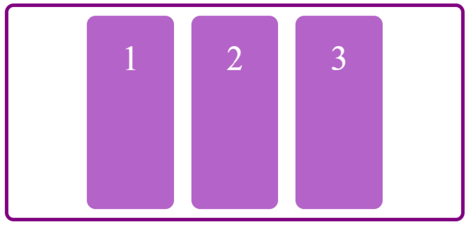
```
.flex-container {
  display: flex;
  align-items: center;
}
```



```
.flex-container {
  display: flex;
  align-items: flex-end;
}
```



```
.flex-container {
  display: flex;
  align-items: stretch;
}
```



CSS
Grid Column Structure

Accenture Interactive

CSS GRID LAYOUT

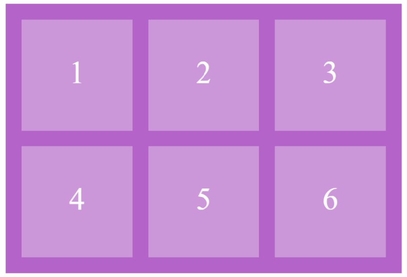
The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

Grid Elements :

A grid layout consists of a parent element, with one or more child elements.

```
<div class=“parent">
  <div class=“child1">1</div>
  <div class=“child2">2</div>
  <div class="child3">3</div>
  <div class="child4">4</div>
  <div class="child5">5</div>
  <div class="child6">6</div>
</div>
```

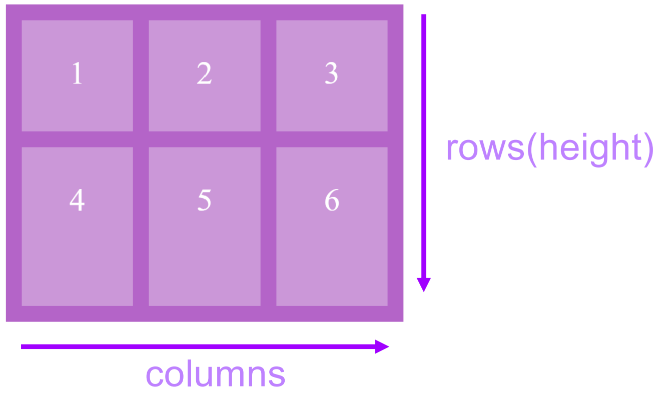
```
.parent{
    display: grid;
    grid: 70px 70px / auto auto auto;
    grid-gap: 10px;
}
```



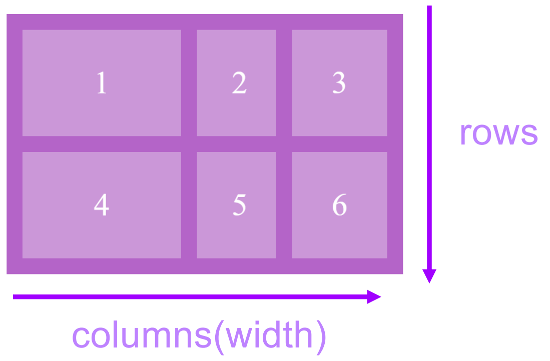
CSS GRID TEMPLATE ROWS/COLUMNS

The grid-template-columns property specifies the number (and the widths) of columns in a grid layout.
The grid-template-rows property specifies the number (and the heights) of the rows in a grid layout.

```
.parent{
    display: grid;
    grid-template-columns: auto auto auto;
    grid-template-rows: 70px 100px;
    //OR
    grid-template: 70px 100px / auto auto auto;
}
```



```
.parent{
    display: grid;
    grid-template-columns: 100px 50px 60px;
    grid-template-rows: auto auto;
    //OR
    grid-template: auto / 100px 50px 60px;
}
```

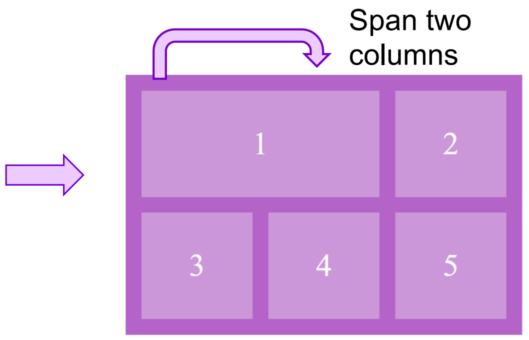


CSS GRID COLUMN

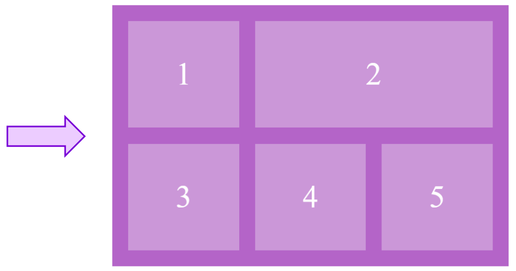
grid-column property to specify on which column to place an item, and how many columns the item will span

```
.parent {
    display: grid;
    grid-template: auto / auto auto auto ;
}
```

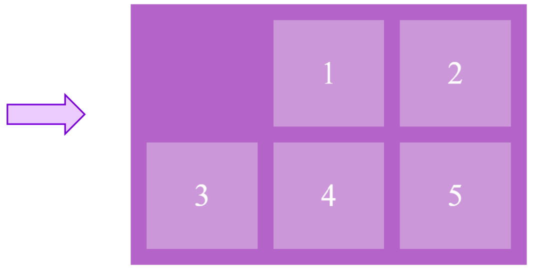
```
.child1{
    grid-column: 1 / span 2;
}
```



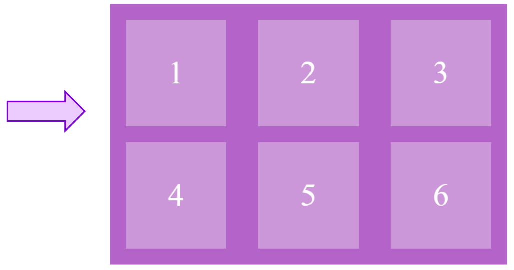
```
.child2 {
 grid-column-end: span 2;
}
```



```
.child1{
  grid-column-start: 2;
}
```



```
.parent {
 grid-column-gap: 20px;
}
```

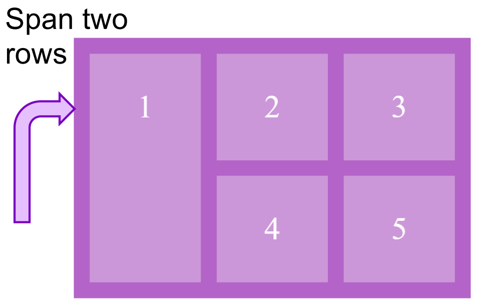


CSS GRID ROW

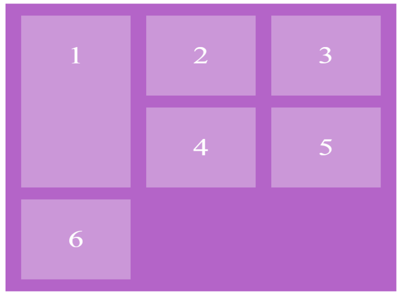
grid-row property to specify on which row to place an item, and how many row the item will span.

```
.parent {
    display: grid;
    grid-template: auto / auto auto auto ;
}
```

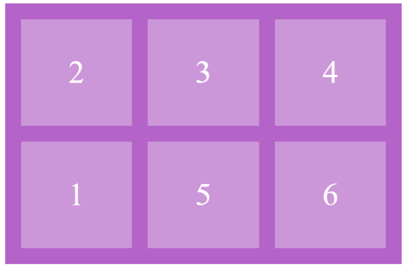
```
.child1 {
 grid-row: 1 / span 2;
}
```



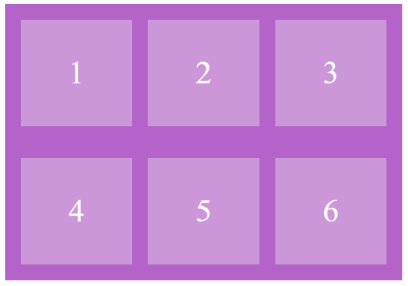
```
.child1 {
    grid-row-end: span 2;
}
```



```
.child1 {
    grid-row-start: 2;
}
```



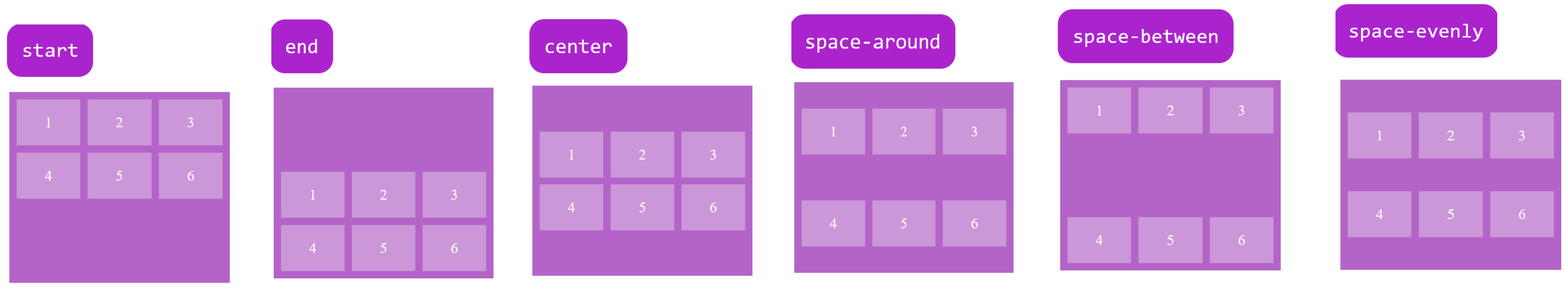
```
.parent {
    grid-row-gap: 20px;
}
```



CSS GRID ALIGN CONTENT, JUSTIFY CONTENT

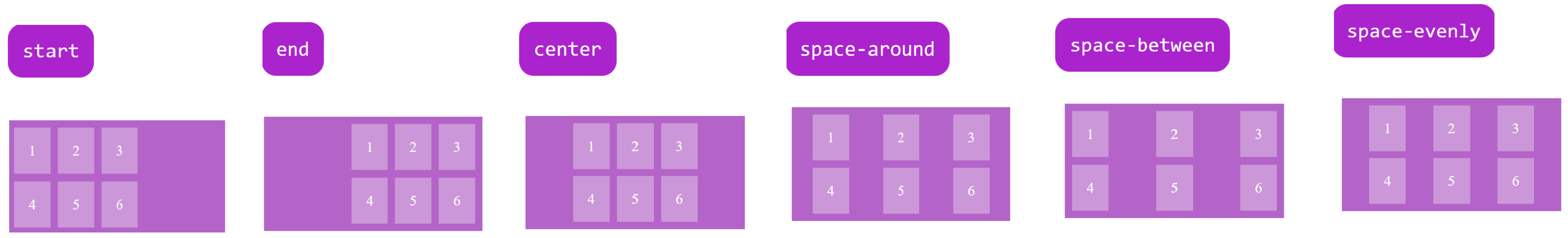
```
.parent {align-content: start | end | center | space-around | space-between | space-evenly}
```

align-content



```
.parent {justify-content: start | end | center | space-around | space-between | space-evenly}
```

justify-content



CSS

Responsive Styles

Accenture Interactive

What is Responsive styles?

Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones):

By using Media queries in CSS we can make our website as more responsive in all the resolutions

Media Query is a popular technique that enables to deliver a style sheet to different devices which have different screen sizes and resolutions respectively

Syntax:

```
@media( media feature ) {
   // CSS Property
}
```

The screen resolutions of different devices are listed below:

```
 /* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}

/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops, 1200px and up) */
@media only screen and (min-width: 1200px) {...}
```

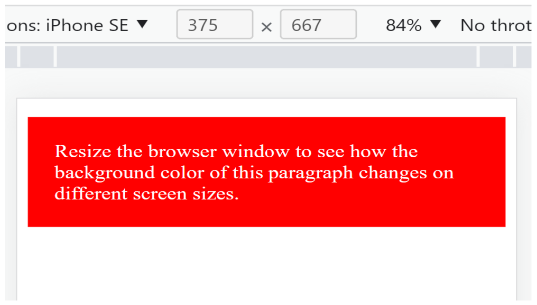
HTML :

```
<p class="example">Resize the browser window to see how the background color of this
paragraph changes on different screen sizes.</p>
```

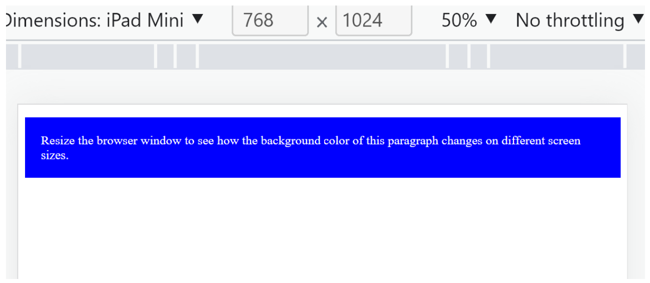
CSS :

```
@media only screen and (max-width: 600px) {
.example {background: red;}
}

@media only screen and (min-width: 600px) {
.example {background: green;}
}

@media only screen and (min-width: 768px) {
.example {background: blue;}
}

@media only screen and (min-width: 992px) {
.example {background: orange;}
}

@media only screen and (min-width: 1200px) {
.example {background: pink;}
}
```

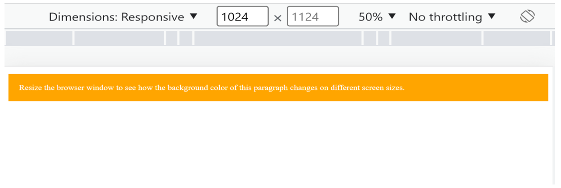
Output for different resolutions :


ons: iPhone SE ▼ 375 × 667 84% ▼ No throt

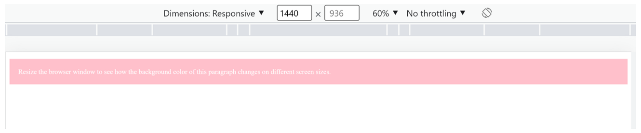
Resize the browser window to see how the background color of this paragraph changes on different screen sizes.


Dimensions: iPad Mini ▼ 768 × 1024 50% ▼ No throttling ▼

Resize the browser window to see how the background color of this paragraph changes on different screen sizes.


Dimensions: Responsive ▼ 1024 × 1124 50% ▼ No throttling ▼

Resize the browser window to see how the background color of this paragraph changes on different screen sizes.


Dimensions: Responsive ▼ 1440 × 936 60% ▼ No throttling ▼

Resize the browser window to see how the background color of this paragraph changes on different screen sizes.

SCSS
Mixins

Accenture Interactive

What is a mixin?

A mixin in CSS preprocessor languages (such as Sass, Less, or Stylus) is a reusable block of styles that can be included in multiple places within a stylesheet. It is similar to a function in programming, as it can accept arguments and produce a resulting set of styles.

SCSS :

```scss
@mixin box-shadow($shadow) {
   -webkit-box-shadow: $shadow;
   -moz-box-shadow: $shadow;
   box-shadow: $shadow;
}

.element {
   @include box-shadow(0 2px 4px #ccc);
}
```

CSS (Output after compilation):

```css
.element {
   -webkit-box-shadow: 0 2px 4px #ccc;
   -moz-box-shadow: 0 2px 4px #ccc;
   box-shadow: 0 2px 4px #ccc;
}
```

CSS

Position

Accenture Interactive

The Position Property

The position property specifies the type of positioning method used for an element.

```
position: static | relative | absolute | fixed | sticky
```

1. static: This is the default position value and means that an element is positioned according to the normal flow of the document. The position of a static element is not affected by top, bottom, left, or right values.
2. relative: A relative positioned element is positioned relative to its normal position in the document flow. An element can be moved away from its normal position using the top, bottom, left, and right properties.
3. absolute: An absolute positioned element is positioned relative to the nearest positioned ancestor element. If no positioned ancestor exists, it is positioned relative to the initial containing block (usually the body element). The position of an absolute element is set using the top, bottom, left, and right properties.
4. fixed: A fixed positioned element is positioned relative to the browser window and does not move when the user scrolls the page. The position of a fixed element is set using the top, bottom, left, and right properties.
5. sticky; is positioned based on the user's scroll position. A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position: fixed).

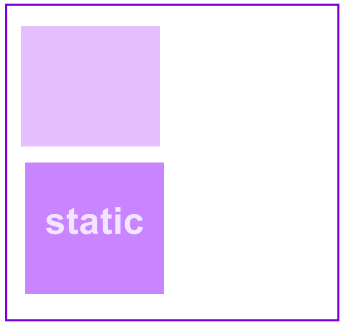
The Position Property

The position property specifies the type of positioning method used for an element.

position: static

HTML elements are positioned static by default. Static positioned elements are not affected by the top, bottom, left, and right properties.

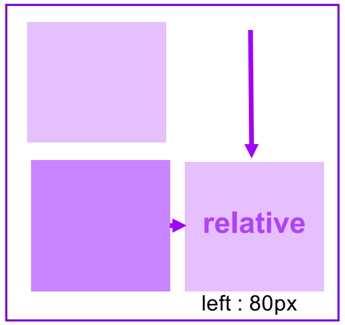
```
div.static{
  position: static;
  left: 80px;
}
```



position:relative

The element is positioned according to the normal flow of document, and then offset relative to itself based on the values of top, bottom, right and left properties.

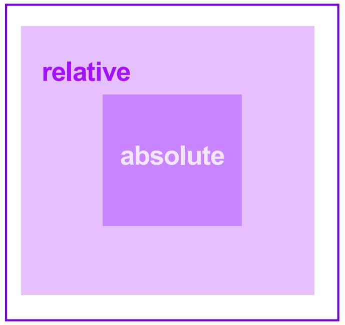
```
div.relative {
  position: relative;
  left: 80px;
}
```



position:absolute

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

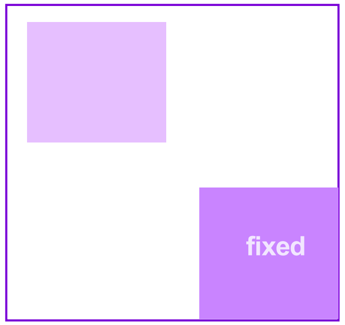
```
div.relative {
  position: relative;
}

div.absolute {
  position: absolute;
  top: 60px;
  left: 60px;
}
```



position: fixed

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element. A fixed element does not leave a gap in the page where it would normally have been located.

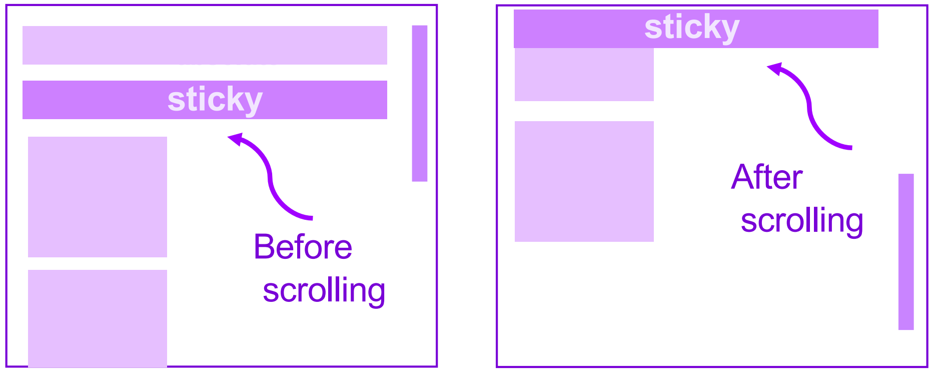
```
div.fixed {
  position: fixed;
  bootom: 0;
  right: 0;
}
```



position: sticky

static

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.



accenture

Demo

Accenture Interactive

accenture

Q&A

Accenture Interactive