Cascading Style Sheet (CSS)
Part 04

CSS Rounded Corners

```
#rcorners1 {
    border-radius: 25px;
    background: #73AD21;
    padding: 20px;
    width: 150px;
    height: 100px;
}

<p id="rcorners1">Rounded corners!</p>
```

CSS Rounded Corners

border-radius: 15px 50px 30px 5px; (TL, TR, BR, BL)

border-radius: 15px 50px 30px; (TL, TR-BL, BR)

border-radius: 15px 50px; (TL-BR, TR-BL)

border-radius: 15px; (All the corners)

CSS Rounded Corners

```
border-radius: 25px;
```

≈

```
border-top-left-radius:     25px;
border-top-right-radius:    25px;
border-bottom-right-radius: 25px;
border-bottom-left-radius:  25px;
```

CSS Border Image (shorthand property)

- To set an image to be used as border around an element instead of normal border

The property has three parts:

1) The image to use as the border
2) Where to slice the image
3) Define whether the middle sections should be repeated or stretched

CSS Border Image (Ex.)

```
#borderimg {
  border: 10px solid transparent;
  padding: 15px;
  width: 300px;
  border-image: url(border.png) 30 round;
}
```

```
<p id="borderimg">border-image: url(border.png) 30 round;</p>
```

Note: slicing is done 30% from all the four sides of a border image

CSS Border Image (Output)

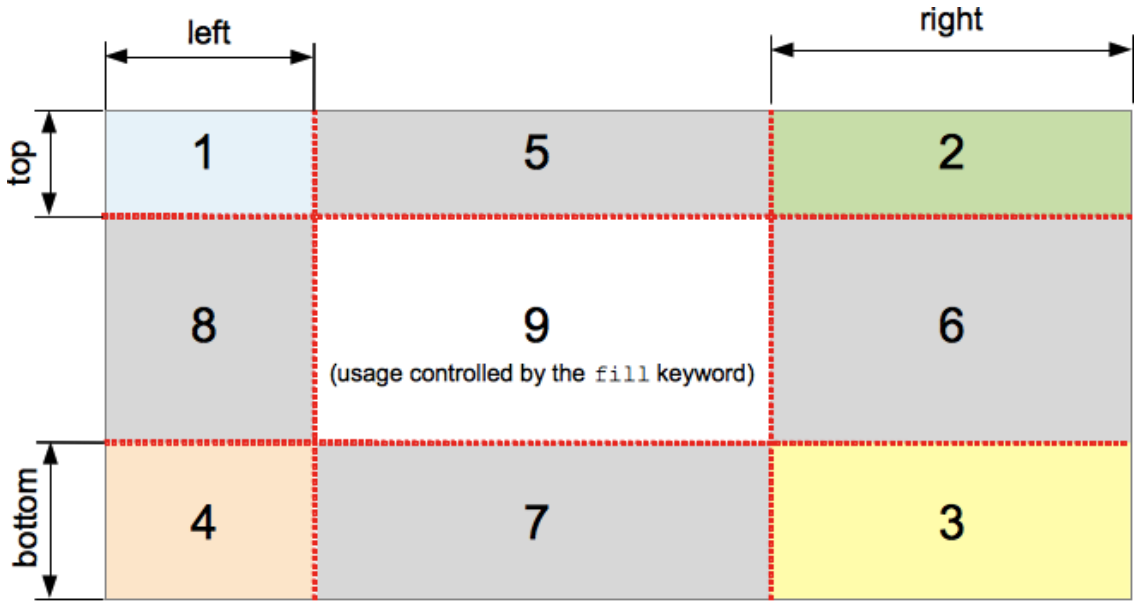
The border-image Property

Here, the middle sections of the image are repeated to create the border:

```
border-image: url(border.png) 30 round;
```

Here is the original image:

CSS Border Image



- Zones 1-4 are corner regions. Each one is used a single time to form the corners of the final border image
- Zones 5-8 are edge regions. These are repeated, scaled, or otherwise modified in the final border image to match the dimensions of the element
- Zone 9 is the middle region. It is discarded by default, but is used like a background image if the keyword fill is set

CSS Border Image

- border-image-source
- border-image-slice
- border-image-width
- border-image-outset and
- border-image-repeat

CSS Transitions

- Allows you to change property values smoothly, over a given duration

CSS Transitions

- transition (shorthand property)
- transition-property
- transition-duration
- transition-timing-function
- transition-delay

```
transition: <property> <duration> <timing-function> <delay>;
```

CSS Transitions

- **transition-property**
  - Specifies the name or names of the CSS properties to which transitions should be applied.
- **transition-duration**
  - Specifies the duration over which transitions should occur. You can specify a single duration that applies to all properties during the transition, or multiple values to allow each property to transition over a different period of time.

CSS Transitions

- **transition-timing-function**
  - Specifies a function to define how intermediate values for properties are computed. Most timing functions can be specified by providing the graph of the corresponding function, as defined by four points defining a cubic bezier.
- **transition-delay**
  - Defines how long to wait between the time a property is changed and the transition actually begins.

CSS Transition (Ex.)

```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}
div:hover {
  width: 300px;
}
```

The transition effect will start when the specified CSS property (width) changes value.

CSS Transition (Output)

The transition Property

Hover over the div element below, to see the transition effect:

Change multiple properties (Ex.2)

```
div {
  width: 50px;
  height: 50px;
  background: green;
  transition: width 2s, height 4s;
}

div:hover {
  width: 200px;
  height: 200px;
}
```

CSS Transition (Output)

The transition Property

Hover over the div element below, to see the transition effect:

CSS Animations

- An animation lets an element gradually change from one style to another.

- You can change as many CSS properties you want, as many times you want.

- To use CSS animation, you must first specify some **keyframes** for the animation.

- Keyframes hold what styles the element will have at certain times.

The @keyframes Rule

- when we specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

- you must bind the animation to an element.

- animation-duration property defines how long time an animation should take to complete

- the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

CSS Animations (Ex.1)

```
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}

@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
```

CSS Animations (Output 1)

Note: When an animation is finished, it changes back to its original style.

CSS Animations (Ex.2)

```
@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}
```

CSS Animations (Output 2)

CSS Animations (Ex.3)

```
div {
  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 4s;
}

@keyframes example {
  0%   {background-color:red; left:0px; top:0px;}
  25%  {background-color:yellow; left:200px; top:0px;}
  50%  {background-color:blue; left:200px; top:200px;}
  75%  {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
```

CSS Animations (Output 3)

CSS Pagination

```
.pagination {
    display: inline-block;
}
.pagination a {
    color: black;
    float: left;
    padding: 8px 16px;
    text-decoration: none;
}

<h2>Simple Pagination</h2>

<div class="pagination">
  <a href="#">&laquo;</a>
  <a href="#">1</a>
  <a href="#">2</a>
  <a href="#">3</a>
  <a href="#">4</a>
  <a href="#">5</a>
  <a href="#">6</a>
  <a href="#">&raquo;</a>
</div>
```

Simple Pagination

« 1 2 3 4 5 6 »

CSS Pagination

```
.pagination {
    display: inline-block;
}
.pagination a {
    color: black;
    float: left;
    padding: 8px 16px;
    text-decoration: none;
}
.pagination a.active {
    background-color: #4CAF50;
    color: white;
}
.pagination a:hover:not(.active) {background-color: #ddd;}
```

```
<h2>Active and Hoverable Pagination</h2>
<p>Move the mouse over the numbers.</p>

<div class="pagination">
  <a href="#">&laquo;</a>
  <a href="#">1</a>
  <a class="active" href="#">2</a>
  <a href="#">3</a>
  <a href="#">4</a>
  <a href="#">5</a>
  <a href="#">6</a>
  <a href="#">&raquo;</a>
</div>
```

Active and Hoverable Pagination

Move the mouse over the numbers.

« 1 2 3 4 5 6 »

CSS Multi-column Layout

Property	Description
column-count	Specifies the number of columns an element should be divided into
column-fill	Specifies how to fill columns
column-gap	Specifies the gap between the columns
column-rule	A shorthand property for setting all the column-rule-* properties
column-rule-color	Specifies the color of the rule between columns
column-rule-style	Specifies the style of the rule between columns
column-rule-width	Specifies the width of the rule between columns
column-span	Specifies how many columns an element should span across
column-width	Specifies a suggested, optimal width for the columns
columns	A shorthand property for setting column-width and column-count

column-count property

```
.newspaper {
    -webkit-column-count: 3; /* Chrome, Safari, Opera */
    -moz-column-count: 3; /* Firefox */
    column-count: 3;
}
```

```
<div class="newspaper">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad
minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut
aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in
vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla
facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

CSS Variables

The var() function can be used to insert the value of a custom property

- Variables should be declared within a CSS selector that defines its scope

- For a global scope you can use either the :root or the body selector.

- The variable name must begin with two dashes (--) and is case sensitive!

The syntax of the var() function is as follows:

var(*custom-name, value*)

CSS Variables

```
:root {
    --main-bg-color: coral;
}
#div1 {
    background-color: var(--main-bg-color);
    padding: 5px;
    width: 150px;
}
```

```
<h1>The var() Function</h1>
<div id="div1">some text goes here</div>
```

The var() Function

some text goes here

CSS Box Sizing

- The CSS **box-sizing** property allows us to <u>include the padding and border in an element's total width and height</u>

- By default, the width and height of an element are:

width + padding + border = actual width of an element

height + padding + border = actual height of an element

- By default box-sizing value is **content-box**

CSS Box Sizing (content-box)

```
.div1 {
    width: 200px;
    height: 50px;
    border: 1px solid blue;
}

.div2 {
    width: 200px;
    height: 50px;
    padding: 10px;
    border: 1px solid red;
}
<div class="div1">First div</div><br>
<div class="div2">Second div</div>
```

First div

Second div

CSS Box Sizing (border-box)

```
.div1 {
    width: 300px;
    height: 100px;
    border: 1px solid blue;
    box-sizing: border-box;
}

.div2 {
    width: 300px;
    height: 100px;
    padding: 50px;
    border: 1px solid red;
    box-sizing: border-box;
}
<div class="div1">Both divs are the same size now!</div><br>
<div class="div2">Hooray!</div>
```

Both divs are the same size now!

Hooray!

CSS Flexbox

The Flexible Box Layout Module, makes it easier to design

- flexible responsive layout structure without using float or positioning
- designed as a one-dimensional layout model

Parent Element (Container)

The flex container becomes flexible by setting the display property to *flex*:

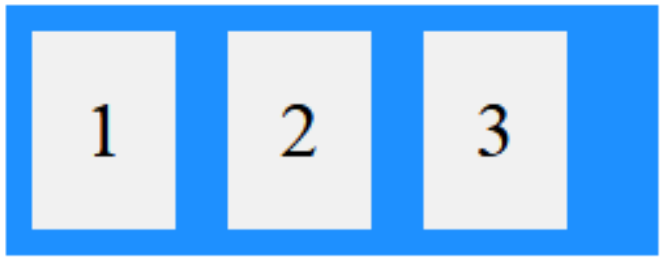
```
.flex-container {
  display: flex;
}
```

CSS Flexbox

```
.flex-container {
  display: flex;
  background-color: DodgerBlue;
  width: 250px;
}

.flex-container > div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}

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



CSS Flexbox

The flex container properties are:

flex-direction:
row | row-reverse | column | column-reverse | initial | inherit;

flex-wrap:
nowrap | wrap | wrap-reverse | initial | inherit;

flex-flow (shorthand property):
flex-direction flex-wrap | initial | inherit;

CSS Flexbox

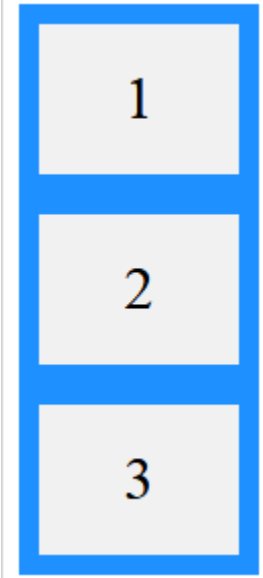
justify-content:
flex-start | flex-end | center | space-between | space-around | initial | inherit;

align-items:
stretch | center | flex-start | flex-end | baseline | initial | inherit;

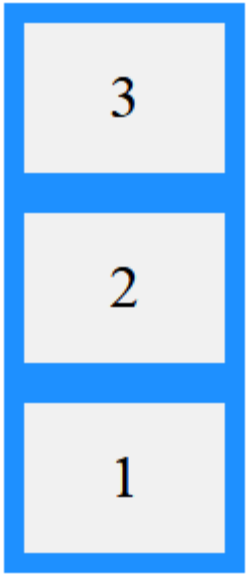
align-content:
stretch | center | flex-start | flex-end | space-between | space-around | initial | inherit;

Flex-direction property

```
.flex-container {
  display: flex;
  flex-direction: column;
  background-color: DodgerBlue;
  width:120px;
}
```



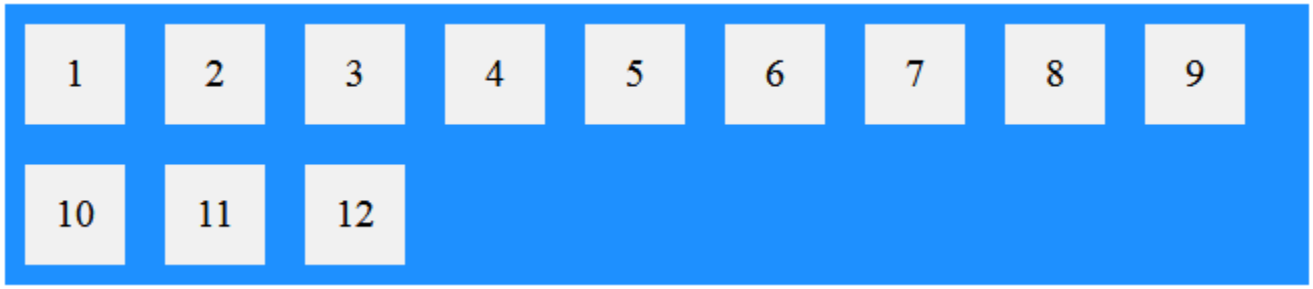

```
.flex-container {
  display: flex;
  flex-direction: column-reverse;
  background-color: DodgerBlue;
  width: 120px;
}
```

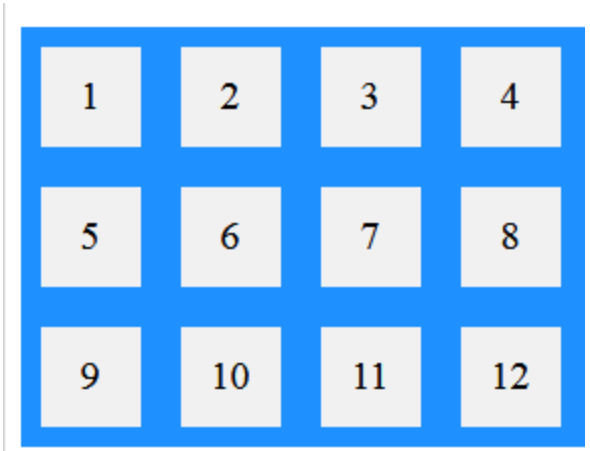


row, row-reverse

The flex-wrap Property

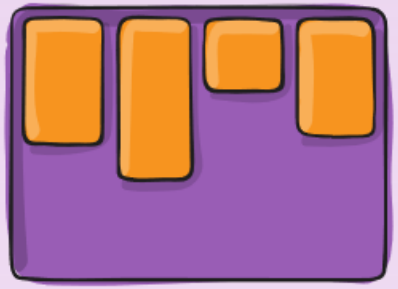
```
.flex-container {
  display: flex;
  flex-wrap: wrap;
  background-color: DodgerBlue;
}
```



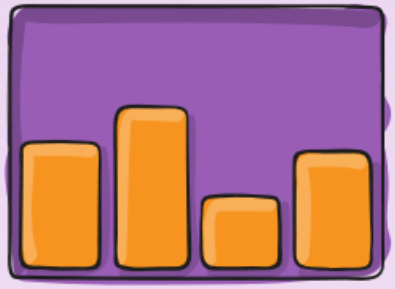


align-items

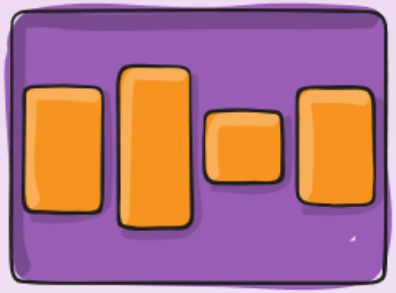
flex-start



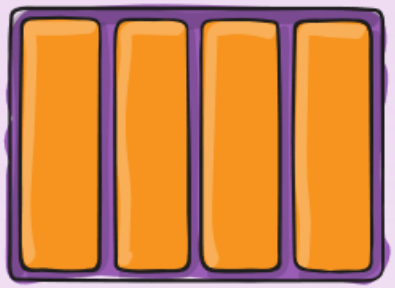
flex-end



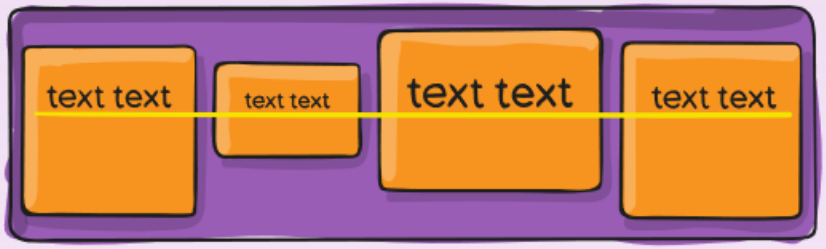
center



stretch

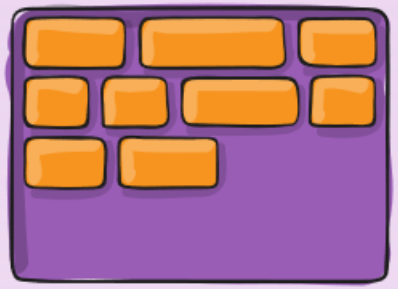


baseline



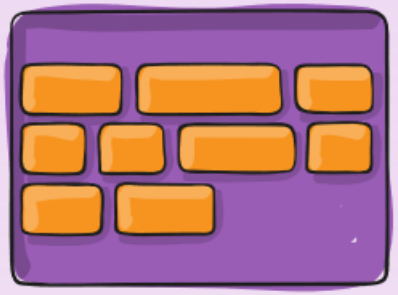
align-content

flex-start

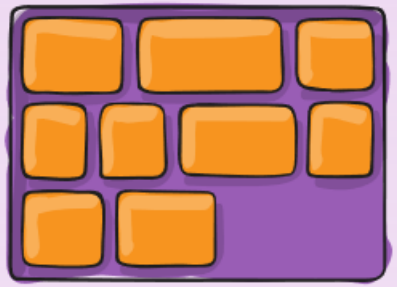


flex-end

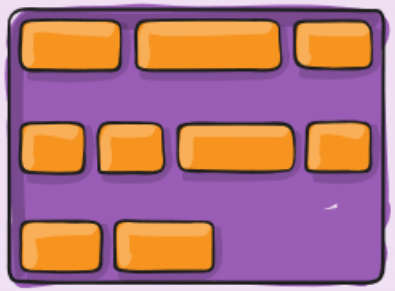
center



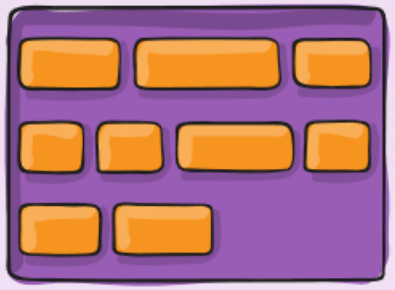
stretch



space-between



space-around



CSS3 Media Queries

Media queries can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (phone in landscape or portrait mode?)
- resolution (1920 X 1080)

CSS2 Introduced Media Types

Value	Description
all	Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screenreaders that "reads" the page out loud

CSS3 Media Query (ex.)

```
body {
    background-color: pink;
}

@media screen and (min-width: 480px) {
    body {
        background-color: lightgreen;
    }
}

<h1>Resize the browser</h1>
```

Resize the browser

Resize the browser

References

- https://www.w3schools.com/css/
- https://developer.mozilla.org/
- https://css-tricks.com/snippets/css/a-guide-to-flexbox/