

UNIT-3: Cascading Style sheet

CSS3 Modules

CSS3 has been split into "modules". It contains the "old CSS specification" (which has been split into smaller pieces). In addition, new modules are added.

Some of the most important CSS3 modules are:

- Selectors
- Box Model
- Backgrounds and Borders
- Image Values and Replaced Content
- Text Effects
- 2D/3D Transformations
- Animations
- Multiple Column Layout
- User Interface

Most of the new CSS3 properties are implemented in modern browsers.

CSS3 Rounded Corners

With the CSS3 **border-radius** property, you can give any element "rounded corners".

CSS3 border-radius - Specify Each Corner

If you specify only one value for the **border-radius** property, this radius will be applied to all 4 corners.

However, you can specify each corner separately if you wish. Here are the rules:

- **Four values:** first value applies to top-left, second value applies to top-right, third value applies to bottom-right, and fourth value applies to bottom-left corner
- **Three values:** first value applies to top-left, second value applies to top-right and bottom-left, and third value applies to bottom-right
- **Two values:** first value applies to top-left and bottom-right corner, and the second value applies to top-right and bottom-left corner
- **One value:** all four corners are rounded equally

Here are three examples:

1. Four values - border-radius: 15px 50px 30px 5px:

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

2. Three values - border-radius: 15px 50px 30px:

3. Two values - border-radius: 15px 50px:

CSS3 Rounded Corners Properties

Property	Description
border-radius	A shorthand property for setting all the four border-*-*-radius properties
border-top-left-radius	Defines the shape of the border of the top-left corner
border-top-right-radius	Defines the shape of the border of the top-right corner
border-bottom-right-radius	Defines the shape of the border of the bottom-right corner
border-bottom-left-radius	Defines the shape of the border of the bottom-left corner

CSS3 Border Images

With the CSS3 **border-image** property, you can set an image to be used as the border around an element.

CSS3 border-image Property

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

The CSS3 **border-image** property allows you to specify an image to be used instead of the normal border around an element.

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

We will use the following image (called "border.png"):



The **border-image** property takes the image and slices it into nine sections, like a tic-tac-toe board. It then places the corners at the corners, and the middle sections are repeated or stretched as you specify.

Note: For **border-image** to work, the element also needs the **border** property set!

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

CSS3 Border Properties

Property	Description
border-image	A shorthand property for setting all the border-image-* properties
border-image-source	Specifies the path to the image to be used as a border
border-image-slice	Specifies how to slice the border image

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

<u>border-image-width</u>	Specifies the widths of the border image
<u>border-image-outset</u>	Specifies the amount by which the border image area extends beyond the border box
<u>border-image-repeat</u>	Specifies whether the border image should be repeated, rounded or stretched

Syntax:

`border-image: source slice width outset repeat|initial|inherit;`

CSS3 Backgrounds

CSS3 contains a few new background properties, which allow greater control of the background element.

In this chapter you will learn how to add multiple background images to one element.

You will also learn about the following new CSS3 properties:

background-size: The **background-size** property specifies the size of the background images.

There are four different syntaxes you can use with this property: the keyword syntax ("auto", "cover" and "contain"), the one-value syntax (sets the width of the image (height becomes "auto")), the two-value syntax (first value: width of the image, second value: height), and the multiple background syntax (separated with comma).

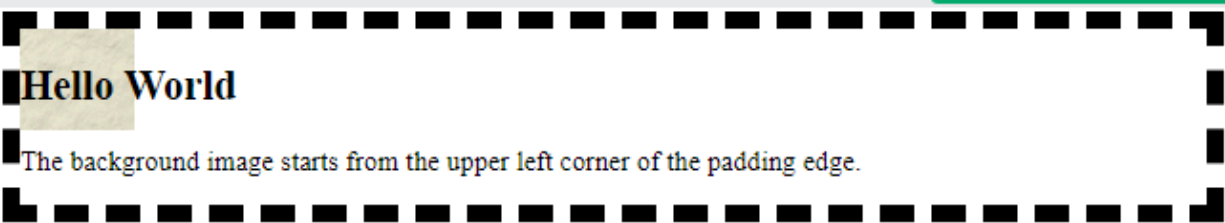
- **background-origin:** The **background-origin** property specifies the origin position (the background positioning area) of a background image

background-origin: padding-box (default):

-

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

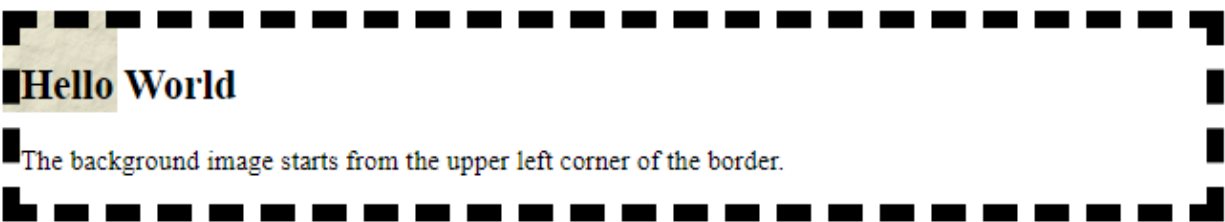
UNIT-3: Cascading Style sheet



■ Hello World

The background image starts from the upper left corner of the padding edge.

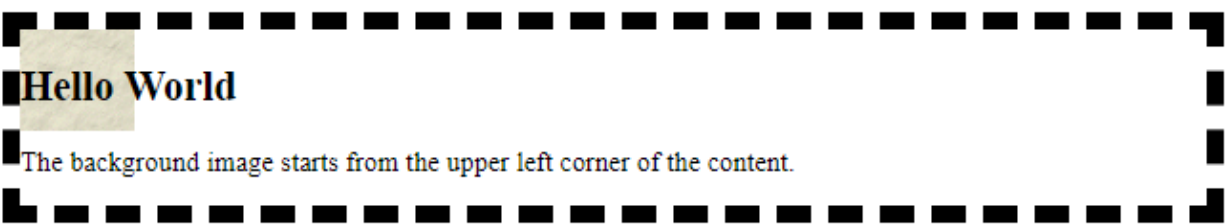
background-origin: border-box:



■ Hello World

The background image starts from the upper left corner of the border.

background-origin: content-box:



■ Hello World

The background image starts from the upper left corner of the content.

-
- **background-clip:** The **background-clip** property defines how far the background(color or image) should extend within an element.

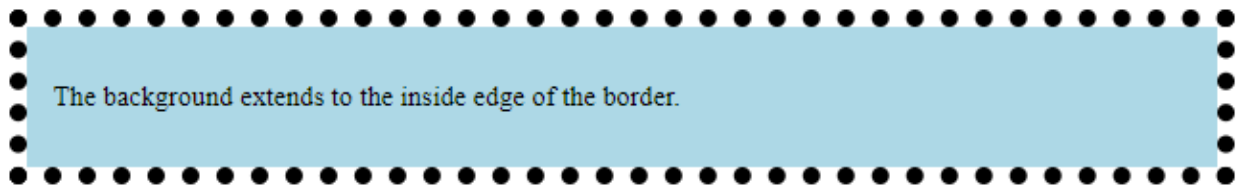
UNIT-3: Cascading Style sheet

The background-clip property defines how far the background should extend within an element.

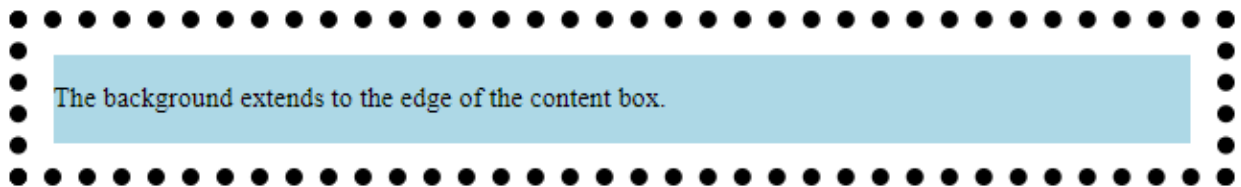
background-clip: border-box (this is default):



background-clip: padding-box:



background-clip: content-box:



- CSS3 Multiple Backgrounds
- CSS3 allows you to add multiple background images for an element, through the **background-image** property.
- The different background images are separated by commas, and the images are stacked on top of each other, where the first image is closest to the viewer.
- The following example has two background images, the first image is a flower (aligned to the bottom and right) and the second image is a paper background (aligned to the top-left corner):

Another way:

```
#example1 {  
  background: url(img_flwr.gif) right bottom no-repeat, url(paper.gif) left top repeat;  
}
```

CSS3 Background Size

The CSS3 **background-size** property allows you to specify the size of background images.

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

Before CSS3, the size of a background image was the actual size of the image. CSS3 allows us to re-use background images in different contexts.

The size can be specified in lengths, percentages, or by using one of the two keywords: contain or cover.

The following example resizes a background image to much smaller than the original image (using pixels):

The two other possible values for `background-size` are `contain` and `cover`.

The `contain` keyword scales the background image to be as large as possible (but both its width and its height must fit inside the content area). As such, depending on the proportions of the background image and the background positioning area, there may be some areas of the background which are not covered by the background image.

The `cover` keyword scales the background image so that the content area is completely covered by the background image (both its width and height are equal to or exceed the content area). As such, some parts of the background image may not be visible in the background positioning area.

CSS3 Gradients

CSS3 gradients let you display smooth transitions between two or more specified colors.

Earlier, you had to use images for these effects. However, by using CSS3 gradients you can reduce download time and bandwidth usage. In addition, elements with gradients look better when zoomed, because the gradient is generated by the browser.

CSS3 defines two types of gradients:

- **Linear Gradients (goes down/up/left/right/diagonally)**
- **Radial Gradients (defined by their center)**

CSS3 Linear Gradients

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

Syntax

`background: linear-gradient(direction, color-stop1, color-stop2, ...);`

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

Linear Gradient - Top to Bottom (this is default)

The following example shows a linear gradient that starts at the top. It starts red, transitioning to yellow:

Linear Gradient - Left to Right

The following example shows a linear gradient that starts from the left. It starts red, transitioning to yellow:

```
#grad {  
    background: red; /* For browsers that do not support gradients */  
    background: -webkit-linear-gradient(left, red , yellow); /* For Safari 5.1 to 6.0 */  
    background: -o-linear-gradient(right, red, yellow); /* For Opera 11.1 to 12.0 */  
    background: -moz-linear-gradient(right, red, yellow); /* For Firefox 3.6 to 15 */  
    background: linear-gradient(to right, red , yellow); /* Standard syntax */  
}
```

Linear Gradient - Diagonal

You can make a gradient diagonally by specifying both the horizontal and vertical starting positions.

The following example shows a linear gradient that starts at top left (and goes to bottom right). It starts red, transitioning to yellow:

```
#grad {  
    background: red; /* For browsers that do not support gradients */  
    background: -webkit-linear-gradient(left top, red, yellow); /* For Safari 5.1 to 6.0 */  
    background: -o-linear-gradient(bottom right, red, yellow); /* For Opera 11.1 to 12.0 */  
    background: -moz-linear-gradient(bottom right, red, yellow); /* For Firefox 3.6 to 15 */  
    background: linear-gradient(to bottom right, red, yellow); /* Standard syntax */  
}
```

Using Angles

If you want more control over the direction of the gradient, you can define an angle, instead of the predefined directions (to bottom, to top, to right, to left, to bottom right, etc.).

Syntax

```
background: linear-gradient(angle, color-stop1, color-stop2);
```

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

The angle is specified as an angle between a horizontal line and the gradient line.

The following example shows how to use angles on linear gradients:

```
#grad {  
    background: red; /* For browsers that do not support gradients */  
    background: -webkit-linear-gradient(-90deg, red, yellow); /* For Safari 5.1 to 6.0 */  
    background: -o-linear-gradient(-90deg, red, yellow); /* For Opera 11.1 to 12.0 */  
    background: -moz-linear-gradient(-90deg, red, yellow); /* For Firefox 3.6 to 15 */  
    background: linear-gradient(-90deg, red, yellow); /* Standard syntax */  
}
```

Using Multiple Color Stops

The following example shows a linear gradient (from top to bottom) with multiple color stops:

```
#grad {  
    background: red; /* For browsers that do not support gradients */  
    background: -webkit-linear-gradient(red, yellow, green); /* For Safari 5.1 to 6.0 */  
    background: -o-linear-gradient(red, yellow, green); /* For Opera 11.1 to 12.0 */  
    background: -moz-linear-gradient(red, yellow, green); /* For Firefox 3.6 to 15 */  
    background: linear-gradient(red, yellow, green); /* Standard syntax */  
}
```

CSS3 Shadow Effects

With CSS3 you can add shadow to text and to elements.

In this chapter you will learn about the following properties:

- text-shadow
- box-shadow

CSS3 Text Shadow

The CSS3 **text-shadow** property applies shadow to text.

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

Text shadow effect!

Next, add a color to the shadow:

Text shadow effect!

Example

```
h1 {  
    text-shadow: 2px 2px red;  
}
```

CSS3 box-shadow Property

The CSS3 **box-shadow** property applies shadow to elements.

In its simplest use, you only specify the horizontal shadow and the vertical shadow:

CSS3 Text

CSS3 contains several new text features.

In this chapter you will learn about the following text properties:

- **text-overflow**
- **word-wrap**
- **word-break**
- CSS3 Text Overflow
- The CSS3 **text-overflow** property specifies how overflowed content that is not displayed should be signaled to the user.
- CSS3 Word Wrapping
- The CSS3 **word-wrap** property allows long words to be able to be broken and wrap onto the next line.
- If a word is too long to fit within an area, it expands outside:
- CSS3 Word Breaking
- The CSS3 **word-break** property specifies line breaking rules.

CSS3 Web Fonts - The @font-face Rule

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

Web fonts allow Web designers to use fonts that are not installed on the user's computer.

When you have found/bought the font you wish to use, just include the font file on your web server, and it will be automatically downloaded to the user when needed.

Your "own" fonts are defined within the CSS3 **@font-face** rule.

Different Font Formats

TrueType Fonts (TTF)

TrueType is a font standard developed in the late 1980s, by Apple and Microsoft. TrueType is the most common font format for both the Mac OS and Microsoft Windows operating systems.

OpenType Fonts (OTF)

OpenType is a format for scalable computer fonts. It was built on TrueType, and is a registered trademark of Microsoft. OpenType fonts are used commonly today on the major computer platforms.

The Web Open Font Format (WOFF)

WOFF is a font format for use in web pages. It was developed in 2009, and is now a W3C Recommendation. WOFF is essentially OpenType or TrueType with compression and additional metadata. The goal is to support font distribution from a server to a client over a network with bandwidth constraints.

The Web Open Font Format (WOFF 2.0)

TrueType/OpenType font that provides better compression than WOFF 1.0.

SVG Fonts/Shapes

SVG fonts allow SVG to be used as glyphs when displaying text. The SVG 1.1 specification define a font module that allows the creation of fonts within an SVG document. You can also apply CSS to SVG documents, and the **@font-face** rule can be applied to text in SVG documents.

Embedded OpenType Fonts (EOT)

EOT fonts are a compact form of OpenType fonts designed by Microsoft for use as embedded fonts on web pages.

*Prof. Ankita Chaudhary
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

```
@font-face {  
    font-family: myFirstFont;  
    src: url(sansation_light.woff);  
}
```

CSS3 Transforms

CSS3 transforms allow you to translate, rotate, scale, and skew elements.

A transformation is an effect that lets an element change shape, size and position.

CSS3 supports 2D and 3D transformations.

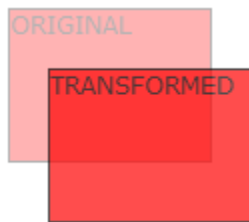
Mouse over the elements below to see the difference between a 2D and a 3D transformation:

CSS3 2D Transforms

In this chapter you will learn about the following 2D transformation methods:

- `translate()`
- `rotate()`
- `scale()`
- `skewX()`
- `skewY()`
- `matrix()`

The `translate()` Method



The `translate()` method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

The following example moves the `<div>` element 50 pixels to the right, and 100 pixels down from its current position:

The `rotate()` Method

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet



The `rotate()` method rotates an element clockwise or counter-clockwise according to a given degree.

The following example rotates the `<div>` element clockwise with 20 degrees:

The `scale()` Method



The `scale()` method increases or decreases the size of an element (according to the parameters given for the width and height).

The following example increases the `<div>` element to be two times of its original width, and three times of its original height:

CSS3 3D Transforms

CSS3 allows you to format your elements using 3D transformations.

Mouse over the elements below to see the difference between a 2D and a 3D transformation:

CSS3 3D Transforms

In this chapter you will learn about the following 3D transformation methods:

- `rotateX()`
- `rotateY()`
- `rotateZ()`

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

The rotateX() Method



The `rotateX()` method rotates an element around its X-axis at a given degree:

Example

```
div {  
    -webkit-transform: rotateX(150deg); /* Safari */  
    transform: rotateX(150deg);  
}
```

The rotateY() Method



The `rotateY()` method rotates an element around its Y-axis at a given degree:

Example

```
div {  
    -webkit-transform: rotateY(130deg); /* Safari */  
    transform: rotateY(130deg);  
}
```

The rotateZ() Method

The `rotateZ()` method rotates an element around its Z-axis at a given degree:

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

Example

```
div {  
    -webkit-transform: rotateZ(90deg); /*  
    transform: rotateZ(90deg);  
}
```

CSS3 Transitions

CSS3 transitions allows you to change property values smoothly (from one value to another), over a given duration.

Example: Mouse over the element below to see a CSS3 transition effect:

CSS3 Animations

CSS3 animations allows animation of most HTML elements without using JavaScript or Flash!

What are CSS3 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 CSS3 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 you specify CSS styles inside the **@keyframes** rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

Example

```
/* The animation code */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

CSS Images

Rounded Images

Use the **border-radius** property to create rounded images:



Example

Rounded Image:

```
img {
  border-radius: 8px;
}
```

Example

Circled Image:

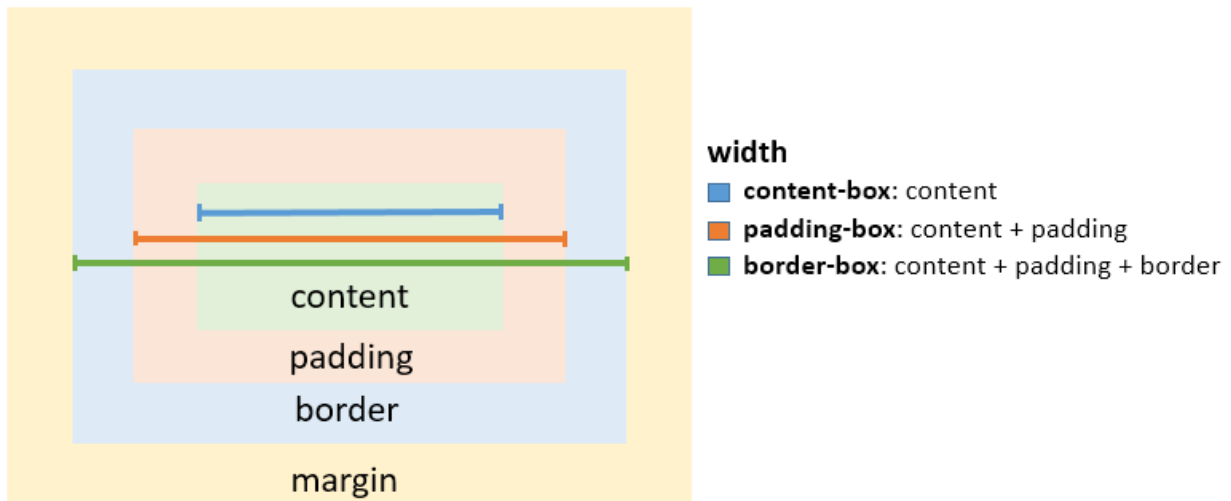
```
img {
  border-radius: 50%;
}
```

box-sizing:

UNIT-3: Cascading Style sheet

Fortunately, CSS allows us to change the box model with the `box-sizing` property for an element. There are three different values for the property available:

- `content-box`: The common box model - width and height only includes the content, not the padding or border
- `padding-box`: Width and height includes the content and the padding, but not the border
- `border-box`: Width and height includes the content, the padding as well as the border



Media Queries

Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.

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

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

Media Query Syntax

A media query consists of a media type and can contain one or more expressions, which resolve to either true or false.

```
@media not|only mediatype and (expressions) {  
    CSS-Code;  
}
```

The result of the query is true if the specified media type matches the type of device the document is being displayed on and all expressions in the media query are true. When a media query is true, the corresponding style sheet or style rules are applied, following the normal cascading rules.

Unless you use the not or only operators, the media type is optional and the **all** type will be implied.

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

You can also have different stylesheets for different media:

```
<link rel="stylesheet" media="mediatype and|not|only (expressions)" href="print.css">
```

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

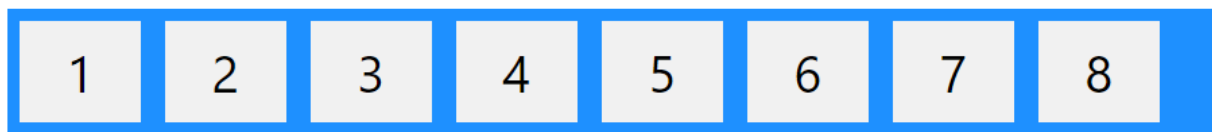
UNIT-3: Cascading Style sheet

CSS Flexbox Layout Module

Before the Flexbox Layout module, there were four layout modes:

- Block, for sections in a webpage
- Inline, for text
- Table, for two-dimensional table data
- Positioned, for explicit position of an element

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



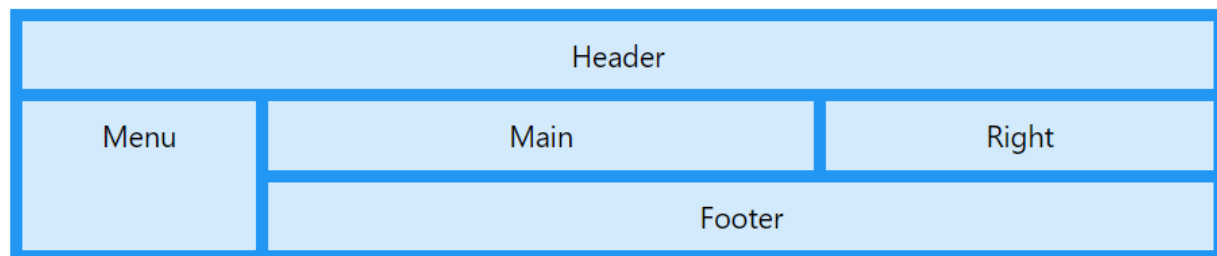
Flexbox Elements

To start using the Flexbox model, you need to first define a flex container.



The element above represents a flex container (the blue area) with three flex items.

CSS Grid Layout Module



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.

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

Grid Elements

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

Example

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

Responsive Web Design - Introduction

What is Responsive Web Design?

Responsive web design makes your web page look good on all devices.

Responsive web design uses only HTML and CSS.

Responsive web design is not a program or a JavaScript.

Designing For The Best Experience For All Users

Web pages can be viewed using many different devices: desktops, tablets, and phones. Your web page should look good, and be easy to use, regardless of the device.

Web pages should not leave out information to fit smaller devices, but rather adapt its content to fit any device:

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet



It is called responsive web design when you use CSS and HTML to resize, hide, shrink, enlarge, or move the content to make it look good on any screen.

Don't worry if you don't understand the example below, we will break down the code, step-by-step, in the next chapters:

What is The Viewport?

The viewport is the user's visible area of a web page.

The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.

Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size.

Then, when we started surfing the internet using tablets and mobile phones, fixed size web pages were too large to fit the viewport. To fix this, browsers on those devices scaled down the entire web page to fit the screen.

This was not perfect!! But a quick fix.

Setting The Viewport

HTML5 introduced a method to let web designers take control over the viewport, through the `<meta>` tag.

You should include the following `<meta>` viewport element in all your web pages:

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

*Prof. Ankita Chaunan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

This 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.

Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:

Size Content to The Viewport

Users are used to scroll websites vertically on both desktop and mobile devices - but not horizontally!

So, if the user is forced to scroll horizontally, or zoom out, to see the whole web page it results in a poor user experience.

Some additional rules to follow:

1. Do NOT use large fixed width elements - For example, if an image is displayed at a width wider than the viewport it can cause the viewport to scroll horizontally. Remember to adjust this content to fit within the width of the viewport.

2. Do NOT let the content rely on a particular viewport width to render well - Since screen dimensions and width in CSS pixels vary widely between devices, content should not rely on a particular viewport width to render well.

3. Use CSS media queries to apply different styling for small and large screens - Setting large absolute CSS widths for page elements will cause the element to be too wide for the viewport on a smaller device. Instead, consider using relative width values, such as width: 100%. Also, be careful of using large absolute positioning values. It may cause the element to fall outside the viewport on small devices.

Responsive Web Design - Grid-View

What is a Grid-View?

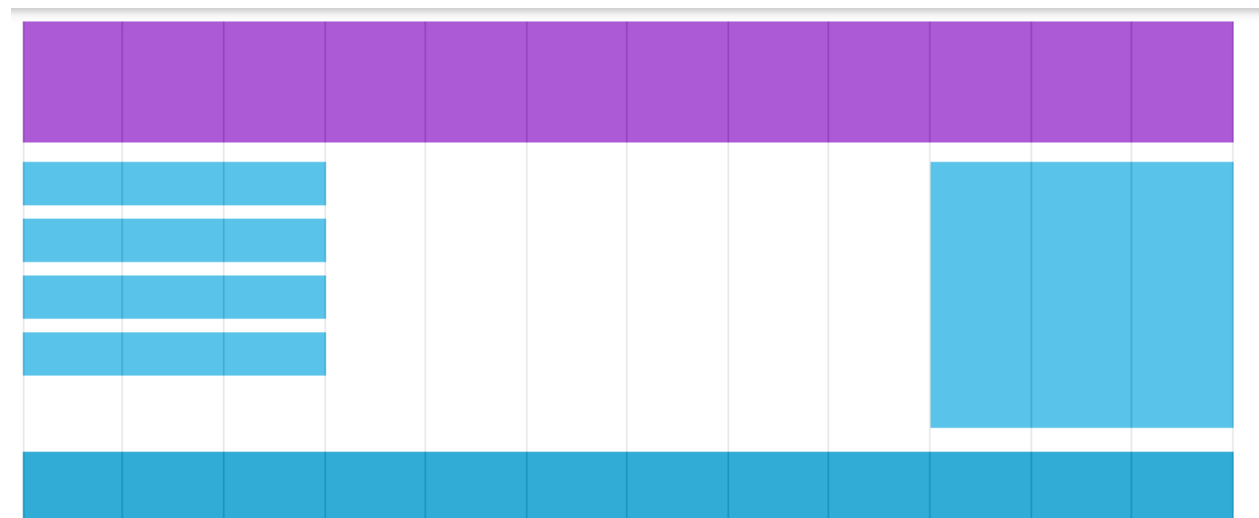
Many web pages are based on a grid-view, which means that the page is divided into columns:

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet



Using a grid-view is very helpful when designing web pages. It makes it easier to place elements on the page.



A responsive grid-view often has 12 columns, and has a total width of 100%, and will shrink and expand as you resize the browser window.

Building a Responsive Grid-View

Lets start building a responsive grid-view.

First ensure that all HTML elements have the **box-sizing** property set to **border-box**. This makes sure that the padding and border are included in the total width and height of the elements.

Add the following code in your CSS:

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*

UNIT-3: Cascading Style sheet

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

The following example shows a simple responsive web page, with two columns:

25%	75%
-----	-----

Example

```
.menu {  
  width: 25%;  
  float: left;  
}  
.main {  
  width: 75%;  
  float: left;  
}
```

Responsive Web Design - Images

Using The width Property

If the **width** property is set to a percentage and the **height** property is set to "auto", the image will be responsive and scale up and down:

Example

```
img {  
  width: 100%;  
  height: auto;  
}
```

Responsive Web Design - Templates

You can create your own temple. Or refer w3schools.com for Responsive Web Design – Templates.

*Prof. Ankita Chauhan
Assistant Professor
Department of Computer Engineering
MBIT*