

CSS

Roi Yehoshua
2018

Cascading Style Sheet

- ▶ **CSS** is a language that describes the style of an HTML document
 - ▶ CSS describes how HTML elements should be displayed
- ▶ Enables the separation of presentation and content
 - ▶ Including aspects such as the layout, colors, and fonts
- ▶ Enables multiple HTML pages to share the same styles
- ▶ Allows to present the same HTML page in different styles
 - ▶ for different rendering methods, such as on-screen, in print
 - ▶ for different screen resolutions and viewing devices
- ▶ The style definitions are normally saved in external .css files
 - ▶ Which are cached by the browser (allows faster loading times of the pages)

Same Page – Different Styles

Welcome to My Homepage

Use the menu to select different Stylesheets

- Stylesheet 1
- Stylesheet 2
- Stylesheet 3
- Stylesheet 4
- No Stylesheet

Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting different stylesheets in the menu, or by selecting one of the following links: [Stylesheet1](#), [Stylesheet2](#), [Stylesheet3](#), [Stylesheet4](#).

No Styles

This page uses DIV elements to group different sections of the HTML page. Click here to see how the page looks like with no stylesheet: [No Stylesheet](#).

Side-Bar

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Lorem ipsum dolor sit amet, consectetur 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 feugiat nulla facilisis.

Welcome to My Homepage

Use the menu to select different Stylesheets

Stylesheet 1

Stylesheet 2

Stylesheet 3

Stylesheet 4

No Stylesheet

Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting different stylesheets in the menu, or by selecting one of the following links: [Stylesheet1](#), [Stylesheet2](#), [Stylesheet3](#), [Stylesheet4](#).

No Styles

This page uses DIV elements to group different sections of the HTML page. Click here to see how the page looks like with no stylesheet: [No Stylesheet](#).

Lorem ipsum dolor sit amet, consectetur 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 feugiat nulla facilisis.

Welcome to My Homepage

Use the menu to select different Stylesheets

Stylesheet 1

Stylesheet 2

Stylesheet 3

Stylesheet 4

No Stylesheet

Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting different stylesheets in the menu, or by selecting one of the following links: [Stylesheet1](#), [Stylesheet2](#), [Stylesheet3](#), [Stylesheet4](#).

No Styles

This page uses DIV elements to group different sections of the HTML page. Click here to see how the page looks like with no stylesheet: [No Stylesheet](#).

Side-Bar

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Lorem ipsum dolor sit amet, consectetur 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 feugiat nulla facilisis.

Welcome to My Homepage

Use the menu to select different Stylesheets

Stylesheet 1

Stylesheet 2

Stylesheet 3

Stylesheet 4

No Stylesheet

Same Page Different Stylesheets

This is a demonstration of how different stylesheets can change the layout of your HTML page. You can change the layout of this page by selecting different stylesheets in the menu, or by selecting one of the following links: [Stylesheet1](#), [Stylesheet2](#), [Stylesheet3](#), [Stylesheet4](#).

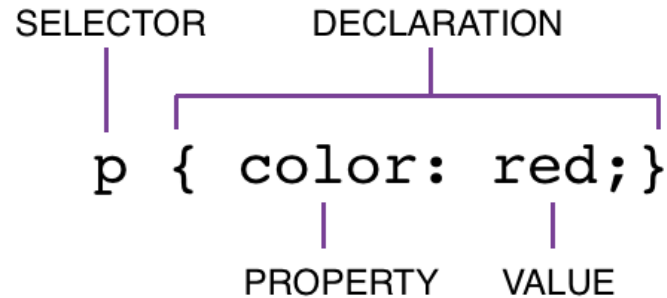
No Styles

This page uses DIV elements to group different sections of the HTML page. Click here to see how the page looks like with no stylesheet: [No Stylesheet](#).

Side-Bar

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

CSS Syntax



- ▶ A style sheet consists of a list of rules
- ▶ Each rule consists of one or more selectors, and a declaration block
- ▶ A declaration block consists of a list of *declarations* in braces
- ▶ Each declaration itself consists of a *property*, a colon (:), and a *value*
- ▶ If there are multiple declarations in a block, a semi-colon (;) must be inserted to separate each declaration
- ▶ CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more

CSS Selectors

- ▶ Universal : * { margin:0; padding:0; }
- ▶ Type : p { font-size:2em; }
- ▶ Class : .info { background:#ff0; }
 - ▶ Multiple : .info.error { color:#900; }
 - ▶ Element with class: p.info { color: #111; }
- ▶ ID : #info { background:#ff0; }
- ▶ Group : td, th, div { font-size:1em; }
- ▶ Descendent : div p { color:#f00; }
- ▶ Child : div > strong { color:#f00; }
- ▶ Adj sibling : p + p { color:#f00; }
- ▶ Attribute : input[type="text"] { margin-left: 5px; }

CSS Selectors Demo

```
*
{
  color:Red;
}

h2
{
  color:Blue;
}

li h2
{
  color:Green;
}

#myListItem1
{
  color:Lime;
}

li.myListItem
{
  color:Navy;
}

.myListItem
{
  color:Orange;
}
```

```
<h2>Css Demo</h2>
this is a css demo
<ul>
  <li>
    <h2>header in list</h2>
    list1 item1
  </li>
</ul>
<ul>
  <li id="myListItem1">list2 item1</li>
  <li class="myListItem">list2 item2</li>
</ul>
<span class="myListItem">test</span>
```

Css Demo

this is a css demo

- **header in list**

list1 item1

- list2 item1

- list2 item2

test

CSS - Where To Write?

- ▶ Inline
- ▶ In <style> tag in the header of the page
 - ▶ Applies to all elements of the same type
- ▶ In an external style sheet (.css file)
 - ▶ Can be shared among multiple HTML pages
 - ▶ Cached in the browser

```
<h1 style="color: #7E8F7C; font-size:50px">Site  
Header</h1>
```

```
<head>  
  <style>  
    h1 {  
      color: #7E8F7C;  
      font-size: 50px;  
    }  
  </style>  
</head>
```

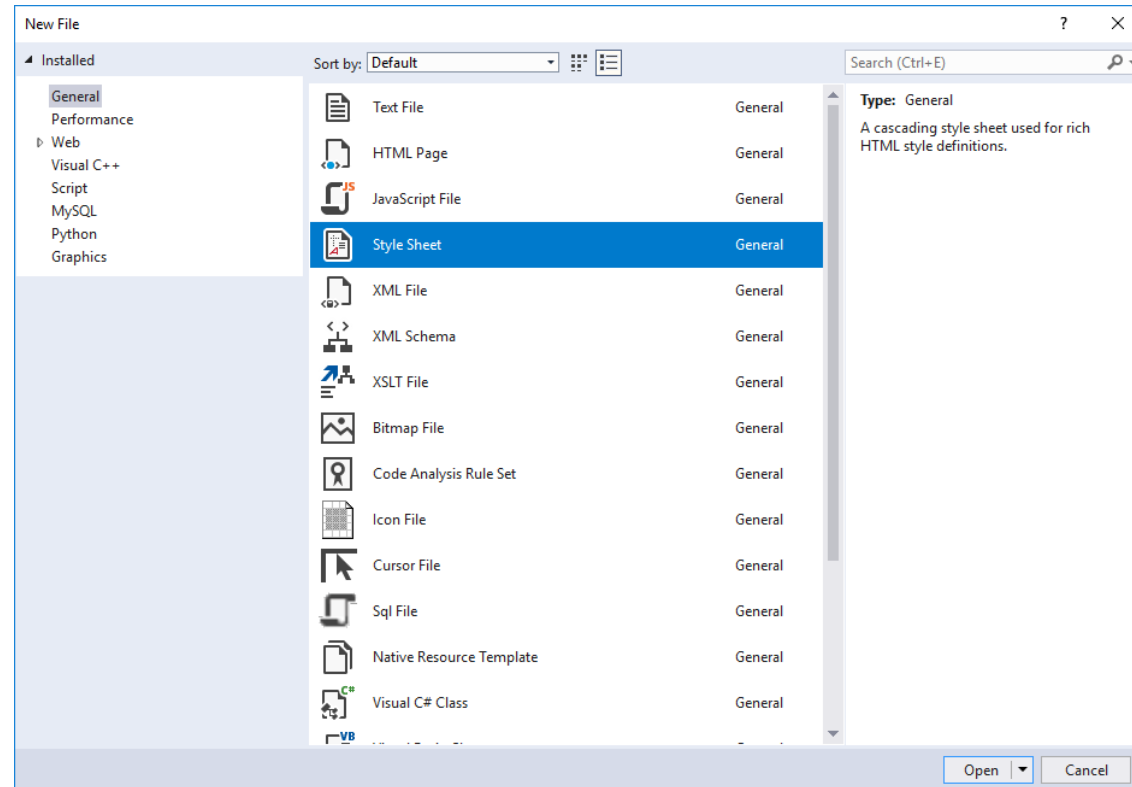
```
<head>  
  <link href="MyStyles.css" rel="stylesheet"/>  
</head>
```

MyStyles.css

```
h1 {  
  color: #7E8F7C;  
  font-size: 50px;  
}
```

Creating a Style Sheet File in VS

- ▶ Click File->New File
- ▶ Then choose Style Sheet



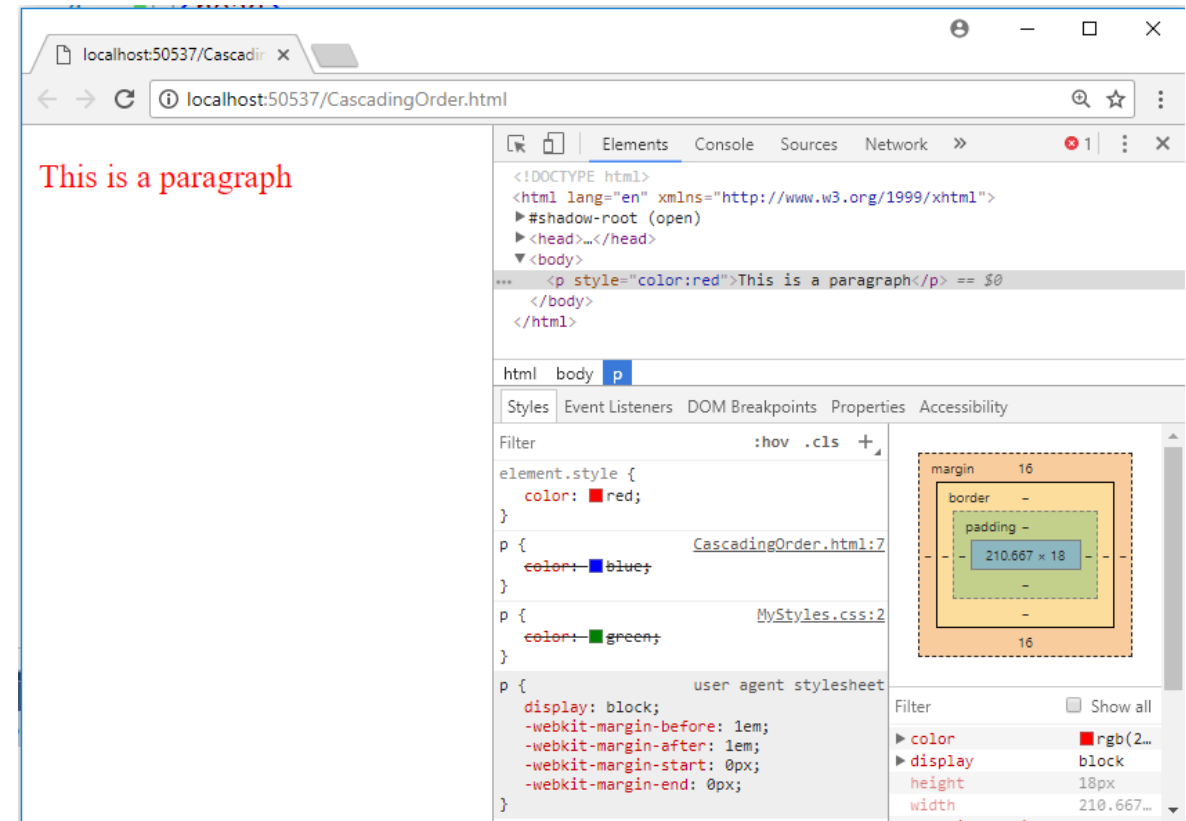
- ▶ Click Open
- ▶ Save the file in the same folder with your HTML pages

Cascading Order

- ▶ What style will be used when there is more than one style specified for an HTML element?
- ▶ Inline style (inside a specific HTML element) has the highest priority
- ▶ Then external and internal style sheets (in the head section)
- ▶ And lastly the browser default
- ▶ You can examine which styles have been overridden by which rules using the browser developer tools

Cascading Order

```
<head>
  <link href="MyStyles.css" rel="stylesheet" />
  <style>
    p {
      color: blue;
    }
  </style>
</head>
<body>
  <p style="color:red">This is a paragraph</p>
</body>
```



Specificity

- ▶ If there are two or more conflicting CSS rules that point to the same element, the browser checks which one is most specific and therefore wins out
- ▶ How to calculate specificity?
 - ▶ Start at 0
 - ▶ add 1000 for style attribute
 - ▶ add 100 for each ID
 - ▶ add 10 for each attribute, class or pseudo-class
 - ▶ add 1 for each element name or pseudo-element

Specificity

- ▶ Consider these three code fragments:

```
A: h1  
B: #content h1  
C: <h1 style="color: red">Heading</h1>
```

- ▶ The specificity of A is 1 (one element)
The specificity of B is 101 (one ID reference and one element)
The specificity of C is 1000 (inline styling)
- ▶ Since $1 < 101 < 1000$, the third rule (C) has a greater level of specificity, and therefore will be applied

CSS Comments

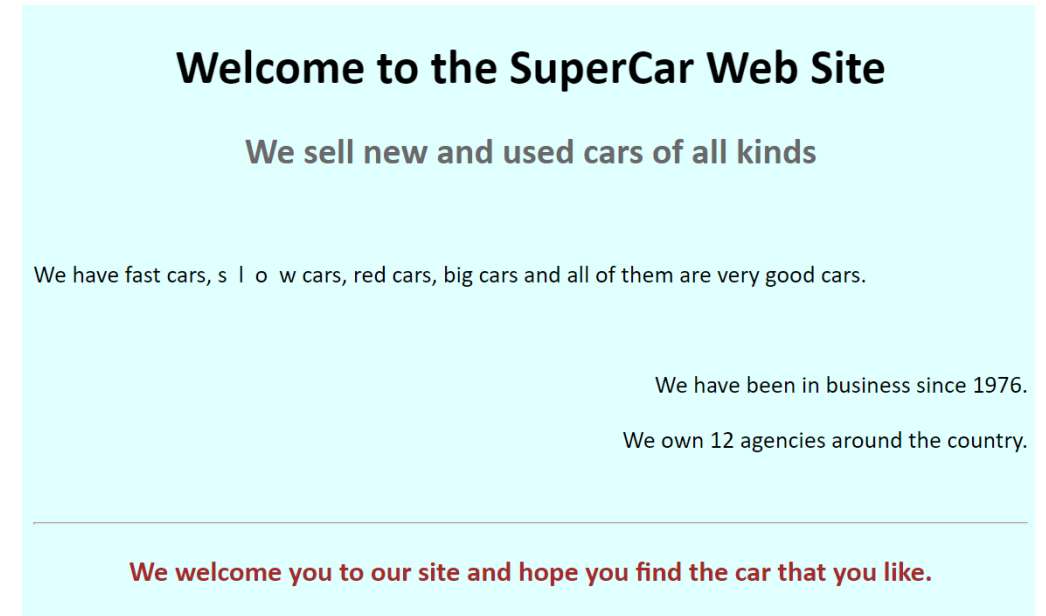
- ▶ A CSS comment starts with `/*` and ends with `*/`
- ▶ Comments can also span multiple lines:

```
p {  
    color: red;  
    /* This is a single-line comment */  
    text-align: center;  
}  
  
/* This is  
   a multi-line  
   comment */
```

Exercise (1)

- ▶ You are given the following HTML:

```
<body>
  <h1>Welcome to the SuperCar Web Site</h1>
  <h2 id="header2">We sell new and used cars of all
kinds</h2><br />
  <p>We have fast cars,
s   l   o   w cars, red cars, big
cars and all of them are very good cars.</p><br />
  <p class="right">We have been in business since 1976.</p>
  <p class="right">We own 12 agencies around the
country.</p>
  <br /><hr />
  <h3 id="header3">We welcome you to our site and hope you
find the car that you like.</h3>
</body>
```



- ▶ Use external CSS style sheet to make the page look like the page on the right
- ▶ Do not change the HTML code!

Backgrounds

- ▶ CSS background properties are used to define the background effects for elements
- ▶ CSS background properties:
 - ▶ background-color
 - ▶ background-image
 - ▶ background-repeat
 - ▶ background-attachment
 - ▶ background-position
 - ▶ background-size (CSS3)
 - ▶ background-origin (CSS3)
 - ▶ background-clip (CSS3)

Background Color

- ▶ The **background-color** property specifies the background color of an element
- ▶ A color is most often specified by:
 - ▶ a valid color name - like "red"
 - ▶ a HEX value - like "#ff0000"
 - ▶ an RGB value - like "rgb(255,0,0)"
- ▶ The background color of a page is set like this:

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

My Strong Page

This page has a light blue background color!

RGBA Colors

- ▶ RGBA color values are an extension of RGB color values with an alpha channel, which specifies the opacity for a color
- ▶ An RGBA color value is specified with: **rgba(*red*, *green*, *blue*, *alpha*)**
- ▶ The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all)

```
<h1 style="background-color:rgba(255, 99, 71, 0);">rgba(255, 99, 71, 0)</h1>  
<h1 style="background-color:rgba(255, 99, 71, 0.2);">rgba(255, 99, 71, 0.2)</h1>  
<h1 style="background-color:rgba(255, 99, 71, 0.4);">rgba(255, 99, 71, 0.4)</h1>  
<h1 style="background-color:rgba(255, 99, 71, 0.6);">rgba(255, 99, 71, 0.6)</h1>  
<h1 style="background-color:rgba(255, 99, 71, 0.8);">rgba(255, 99, 71, 0.8)</h1>  
<h1 style="background-color:rgba(255, 99, 71, 1);">rgba(255, 99, 71, 1)</h1>
```

rgba(255, 99, 71, 0)

rgba(255, 99, 71, 0.2)

rgba(255, 99, 71, 0.4)

rgba(255, 99, 71, 0.6)

rgba(255, 99, 71, 0.8)

rgba(255, 99, 71, 1)

Background Image

- ▶ The background-image property specifies an image to use as the background of an element
- ▶ By default, the image is repeated so it covers the entire element
- ▶ When using a background image, use an image that does not disturb the text
- ▶ The background image for a page can be set like this:

```
body {  
  background-image: url("images/bricks.png");  
}
```



Background Image - Repeat Horizontally or Vertically

- ▶ By default, the background-image property repeats an image both horizontally and vertically
- ▶ Some images should be repeated only horizontally or vertically, or they will look strange
- ▶ To repeat an image horizontally use **background-repeat: repeat-x;**
- ▶ To repeat an image vertically use **background-repeat: repeat-y;**

```
body {  
  background-image: url("gradient_bg.png");  
  background-repeat: repeat-x;  
}
```

Hello World!

Here, a background image is repeated only horizontally!

Background Image - Set position and no-repeat

- ▶ Showing the background image only once is specified by **background-repeat: none**
- ▶ The position of the image can be specified by the background-position property

```
body {  
  background-image: url("images/tree.png");  
  background-repeat: no-repeat;  
  background-position: right top;  
  margin-right: 200px;  
}
```

My Strong Page

This page has an image as the background!

Now the background image is only shown once, and positioned away from the text.

In this example we have also added a margin on the right side, so the background image will never disturb the text.



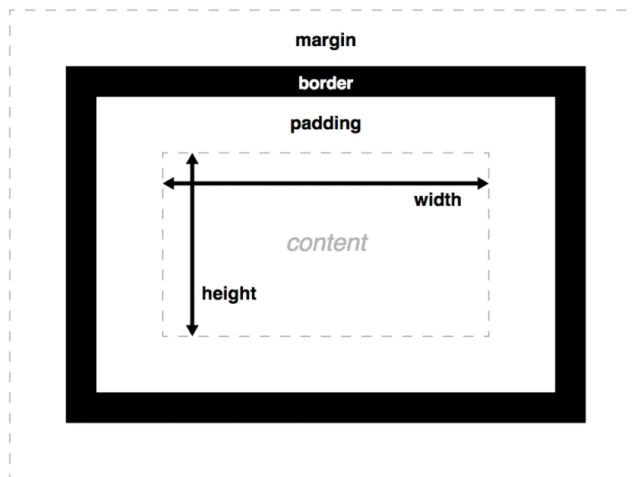
Background – Shorthand Property

- ▶ To shorten the code, it is also possible to specify all the background properties in one single property called background
- ▶ When using the shorthand property the order of the property values is:
 - ▶ background-color
 - ▶ background-image
 - ▶ background-repeat
 - ▶ background-attachment
 - ▶ background-position
- ▶ Example:

```
body {  
    background: #ffffff url("img_tree.png") no-repeat right top;  
}
```

CSS Box Model

- ▶ All HTML elements can be considered as boxes
- ▶ Each box consists of:
 - ▶ **Content** - The content of the box, where text and images appear
 - ▶ **Padding** - Clears an area around the content. The padding is transparent
 - ▶ **Border** - A border that goes around the padding and content
 - ▶ **Margin** - Clears an area outside the border. The margin is transparent

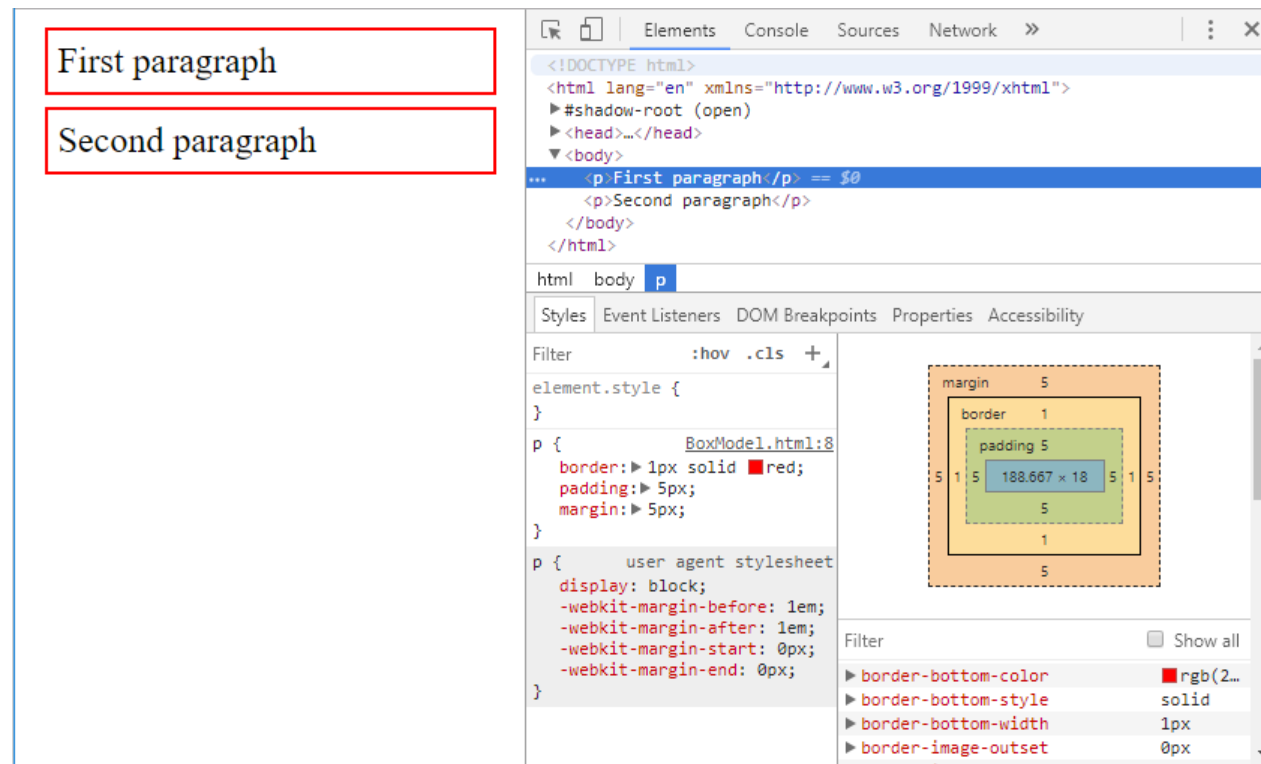


```
width:250px;  
padding:10px;  
border:5px solid gray;  
margin:10px;
```

Let's do the math:
250px (width)
+ 20px (left and right padding)
+ 10px (left and right border)
+ 20px (left and right margin)
= 300px

CSS Box Model In Developer Tools

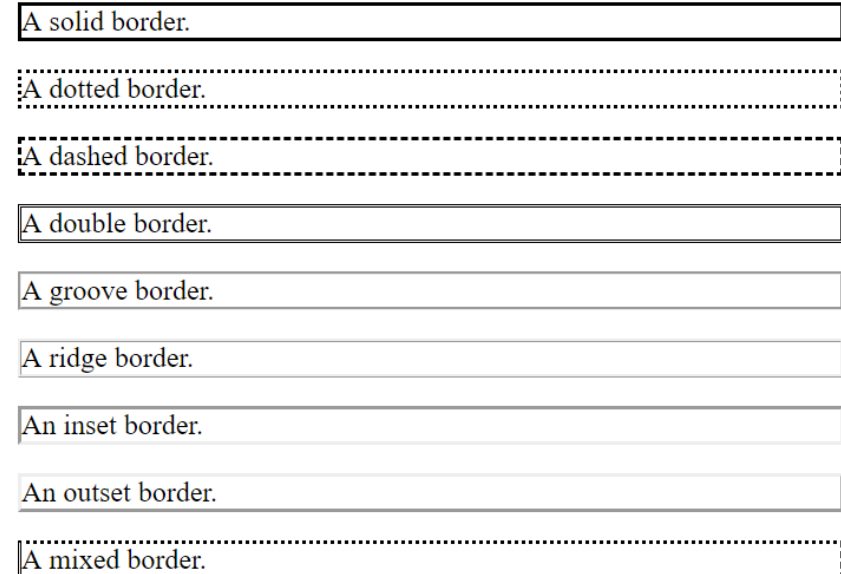
- ▶ You can examine the box model of each individual element on the page by opening up the browser developer tools and clicking on the elements in the DOM inspector



Borders

- ▶ The CSS border properties allow you to specify the style, width, and color of an element's border
- ▶ The **border-style** property specifies what kind of border to display
 - ▶ This property is required for all the other border properties to take affect
 - ▶ It can have from one to four values (for the top/right/bottom/left borders)

```
p.solid { border-style: solid; }  
p.dotted { border-style: dotted; }  
p.dashed { border-style: dashed; }  
p.double { border-style: double; }  
p.groove { border-style: groove; }  
p.ridge { border-style: ridge; }  
p.inset { border-style: inset; }  
p.outset { border-style: outset; }  
p.mix { border-style: dotted dashed solid double; }
```



Border Width

- ▶ The **border-width** property specifies the width of the four borders
- ▶ The width can be set as a specific size (in px, pt, cm, em, etc.) or by using one of the three pre-defined values: thin, medium, or thick
- ▶ The border-width property can have from one to four values (for the top/right/bottom/left borders)

```
p.one {  
  border-style: solid;  
  border-width: 5px;  
}  
p.two {  
  border-style: solid;  
  border-width: medium;  
}  
p.three {  
  border-style: solid;  
  border-width: 2px 10px 4px 20px;  
}
```



Border Color

- ▶ The **border-color** property is used to set the color of the four borders
- ▶ The border-color property can have from one to four values (for the top/right/bottom/left borders)
- ▶ If border-color is not set, it inherits the color of the element

```
p.color_one {  
    border-style: solid;  
    border-color: red;  
}  
  
p.color_two {  
    border-style: solid;  
    border-color: red green blue yellow;  
}
```

A solid red border

A solid multicolor border

Border – Individual Sides

- ▶ There are also properties for specifying each of the borders (top, right, bottom, and left)
- ▶ If the **border-style** property has four values: **border-style: dotted solid double dashed;**
 - ▶ top border is dotted
 - ▶ right border is solid
 - ▶ bottom border is double
 - ▶ left border is dashed
- ▶ If the border-style property has two values: **border-style: dotted solid;**
 - ▶ top and bottom borders are dotted
 - ▶ right and left borders are solid
- ▶ If the border-style property has one value: **border-style: dotted;**
 - ▶ all four borders are dotted
- ▶ The same works with **border-width** and **border-color**

Border – Individual Sides

```
p.individual_sides {  
    border-top-style: dotted;  
    border-right-style: solid;  
    border-bottom-style: dotted;  
    border-left-style: solid;  
}  
  
/* The same as: */  
p.individual_sides {  
    border-style: dotted solid;  
}
```

Two different border styles.

Border – Shorthand Property

- ▶ The **border** property is a shorthand property for the following individual border properties:
 - ▶ border-width
 - ▶ border-style (required)
 - ▶ border-color

```
p.border {  
  border: 5px solid red;  
}
```

This property is a shorthand property for border-width, border-style, and border-color.

- ▶ You can also specify all the individual border properties for just one side:

```
p.leftBorder {  
  border-left: 6px solid red;  
  background-color: lightgrey;  
}
```

This property is a shorthand property for border-left-width, border-left-style, and border-left-color.

Rounded Borders

- ▶ The **border-radius** property is used to add rounded borders to an element
 - ▶ This property is not supported in IE8 and earlier versions

```
p.normal {  
    border: 2px solid red;  
}  
p.round1 {  
    border: 2px solid red;  
    border-radius: 5px;  
}  
p.round2 {  
    border: 2px solid red;  
    border-radius: 8px;  
}  
p.round3 {  
    border: 2px solid red;  
    border-radius: 12px;  
}
```

Normal border

Round border

Rounder border

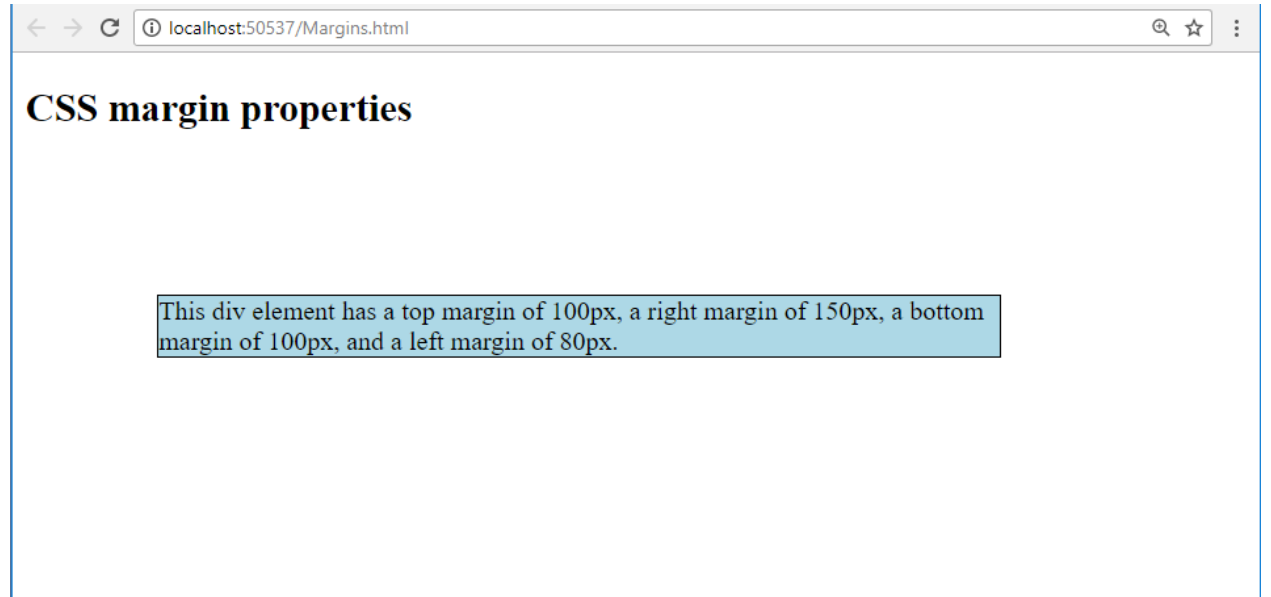
Roudest border

CSS Margins

- ▶ The CSS **margin** properties are used to create space around elements, outside of any defined borders
- ▶ CSS has properties for specifying the margin for each side of an element:
 - ▶ margin-top
 - ▶ margin-right
 - ▶ margin-bottom
 - ▶ margin-left
- ▶ All the margin properties can have the following values:
 - ▶ auto - the browser calculates the margin
 - ▶ length - specifies a margin in px, pt, cm, etc.
 - ▶ % - specifies a margin in % of the width of the containing element
 - ▶ inherit - specifies that the margin should be inherited from the parent element
- ▶ **Tip:** Negative values are allowed

Margin – Individual Sides

```
div {  
  border: 1px solid black;  
  margin-top: 100px;  
  margin-bottom: 100px;  
  margin-right: 150px;  
  margin-left: 80px;  
  background-color: lightblue;  
}
```



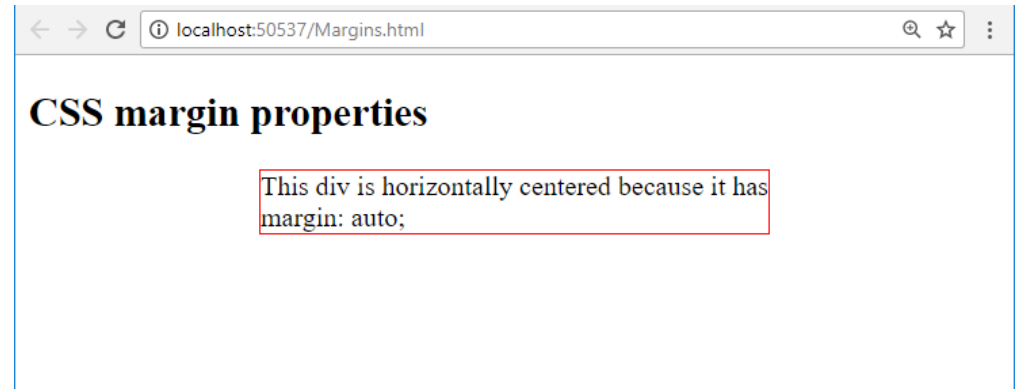
Margin – Shorthand Property

- ▶ The **margin** property is a shorthand property for the individual margin properties
- ▶ If the margin property has four values: **margin: 25px 50px 75px 100px;**
 - ▶ top margin is 25px
 - ▶ right margin is 50px
 - ▶ bottom margin is 75px
 - ▶ left margin is 100px
- ▶ If the margin property has two values: **margin: 25px 50px;**
 - ▶ top and bottom margins are 25px
 - ▶ right and left margins are 50px
- ▶ If the margin property has one value: **margin: 25px;**
 - ▶ all four margins are 25px

Margin auto

- ▶ You can set the **margin** property to **auto** to horizontally center the element within its container
- ▶ The element will then take up the specified width, and the remaining space will be split equally between the left and right margins

```
div2 {  
  width: 300px;  
  margin: auto;  
  border: 1px solid red;  
}
```



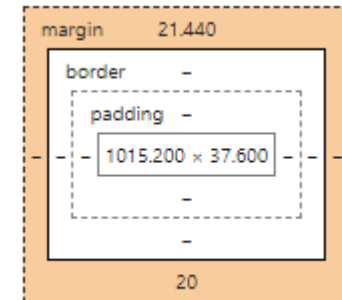
Margin Collapse

- ▶ When two boxes touch against one another, their top and bottom margins are collapsed into a single margin that is equal to the largest of the two margins
 - ▶ This does not happen on left and right margins
- ▶ Look at the following example:

```
h1 {  
    margin-bottom: 20px;  
}  
h2 {  
    margin-top: 20px;  
}
```

Heading 1

Heading 2



- ▶ In the example above, the `<h1>` element has a bottom margin of 20px and the `<h2>` element has a top margin set to 20px.
- ▶ Common sense would seem to suggest that the vertical margin between the `<h1>` and the `<h2>` would be a total of 40px. But due to margin collapse, the actual margin ends up being 20px.

CSS Padding

- ▶ The CSS **padding** properties are used to generate space around an element's content, inside of any defined borders
- ▶ CSS has properties for specifying the margin for each side of an element:
 - ▶ padding-top
 - ▶ padding-right
 - ▶ padding-bottom
 - ▶ padding-left
- ▶ All the margin properties can have the following values:
 - ▶ *length* - specifies a padding in px, pt, cm, etc.
 - ▶ % - specifies a padding in % of the width of the containing element
 - ▶ inherit - specifies that the padding should be inherited from the parent element
- ▶ **Note:** Negative values are not allowed

Padding – Individual Sides

```
div {  
  border: 1px solid black;  
  background-color: lightblue;  
  padding-top: 50px;  
  padding-right: 30px;  
  padding-bottom: 50px;  
  padding-left: 80px;  
}
```

CSS padding properties

This div element has a top padding of 50px, a right padding of 30px, a bottom padding of 50px, and a left padding of 80px.

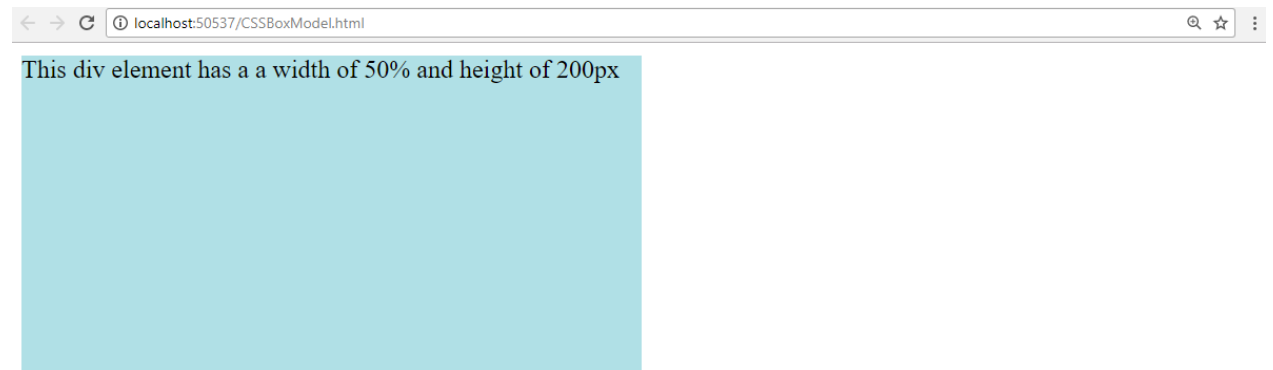
Padding – Shorthand Property

- ▶ The **padding** property is a shorthand property for the individual padding properties
- ▶ If the padding property has four values: **padding: 25px 50px 75px 100px;**
 - ▶ top padding is 25px
 - ▶ right padding is 50px
 - ▶ bottom padding is 75px
 - ▶ left padding is 100px
- ▶ If the padding property has two values: **padding: 25px 50px;**
 - ▶ top and bottom margins are 25px
 - ▶ right and left margins are 50px
- ▶ If the margin property has one value: **padding: 25px;**
 - ▶ all four paddings are 25px

Width and Height

- ▶ The **width** and **height** properties are used to set the width and height of an element
- ▶ They can be specified by:
 - ▶ **auto** (default) - the browser calculates the width and height
 - ▶ length values, like px, cm, etc.
 - ▶ percent (%) of the containing block
- ▶ The width and height properties do not include padding, borders, or margins
 - ▶ They set the size of the area inside the element!

```
div {  
  width: 50%;  
  height: 200px;  
  background-color: powderblue;  
}
```



Box Sizing

- ▶ The **box-sizing** property (CSS3) defines how the width and height of an element are calculated: should they include padding and borders, or not
- ▶ Possible values:
 - ▶ **content-box** (default) - The width and height properties (and min/max properties) includes only the content. Border and padding are not included.
 - ▶ **border-box** - The width and height properties (and min/max properties) includes content, padding and border
 - ▶ **inherit** - Inherits this property from its parent element

```
div.box {  
  box-sizing: border-box;  
  width: 50%;  
  border: 5px solid blue;  
  float: left;  
}
```

This div occupies the left half

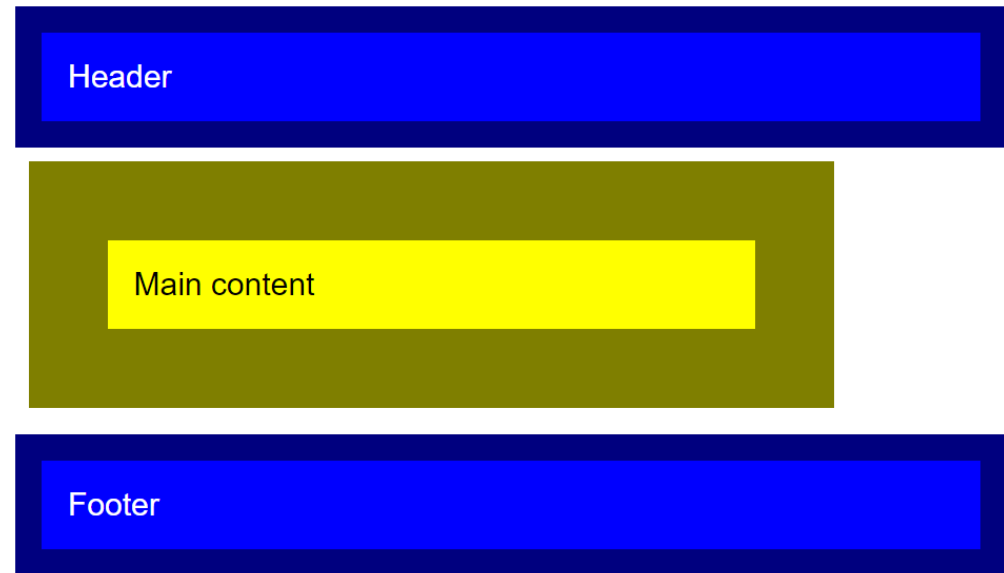
This div occupies the right half

Exercise (2)

- ▶ You are given the following HTML:

```
<div id="wrapper">  
  <header>Header</header>  
  <main>Main content</main>  
  <footer>Footer</footer>  
</div>
```

- ▶ Use CSS to make the page look like this:



Text Color

- ▶ The **color** property is used to set the color of the text
- ▶ The color is specified by:
 - ▶ a color name - like "red"
 - ▶ a HEX value - like "#ff0000"
 - ▶ an RGB value - like "rgb(255,0,0)"
- ▶ The default text color for a page is defined in the body selector

```
body {  
  color: blue;  
}  
h1 {  
  color: green;  
}
```

This is heading 1

The text of this paragraph is blue, as inherited from the body selector.

Text Alignment

- ▶ The **text-align** property is used to set the horizontal alignment of a text
- ▶ A text can be left or right aligned, centered, or justified
 - ▶ left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left
- ▶ The following example shows center aligned, and left and right aligned text:

```
h2.center {  
  text-align: center;  
}  
  
h2.left {  
  text-align: left;  
}  
  
h2.right {  
  text-align: right;  
}
```

Heading 2 (left)

Heading 1 (center)

Heading 3 (right)

Text Alignment

- ▶ When the **text-align** property is set to **justify**, each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers):

```
div {  
  border: 1px solid black;  
  padding: 10px;  
  width: 200px;  
  height: 200px;  
  text-align: justify;  
}
```

In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since. 'Whenever you feel like criticizing anyone,' he told me, 'just remember that all the people in this world haven't had the advantages that you've had.'

Text Decoration

- ▶ The **text-decoration** property is used to set or remove decorations from text
- ▶ Possible values of text-decoration:
 - ▶ overline
 - ▶ line-through
 - ▶ underline
 - ▶ It is not recommended to underline text that is not a link, as this often confuses the reader
 - ▶ none – often used to removed underlines from links

```
h2.overline {  
    text-decoration: overline;  
}  
h2.linethrough {  
    text-decoration: line-through;  
}  
h2.underline {  
    text-decoration: underline;  
}
```

This is heading 1

~~This is heading 2~~

This is heading 3

Text Direction

- ▶ The **direction** property is used to change the text direction of an element
- ▶ Use **rtl** for languages written from right to left (like Hebrew or Arabic), and **ltr** for those written from left to right (like English and most other languages)

```
p.rtl {  
  direction: rtl;  
}
```

This is the default text direction.

.This is right-to-left text direction

CSS Fonts

- ▶ The CSS font properties define the font family, boldness, size, and the style of a text
- ▶ There are two types of font family names:
 - ▶ **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
 - ▶ **font family** - a specific font family (like "Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman	Serif fonts have small lines at the ends on some characters
	Georgia	
Sans-serif	Arial	"Sans" means without - these fonts do not have the lines at the ends of characters
	Verdana	
Monospace	Courier New	All monospace characters have the same width
	Lucida Console	



- ▶ On computer screens, sans-serif fonts are considered easier to read than serif fonts.

Font Family

- ▶ The font family of a text is set with the **font-family** property
- ▶ The font-family property should hold several font names as a "fallback" system
 - ▶ If the browser does not support the first font, it tries the next font, and so on.
 - ▶ Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available
- ▶ **Note:** If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman"

```
p.serif {  
    font-family: "Times New Roman", Times, serif;  
}  
p.sansserif {  
    font-family: Arial, Helvetica, sans-serif;  
}
```

This paragraph is shown in the Times New Roman font.

This paragraph is shown in the Arial font.

Font Style

- ▶ The **font-style** property is mostly used to specify italic text
- ▶ This property has three values:
 - ▶ normal - The text is shown normally
 - ▶ italic - The text is shown in italics
 - ▶ oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

```
p.normal {  
    font-style: normal;  
}  
p.italic {  
    font-style: italic;  
}  
p.oblique {  
    font-style: oblique;  
}
```

This is a paragraph in normal style.

This is a paragraph in italic style.

This is a paragraph in oblique style.

Font Size

- ▶ The **font-size** property sets the size of the text
- ▶ The font-size value can be an absolute (using px or pt), or relative size (using em or %)
- ▶ To maximize accessibility, it is generally best to use values that are relative to the user's default font size
- ▶ To allow users to resize the text (in the browser menu), many developers use **em**
 - ▶ 1em is equal to the current font size
 - ▶ The default text size in browsers is 16px. So, the default size of 1em is 16px.
 - ▶ The size can be calculated from pixels to em using this formula: $pixels/16=em$

```
h1.larger {  
  font-size: 2.5em; /* 40px/16=2.5em */  
}
```

This heading has a default font size (2em)

This heading has a larger font size (2.5em)

Font Weight

- ▶ The **font-weight** property specifies the weight of a font. Possible values:
 - ▶ Normal - Normal font weight. Same as 400.
 - ▶ Bold - Bold font weight. Same as 700.
 - ▶ Lighter - One font weight lighter than the parent element (among the available weights of the font)
 - ▶ Bolder - One font weight heavier than the parent element (among the available weights of the font)
 - ▶ 100, 200, 300, 400, 500, 600, 700, 800, 900 - Numeric font weights
- ▶ Some fonts are only available in normal and bold

```
p.normal {  
    font-weight: normal;  
}  
p.light {  
    font-weight: lighter;  
}  
p.thick {  
    font-weight: bold;  
}  
p.thicker {  
    font-weight: 900;  
}
```

This is a paragraph.

This is a paragraph.

This is a paragraph.

This is a paragraph.

Font Style

- ▶ The **font-style** property is mostly used to specify italic text
- ▶ This property has three values:
 - ▶ normal - The text is shown normally
 - ▶ italic - The text is shown in italics
 - ▶ oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

```
p.normal {  
    font-style: normal;  
}  
p.italic {  
    font-style: italic;  
}  
p.oblique {  
    font-style: oblique;  
}
```

This is a paragraph in normal style.

This is a paragraph in italic style.

This is a paragraph in oblique style.

Pseudo-Classes

- ▶ A pseudo-class is used to define a special state of an element
- ▶ For example, it can be used to:
 - ▶ Style an element when a user mouses over it
 - ▶ Style visited and unvisited links differently
 - ▶ Style an element when it gets focus
- ▶ The syntax of pseudo-classes:

```
selector:pseudo-class {  
    property:value;  
}
```

- ▶ An example of using the :hover pseudo-class on a <div> element:

```
div:hover {  
    background-color: blue;  
}
```



Mouse Over Me

Anchor Pseudo-Classes

- ▶ Links can be displayed in different ways:

```
/* unvisited link */
a:link {
    color: red;
}
/* visited link */
a:visited {
    color: green;
}
/* mouse over link */
a:hover {
    color: hotpink;
}
/* selected link */
a:active {
    color: blue;
}
```

This is a link

This is a link

This is a link

This is a link

- ▶ Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective, and a:active MUST come after a:hover in the CSS definition.

Pseudo-classes and CSS Classes

- ▶ Pseudo-classes can be combined with CSS classes
- ▶ When you hover over the link in the example, it will change color:

```
a.highlight:hover {  
    color: darkgoldenrod;  
}
```

```
<p><a class="highlight" href="Page2.html">Page2</a></p>
```

Page2

Pseudo-Elements

- ▶ A CSS pseudo-element is used to style specified parts of an element.
- ▶ For example, it can be used to:
 - ▶ Style the first letter, or line, of an element
 - ▶ Insert content before, or after, the content of an element
- ▶ The syntax of pseudo-elements:

```
selector::pseudo-element {  
    property:value;  
}
```

- ▶ The double colon replaced the single-colon notation for pseudo-elements in CSS3

```
h1::before {  
    content: url(images/smiley.png);  
}
```

😄 **This is a heading**

The ::before pseudo-element inserts content before the content of an element.

CSS Icons

- ▶ The simplest way to add an icon to your page is with an icon library, such as Font Awesome
- ▶ Add the name of the specified icon class to any inline HTML element (like)
 - ▶ No downloading or installation is required!

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
</head>
<body>
  <span class="fa fa-heart"></span>
  <span class="fa fa-car"></span>
  <span class="fa fa-file"></span>
  <span class="fa fa-bars"></span><br/>
  <span class="fa fa-cloud"></span>
  <span class="fa fa-cloud" style="font-size:24px;color:red;"></span>
  <span class="fa fa-cloud" style="font-size:36px;color:lightblue;"></span>
</body>
</html>
```



CSS Layout – The display Property

- ▶ The **display** property specifies if/how an element is displayed
- ▶ Every HTML element has a default display depending on what type of element it is
 - ▶ The default display value for most elements is **block** or **inline**
- ▶ A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can)
 - ▶ Examples of block-level elements: `<div>`, `<h1>` - `<h6>`, `<p>`, `<form>`, `<header>`, `<footer>`, `<section>`
- ▶ An inline element does not start on a new line and only takes up as much width as necessary
 - ▶ Examples of inline elements: ``, `<a>`, ``

Override The Default Display Value

- ▶ Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.
- ▶ A common example is making inline elements for horizontal menus:

```
li {  
  display: inline;  
}
```

Display a list of links as a horizontal menu:

[HTML](#) [CSS](#) [JavaScript](#)

- ▶ Setting the display property of an element only changes **how the element is displayed**, NOT what kind of element it is
 - ▶ e.g., an inline element with display: block; is not allowed to have other block elements inside it

Inline-Block

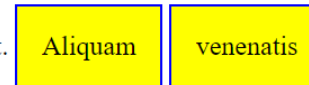
- ▶ **display: inline-block** allows to set a width and height on an inline element
- ▶ Also, the top and bottom margins/paddings are respected, in contrast to display:inline
- ▶ Compared to display: block, the major difference is that display: inline-block does not add a line-break after the element, so the element can sit next to other elements
- ▶ The following example shows the different behavior of display: inline, display: inline-block and display: block:

Inline vs. Inline-Block vs. Block

```
span.a {  
  display: inline;  
  width: 70px;  
  height: 70px;  
  padding: 15px;  
  border: 1px solid blue;  
  background-color: yellow;  
}  
span.b {  
  display: inline-block;  
  width: 70px;  
  height: 70px;  
  padding: 15px;  
  border: 1px solid blue;  
  background-color: yellow;  
}  
span.c {  
  display: block;  
  width: 70px;  
  height: 70px;  
  padding: 15px;  
  border: 1px solid blue;  
  background-color: yellow;  
}
```

display: inline

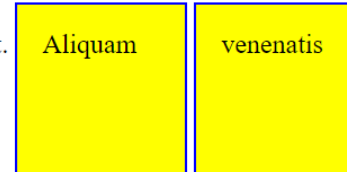
Lorem ipsum dolor sit amet, consectetur adipiscing elit.



gravida nisl sit amet facilisis.

display: inline-block

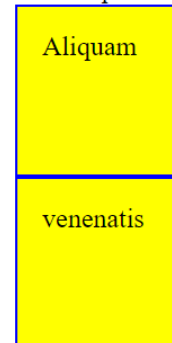
Lorem ipsum dolor sit amet, consectetur adipiscing elit.



gravida nisl sit amet facilisis.

display: block

Lorem ipsum dolor sit amet, consectetur adipiscing elit.



gravida nisl sit amet facilisis.

Hide an Element

- ▶ Hiding an element can be done by setting the **display** property to **none**
 - ▶ The element will be hidden, and the page will be displayed as if the element is not there

```
h1.hidden {  
  display: none;  
}  
  
<h1>This is a visible heading</h1>  
<h1 class="hidden">This is a hidden heading</h1>
```

This is a visible heading

Notice that the h1 element with display: none; does not take up any space.

- ▶ **visibility:hidden;** also hides an element
 - ▶ However, the element will still take up the same space as before

```
h1.hidden2 {  
  visibility: hidden;  
}  
  
<h1>This is a visible heading</h1>  
<h1 class="hidden2">This is a hidden heading</h1>
```

This is a visible heading

Notice that the hidden heading still takes up space.

CSS Layout – The position Property

- ▶ The position property specifies the type of positioning method used for an element
- ▶ There are five different position values:
 - ▶ Static (default)
 - ▶ relative
 - ▶ fixed
 - ▶ absolute
 - ▶ sticky
- ▶ Elements are then positioned using the top, bottom, left, and right properties.
 - ▶ These positions work differently depending on the position value

Static Position

- ▶ HTML elements are positioned static by default
- ▶ An element with static position is positioned according to the normal flow of the page
- ▶ Static positioned elements are not affected by the top, bottom, left, and right properties

```
div.static {  
  position: static;  
  border: 3px solid #73AD21;  
}
```

position: static;

This div element has position: static;

Relative Position

- ▶ An element with **position: relative;** is positioned relative to its normal position
- ▶ Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position
- ▶ Other content will not be adjusted to fit into any gap left by the element

```
div.relative {  
  position: relative;  
  left: 30px;  
  border: 3px solid #73AD21;  
}
```

position: relative;

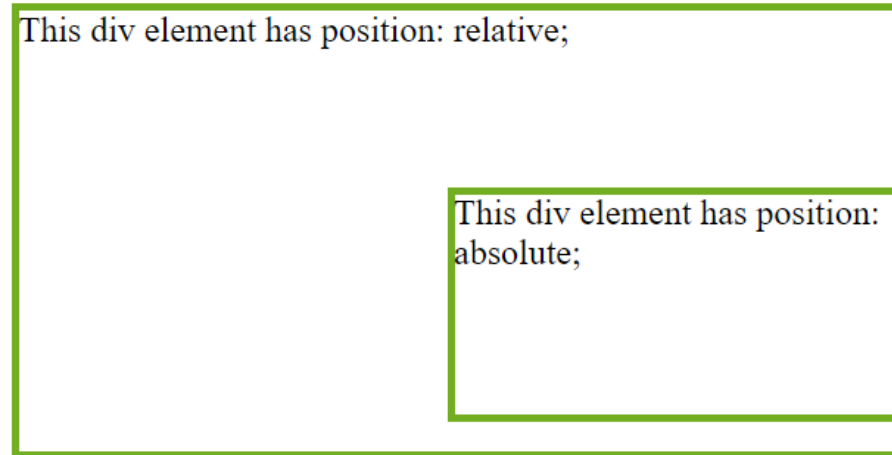
This div element has position: relative;

Absolute Position

- ▶ An element with **position: absolute;** is positioned relative to the nearest positioned ancestor (i.e., an element whose position is anything except static)
- ▶ If it has not positioned ancestors, it uses the document body

```
div.relative {  
  position: relative;  
  width: 400px;  
  height: 200px;  
  border: 3px solid #73AD21;  
}  
  
div.absolute {  
  position: absolute;  
  top: 80px;  
  right: 0;  
  width: 200px;  
  height: 100px;  
  border: 3px solid #73AD21;  
}
```

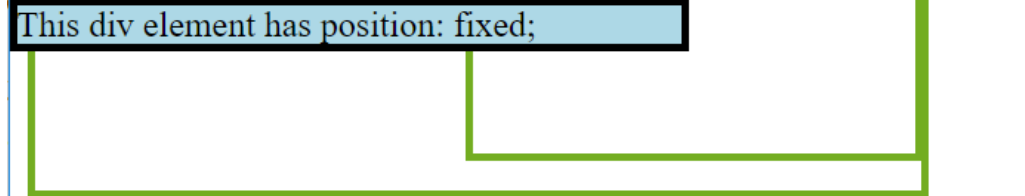
position: absolute;



Fixed Position

- ▶ 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
- ▶ A fixed element does not leave a gap in the page where it would normally have been located

```
div.fixed {  
  position: fixed;  
  top: 0;  
  left: 0;  
  width: 300px;  
  border: 3px solid black;  
  background-color:  
lightblue;  
}
```



This diagram illustrates a fixed element. A light blue rectangular box with a thick black border is positioned at the top of the page. The text "This div element has position: fixed;" is written inside the box. A green L-shaped line extends from the bottom-right corner of the box, indicating the element's fixed position relative to the viewport.

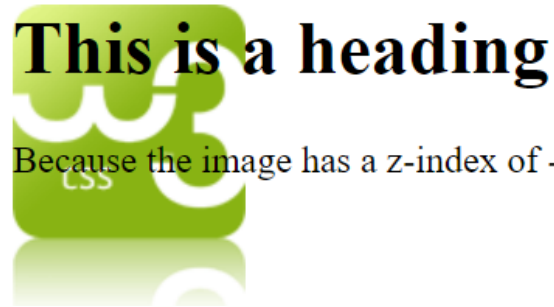
position: fixed;

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.

Overlapping Elements

- ▶ When elements are positioned, they can overlap other elements
- ▶ The **z-index** property specifies the stack order of an element (which element should be placed in front of, or behind, the others)
- ▶ An element can have a positive or negative stack order:

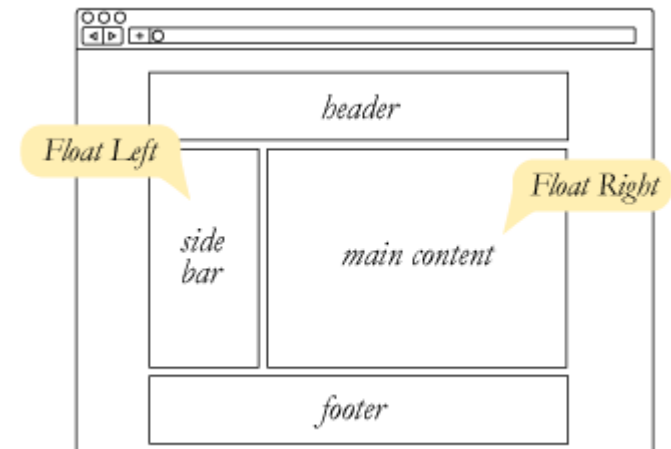
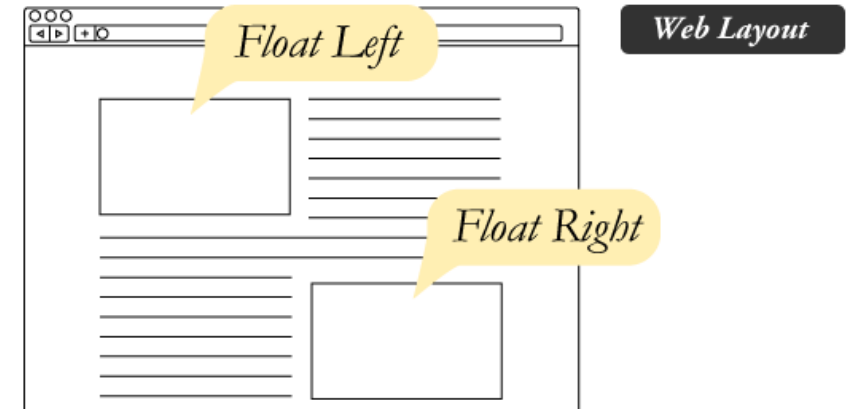
```
img {  
  position: absolute;  
  left: 0;  
  top: 0;  
  width: 100px;  
  z-index: -1;  
}
```



Because the image has a z-index of -1, it will be placed behind the text.

Float

- ▶ With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it
- ▶ The elements **after** the floating element will flow around it
 - ▶ The **clear** property turns off the floating
- ▶ The elements before the floating element will not be affected
- ▶ Aside from the simple example of wrapping text around images, floats can be used to create **entire web layouts**



Float

- ▶ The following example specifies that an image should float to the **left** in a text:

```
img {  
    float: left;  
}
```

In this example, the image will float to the left in the paragraph, and the text in the paragraph will wrap around the image.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

- ▶ The following example specifies that an image should float to the **right** in a text:

```
img {  
    float: right;  
}
```

In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap around the image.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.



The clear Property

- ▶ The **clear** property specifies whether an element can be next to floating elements that precede it or must be moved down (cleared) below them
- ▶ The clear property can have one of the following values:
 - ▶ none - the element is *not* moved down to clear past floating elements (default)
 - ▶ left - the element is moved down to clear past *left* floats
 - ▶ right - the element is moved down to clear past *right* floats
 - ▶ both - the element is moved down to clear past *both* left and right floats

The clear Property

```
.div1 {  
  float: left;  
  width: 100px;  
  height: 50px;  
  margin: 10px;  
  border: 3px solid #73AD21;  
}  
.div2 {  
  border: 1px solid red;  
}  
.div3 {  
  float: left;  
  width: 100px;  
  height: 50px;  
  margin: 10px;  
  border: 3px solid #73AD21;  
}  
.div4 {  
  border: 1px solid red;  
  clear: left;  
}
```

Without clear



div1

div2 - Notice that div2 is after div1 in the HTML code. However, since div1 floats to the left, the text in div2 flows around div1.

With clear



div3

div4 - Here, clear: left; moves div4 down below the floating div3. The value "left" clears elements floated to the left. You can also clear "right" and "both".

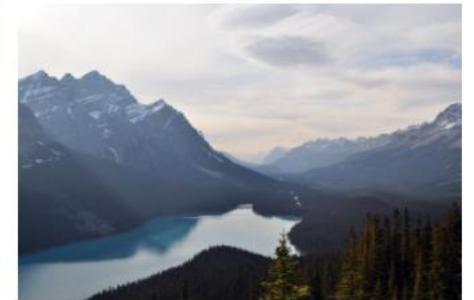
Images Side By Side

- ▶ With the float property, it is easy to float images side by side:

```
.img-container {  
  float: left;  
  width: 33.33%;  
  padding: 5px;  
  box-sizing: border-box;  
}  
  
img {  
  width: 100%;  
}
```

```
<div class="img-container">  
    
</div>
```

Images Side by Side



Exercise (3)

- ▶ Use CSS to create an image gallery with 4 pictures
- ▶ Add a description text below each image
- ▶ When the user hovers over an image, its borders will change color to black
- ▶ Clicking on an image will open a new page displaying the image in its full size



Fjords



A forest



Northern lights



Mountains

Overflow

- ▶ The **overflow** property controls what happens to content that is too big to fit into a specified area
- ▶ It has the following values:
 - ▶ visible - Default. The overflow is not clipped. It renders outside the element's box
 - ▶ hidden - The overflow is clipped, and the rest of the content will be invisible
 - ▶ scroll - The overflow is clipped, but a scrollbar is added to see the rest of the content
 - ▶ auto - If overflow is clipped, a scrollbar should be added to see the rest of the content
- ▶ The overflow property only works for block elements with a specified height

```
div {  
  width: 250px;  
  height: 50px;  
  background-color: #eee;  
  overflow: auto;  
}
```

You can use the overflow property when you want to have better control of the layout. The overflow property

Exercise (4)

- ▶ You are given a list of hyperlinks:

```
<ul>
  <li><a href="#home" class="active">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>
```

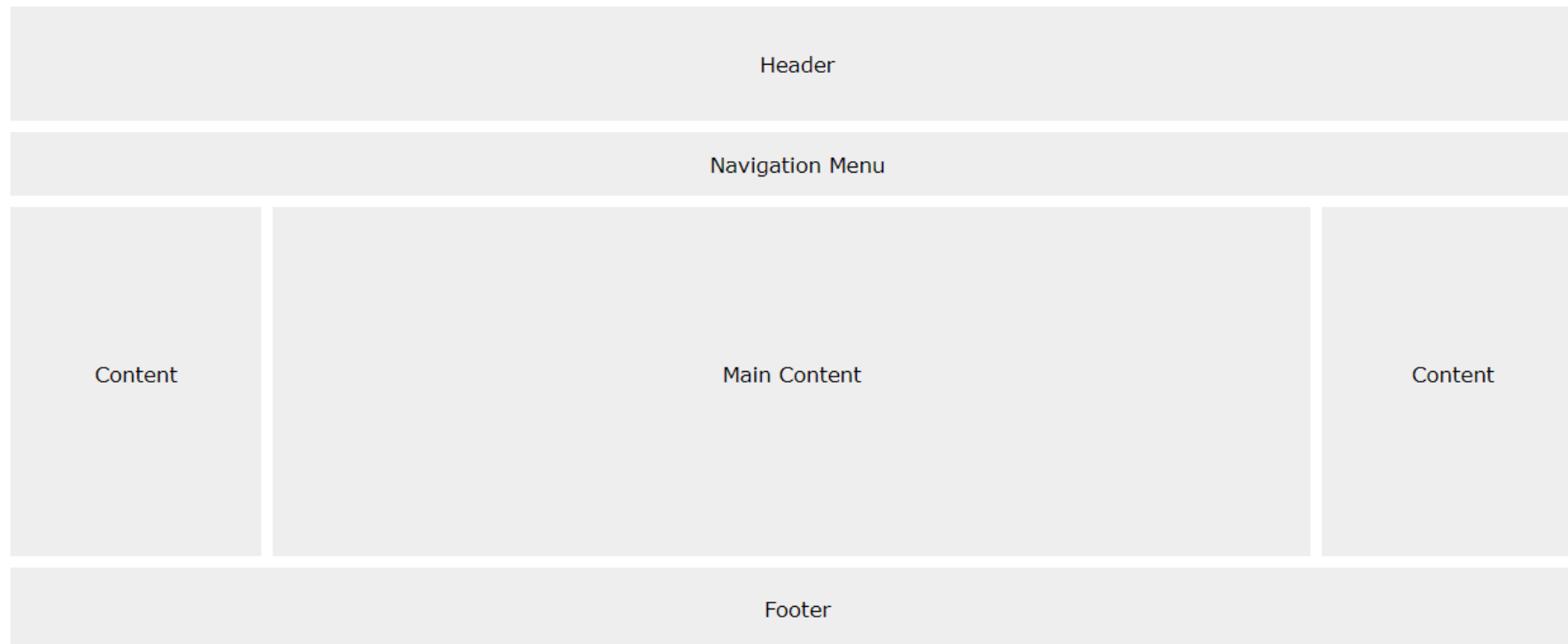
- ▶ Use float to create the following horizontal menu:



- ▶ Try not to specify the width of each (hint: use the overflow property)

Website Layout

- ▶ A website is often divided into headers, menus, content and a footer:




Header

- ▶ A header is usually located at the top of the website (or right below a top navigation menu). It often contains a logo or the website name:

```
header {  
  background-color: #f1f1f1;  
  padding: 20px;  
  text-align: center;  
}
```

```
<header>  
  <h1>Header</h1>  
</header>
```



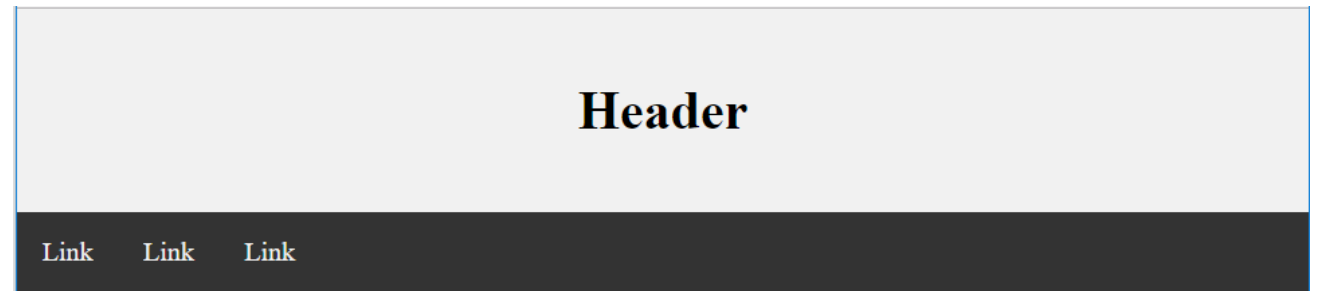
Header

Navigation Bar

- ▶ A navigation bar contains a list of links to help visitors navigating through your website:

```
nav {  
  overflow: hidden;  
  background-color: #333;  
}  
  
nav a {  
  float: left;  
  display: block;  
  color: #f2f2f2;  
  text-align: center;  
  padding: 15px;  
  text-decoration: none;  
}  
  
nav a:hover {  
  background-color: #ddd;  
  color: black;  
}
```

```
<nav>  
  <a href="#">Link</a>  
  <a href="#">Link</a>  
  <a href="#">Link</a>  
</nav>
```



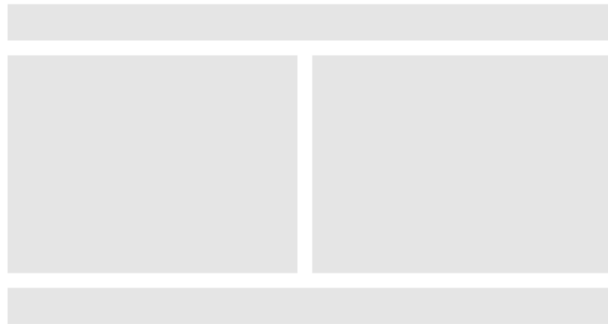
Content

- ▶ The layout in this section, often depends on the target users
- ▶ The most common layout is one (or combining them) of the following:
 - ▶ **1-column** (often used for mobile browsers)
 - ▶ **2-column** (often used for tablets and laptops)
 - ▶ **3-column layout** (only used for desktops)

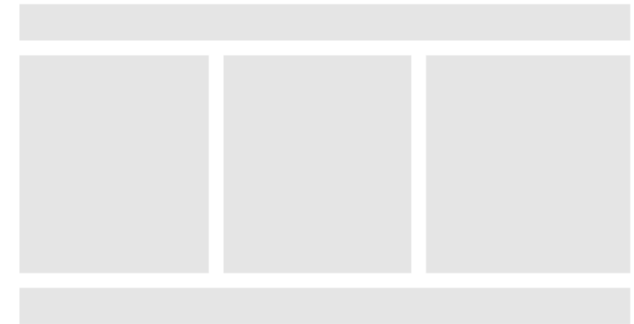
1-column:



2-column:



3-column:

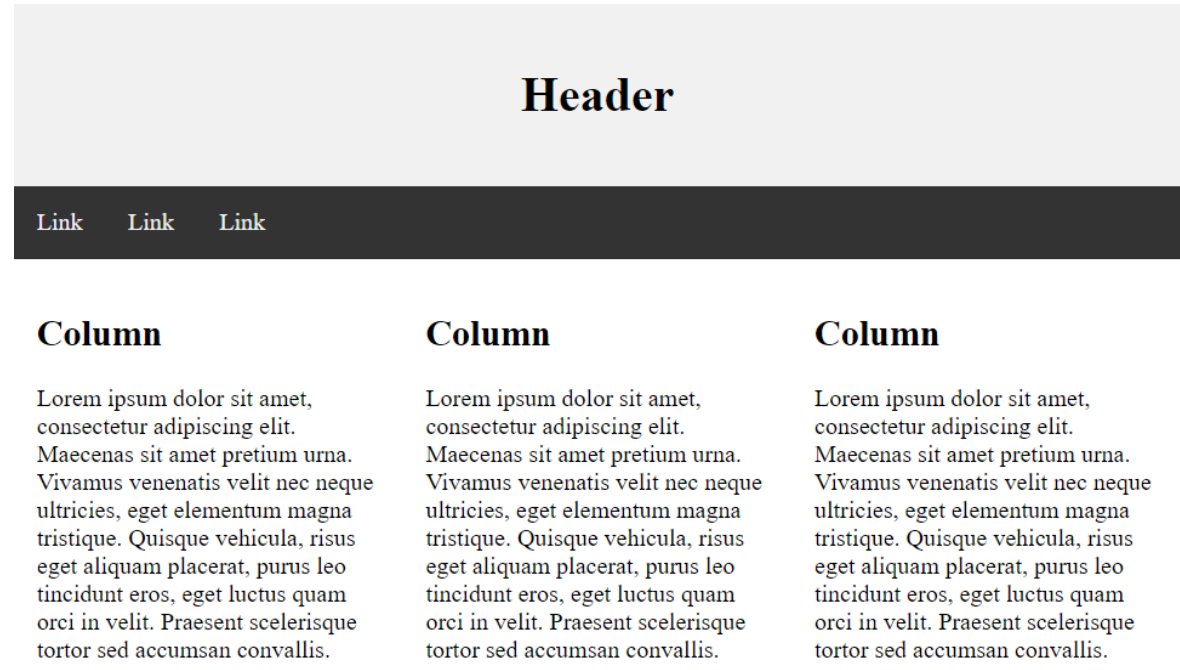


Content

- For example, we will create a 3-column layout:

```
.column {  
  float: left;  
  width: 33.33%;  
  padding: 15px;  
}
```

```
<main>  
  <div class="column">  
    <h2>Column</h2>  
    <p>Lorem ipsum...</p>  
  </div>  
  <div class="column">  
    <h2>Column</h2>  
    <p>Lorem ipsum...</p>  
  </div>  
  <div class="column">  
    <h2>Column</h2>  
    <p>Lorem ipsum...</p>  
  </div>  
</main>
```



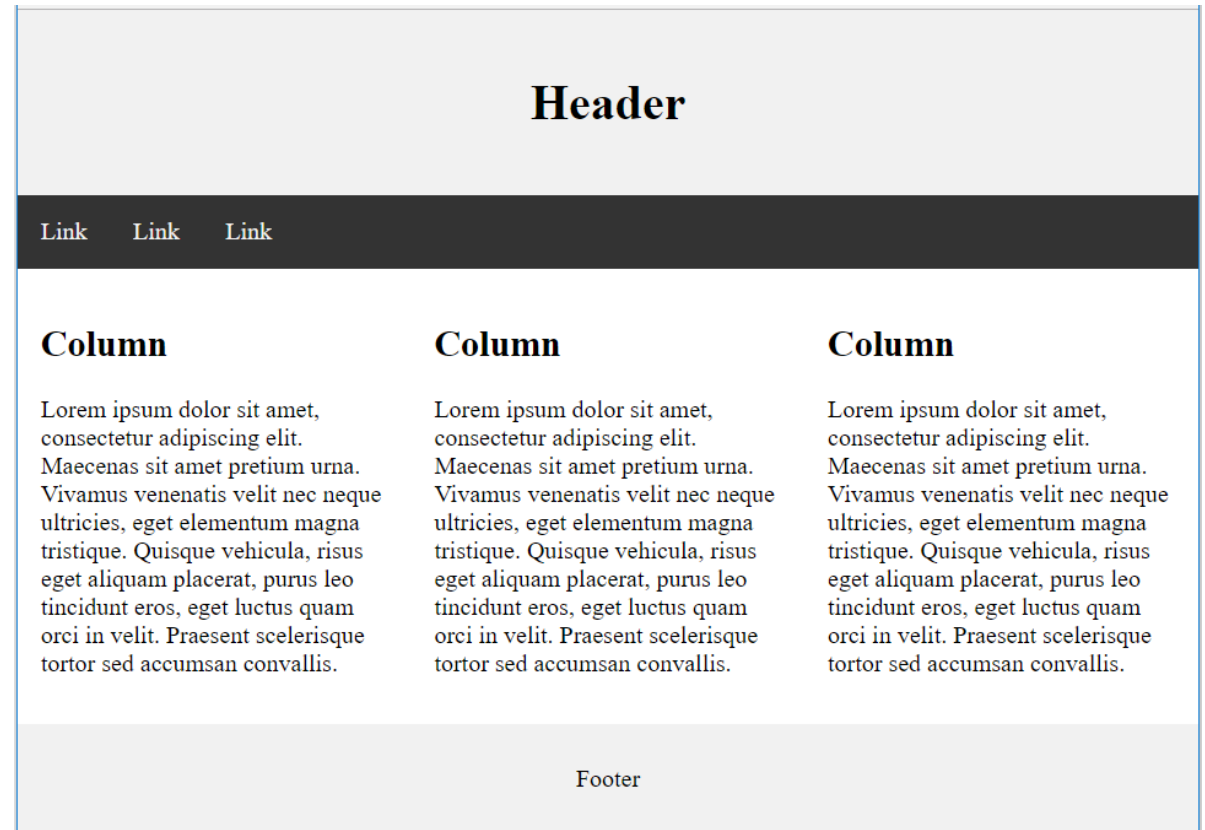
To create a 2-column layout, change the width to 50%, to create a 4-column layout, use 25%, etc.

Footer

- ▶ The footer is placed at the bottom of your page. It often contains information like copyright and contact info:

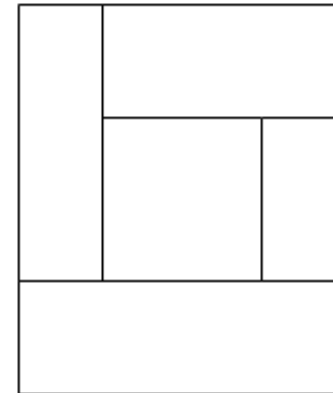
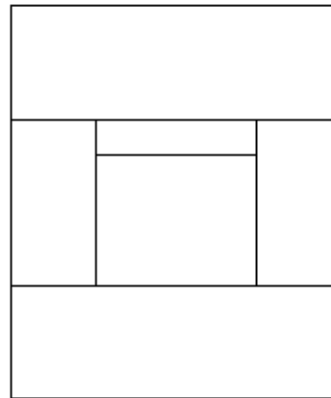
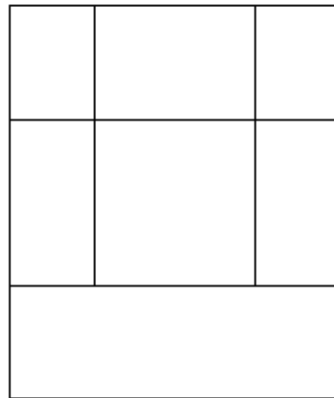
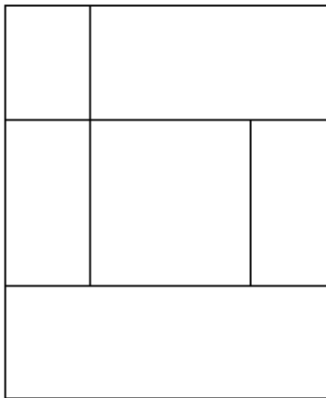
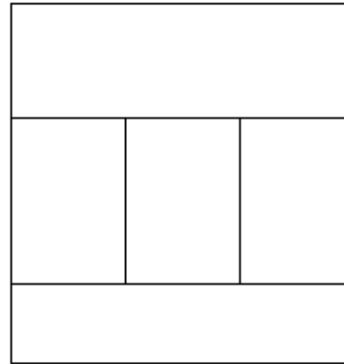
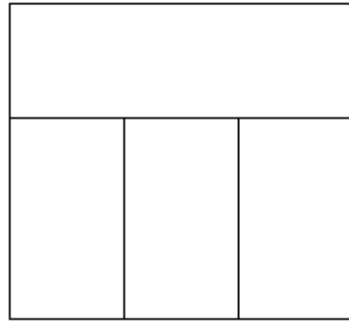
```
footer {  
  background-color: #f1f1f1;  
  padding: 10px;  
  text-align: center;  
  clear: both; /* clear the floats */  
}
```

```
<footer>  
  <p>Footer</p>  
</footer>
```



Exercise (5)

- ▶ Create the following page layouts using only the float property:



Exercise (6)

- ▶ Create the following page using the page layout techniques we've learned:

Beijing

The Flight

The City

The Culture

The Food

The City

Beijing is the capital of the People's Republic of China, the world's second most populous city proper, and most populous capital city.

The city, located in northern China, is governed as a direct-controlled municipality under the national government with 16 urban, suburban, and rural districts

As a city combining both modern and traditional architecture, Beijing is an ever-changing megacity rich in history but also truly modern.

Footer Text

CSS Gradients

- ▶ CSS gradients let you display smooth transitions between two or more specified colors
- ▶ CSS defines two types of gradients:
 - ▶ **Linear Gradients** (goes down/up/left/right/diagonally)
 - ▶ **Radial Gradients** (defined by their center)
- ▶ IE9 and earlier versions do not support gradients

Lorem ipsum dolor sit amet, vim iisque interesset no, inani integre deleniti ex mei, cu usu sint nominati. At lorem malis habemus duo, in recusabo nominati ius. Tollit scripserit sit ex, duo ei labore definiebas. Eu qui tempor aperiri blandit. Nam id legendos scribentur, no erant congue exerci qui, an quo accumsan oportere consequuntur. Per et inani noluisse eloquentiam, te nec habemus appareat maiestatis, sea id meis aliquip impedit.

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, ...);
```

Linear Gradients

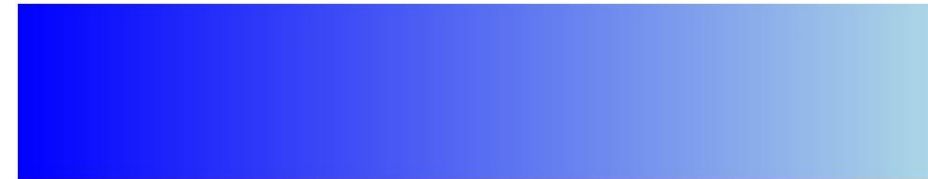
▶ Top to bottom (default)

```
#grad1 {  
  background: linear-gradient(blue, lightblue);  
}
```



▶ Left to right

```
#grad2 {  
  background: linear-gradient(to right, blue,  
lightblue);  
}
```



▶ Diagonal

```
#grad3 {  
  background: linear-gradient(to bottom right,  
blue, lightblue);  
}
```



Using Angles

- ▶ If you want more control over the direction of the gradient, you can define an angle, instead of the predefined directions

- ▶ Syntax:

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

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

```
#grad4 {  
  background: linear-gradient(30deg, yellow, red);  
}
```



Using Transparency

- ▶ CSS gradients also support transparency, which can be used to create fading effect.
- ▶ To add transparency, we use the `rgba()` function to define the color stops
- ▶ The following example shows a linear gradient that starts from the left fully transparent, transitioning to full color red:

```
#grad5 {  
  background: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));  
}
```



Using Multiple Color Stops

- ▶ The following example shows how to create a linear gradient (from left to right) with the color of the rainbow and some text:

```
#grad6 {  
  background: linear-gradient(to right, red, orange, yellow, green, blue, indigo, violet);  
  text-align: center;  
  color: #888888;  
  font-size: 40px;  
  font-weight: bold  
}  
  
<div id="grad6">  
  Gradient Background  
</div>
```



Radial Gradients

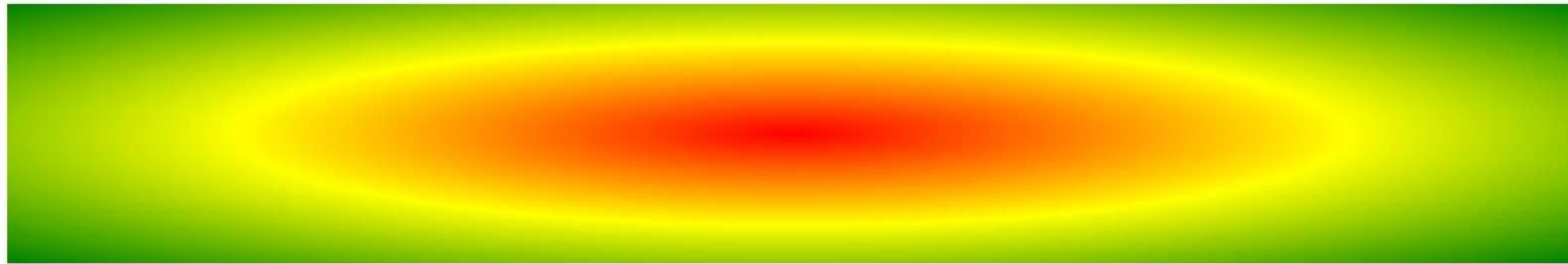
- ▶ A radial gradient is defined by its center

- ▶ Syntax:

```
background: radial-gradient(shape size at position, start-color, ..., last-color);
```

- ▶ By default, shape is ellipse, size is farthest-corner, and position is center
- ▶ The following example shows a radial gradient with evenly spaced color stops:

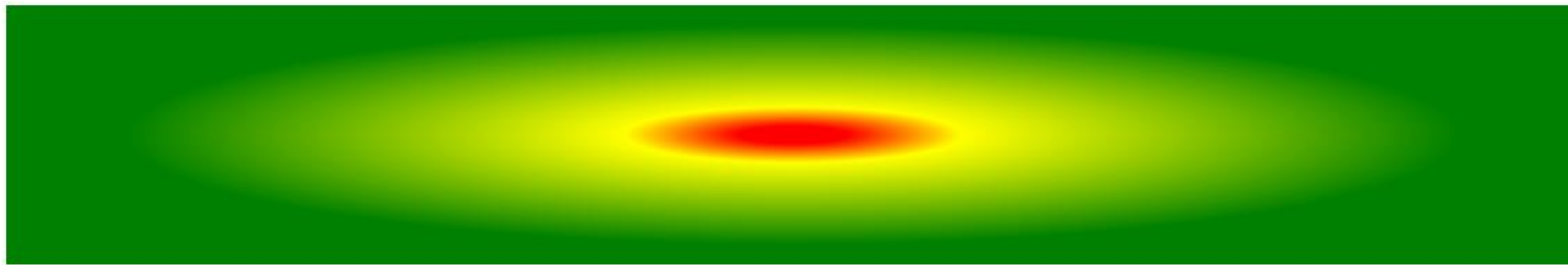
```
#grad1 {  
  background: radial-gradient(red, yellow, green);  
}
```



Radial Gradients - Differently Spaced Color Stops

- ▶ The following example shows a radial gradient with differently spaced color stops:

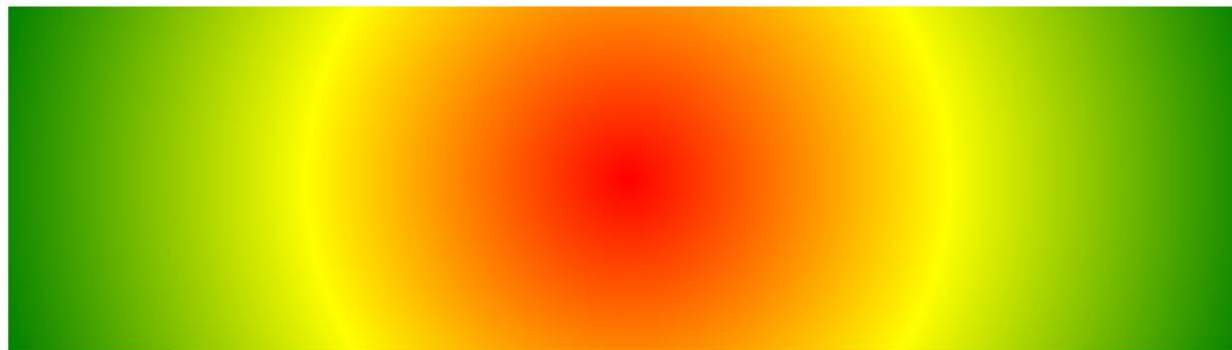
```
#grad2 {  
  background: radial-gradient(red 5%, yellow 15%, green 60%);  
}
```



Set Shape

- ▶ The shape parameter defines the shape. It can take the value circle or ellipse.
 - ▶ The default value is ellipse
- ▶ The following example shows a radial gradient with the shape of a circle:

```
#grad3 {  
  background: radial-gradient(circle, red, yellow, green);  
}
```



CSS Shadows

- ▶ With CSS you can add shadow to text and to elements
- ▶ The **text-shadow** property applies shadow to text
- ▶ In its simplest use, you only specify the horizontal shadow and the vertical shadow:

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

Text-shadow effect

- ▶ Next, add a color to the shadow:

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

Text-shadow effect

- ▶ Then, add a blur effect to the shadow:

```
h1 {  
  text-shadow: 2px 2px 5px red;  
}
```

Text-shadow effect

Multiple Shadows

- ▶ To add more than one shadow to the text, you can add a comma-separated list of shadows
- ▶ The following example shows a white text with black, blue, and darkblue shadow:

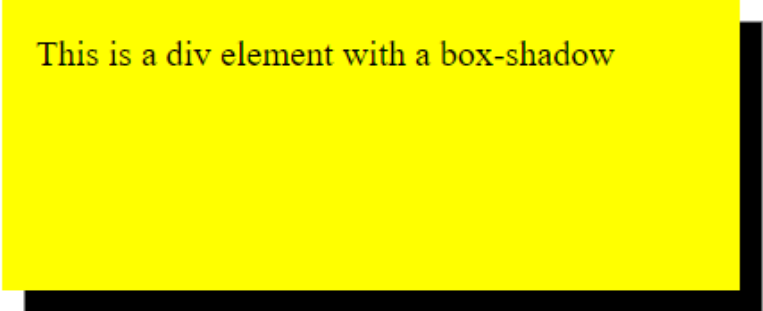
```
h1.multiple-shadows {  
  color: white;  
  text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;  
}
```

Text-shadow effect

Box Shadow

- ▶ The CSS **box-shadow** property applies shadow to elements
- ▶ In its simplest use, you only specify the horizontal shadow and the vertical shadow:

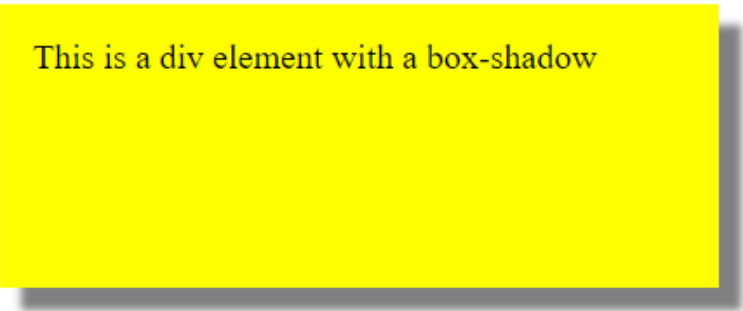
```
div {  
  width: 300px;  
  height: 100px;  
  padding: 15px;  
  background-color: yellow;  
  box-shadow: 10px 10px;  
}
```



This is a div element with a box-shadow

- ▶ You can also add a color and a blur effect to the shadow:

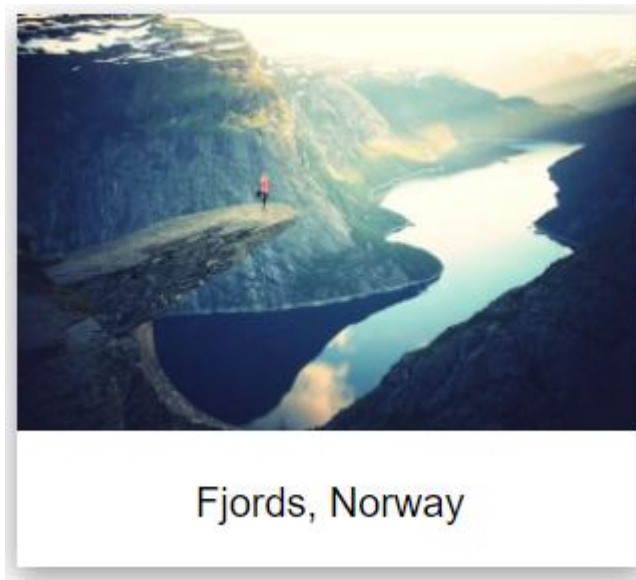
```
div {  
  width: 300px;  
  height: 100px;  
  padding: 15px;  
  background-color: yellow;  
  box-shadow: 10px 10px 5px grey;  
}
```



This is a div element with a box-shadow

Exercise (7)

- ▶ Use the box-shadow property to create a paper-like card:



Transforms

- ▶ CSS transforms allow you to translate, rotate, scale, and skew elements
- ▶ A transformation is an effect that lets an element change shape, size and position
- ▶ You can use one of the following methods for transformations:
 - ▶ `translate()`
 - ▶ `rotate()`
 - ▶ `scale()`
 - ▶ `skewX()`
 - ▶ `skewY()`
 - ▶ `skew()`
 - ▶ `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:

```
div {  
  width: 300px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
}  
  
div.translate {  
  transform: translate(50px, 100px);  
}
```

The translate() Method

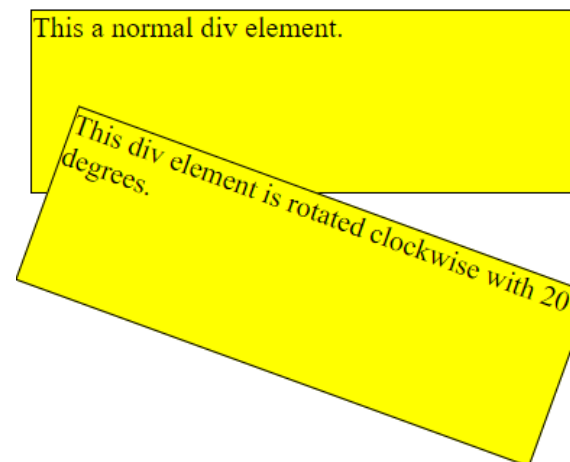
This div element is moved 50 pixels to the right, and 100 pixels down from its current position.

The rotate() Method

- ▶ The **rotate()** method rotates an element clockwise or counter-clockwise according to a given degree
 - ▶ Using negative values will rotate the element counter-clockwise
- ▶ The following example rotates the <div> element clockwise with 20 degrees:

```
div {  
  width: 300px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
}  
  
div.rotate {  
  transform: rotate(20deg);  
}
```

The rotate() Method

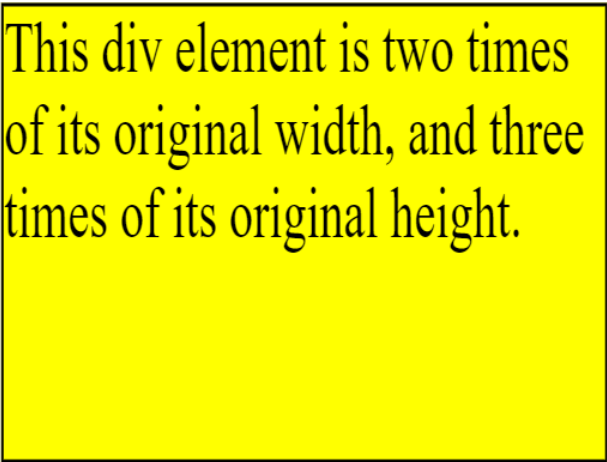


The scale() Method

- ▶ 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:

```
div {  
  margin: 150px;  
  width: 200px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
}  
  
div.scale {  
  transform: scale(2,3);  
}
```

The scale() Method



This div element is two times of its original width, and three times of its original height.

The skew() Method

- ▶ The **skew()** method skews an element along the X and Y-axis by the given angles.
- ▶ The following example skews the <div> element 20 degrees along the X-axis, and 10 degrees along the Y-axis:

```
div {  
  margin: 20px;  
  width: 300px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
}  
  
div.skew {  
  transform: skew(20deg,10deg);  
}
```

The skew() Method

This a normal div element.

This div element is skewed 20 degrees along the X-axis, and 10 degrees along the Y-axis.

Transitions

- ▶ CSS transitions allows you to change property values smoothly (from one value to another), over a given duration
- ▶ To create a transition effect, you must specify two things:
 - ▶ The CSS property you want to add an effect to
 - ▶ The duration of the effect
- ▶ The transition effect will start when the specified CSS property changes value
- ▶ The following <div> has a transition effect for the width property with duration of 2s:

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

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



Specify the Speed Curve of the Transition

- ▶ The **transition-timing-function** property specifies the speed curve of the transition
- ▶ It can have the following values:
 - ▶ ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
 - ▶ linear - specifies a transition effect with the same speed from start to end
 - ▶ ease-in - specifies a transition effect with a slow start
 - ▶ ease-out - specifies a transition effect with a slow end
 - ▶ ease-in-out - specifies a transition effect with a slow start and end
 - ▶ cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function

```
#div2 {  
  transition-timing-function: ease;  
}
```



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
- ▶ You can define the keyframes using the keywords “from” and “to” (which represent 0% (start) and 100% (complete)) or by using percentages
 - ▶ By using percentages, you can add as many style changes as you like

Animations

- ▶ The following example applies an animation to the <div> element, which lasts for 4 seconds, and gradually changes its background-color from red to green:

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

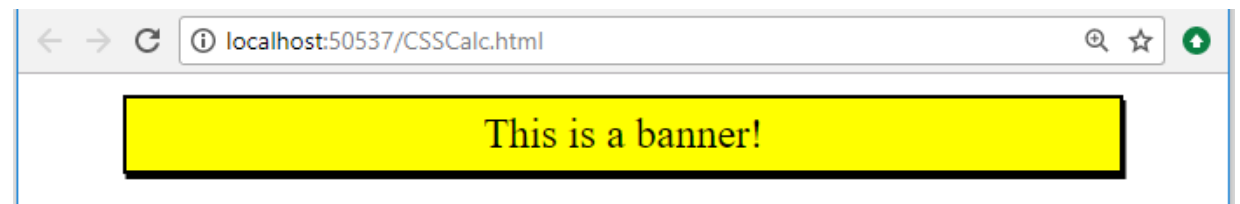
/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```



CSS calc()

- ▶ The **calc()** function lets you perform calculations when specifying CSS property values
- ▶ The operands in the expression may be any CSS units value
- ▶ You can use different units for each value in your expression, if you wish
- ▶ calc() makes it easy to position an object with a set margin.
- ▶ For example, the CSS creates a banner that stretches across the window, with a 40-pixel gap between both sides of the banner and the edges of the window:

```
.banner {  
  position: absolute;  
  left: calc(40px);  
  width: calc(100% - 80px);  
  border: solid black 1px;  
  box-shadow: 1px 2px;  
  background-color: yellow;  
  padding: 5px;  
  text-align: center;  
  box-sizing: border-box;  
}
```



Exercise (8)

- ▶ Using CSS create an animation that will move a div element from the left side of the page to its right side
- ▶ The duration of the animation should be 8 seconds
- ▶ After 2 seconds the div should get to the center of the screen

