

2016

CONTINUE TO DIGITOLIZE THE WORLD



CSS

Tumwesige k James

LWFG ltd

11/14/2016



Contents

CHAPTER 1.....	14
1. What is CSS?	14
CSS Demo - One HTML Page - Multiple Styles!.....	16
Welcome to My Homepage.....	17
No Styles	17
Why Use CSS?	17
CSS Solved a Big Problem.....	17
CSS Saves a Lot of Work!	18
CSS Syntax	18
The element Selector	19
The id Selector	19
The class Selector	19
Grouping Selectors.....	20
CSS Comments.....	20
CSS How to.....	21
Three Ways to Insert CSS	21
External Style Sheet	21
Internal Style Sheet.....	22
Inline Styles	22
Multiple Style Sheets	23
Multiple Style Sheets	23
Cascading Order.....	24
GB (Red, Green, Blue)	25
Hexadecimal Colors	25
2. CSS Backgrounds	26
Background Color	26
CSS Backgrounds	26
Background Color	27
Background Image	27
Background Image - Repeat Horizontally or Vertically.....	28
Background Image - Set position and no-repeat.....	28
Background Image - Fixed position.....	29
Background - Shorthand property.....	29

CSS Borders	30
CSS Border Properties	30
Border Style	30
Result:	30
Border Width	32
Border Color	32
Border - Individual Sides	33
Border - Individual Sides	33
Border - Shorthand Property	34
Rounded Borders	35
3. CSS Margins.....	36
CSS Margins.....	36
Margin - Individual Sides	36
Margin - Shorthand Property	37
The auto Value	38
The inherit Value	38
Margin Collapse	38
CSS Padding.....	39
CSS Padding.....	39
Padding - Individual Sides.....	39
Padding - Shorthand Property.....	39
Setting max-width.....	41
Setting height and width.....	42
CSS Box Model	42
The CSS Box Model.....	42
Width and Height of an Element.....	43
CSS Outline	44
CSS Outline	44
Outline Style.....	44
Outline Color	45
Outline Width	46
Outline - Shorthand property	46
4. CSS Text.....	47
TEXT FORMATTING	47
Text Color.....	47
Text Alignment	47

Text Decoration	48
Text Transformation	49
Text Indentation	49
Letter Spacing	49
Line Height	50
Text Direction	50
Word Spacing.....	50
CSS Fonts	50
Difference Between Serif and Sans-serif Fonts	51
CSS Font Families	51
Font Family.....	51
Font Style	52
Font Size	52
Set Font Size With Pixels	53
Set Font Size With Em	53
Use a Combination of Percent and Em.....	54
Font Weight	54
Font Variant.....	55
5. CSS Icons	55
How To Add Icons	55
Font Awesome Icons.....	55
Bootstrap Icons.....	56
Google Icons.....	56
CSS Links	57
Styling Links	57
Text Decoration.....	58
Background Color	58
Advanced - Link Buttons	59
6. CSS Lists	59
Coffee	59
HTML Lists and CSS List Properties	60
Different List Item Markers.....	60
An Image as The List Item Marker	60
Position The List Item Markers	61
List - Shorthand property	61
Styling List With Colors	61

7. CSS Tables	62
CSS Tables.....	62
Table Borders.....	63
Table Width and Height	64
Horizontal Alignment	64
Horizontal Alignment	64
Vertical Alignment.....	65
Table Padding	65
Horizontal Dividers	65
Hoverable Table.....	66
Striped Tables	66
Table Color.....	66
Responsive Table	67
EXAMPLE SOUCE OF THE TABBLE ABOVE.....	67
CSS Layout - The display Property.....	70
The display Property	70
Block-level Elements.....	71
Inline Elements.....	71
Display: none;	71
Override The Default Display Value	71
Hide an Element - display:none or visibility:hidden?	72
a. CSS Layout - width and max-width.....	73
Using width, max-width and margin: auto;	73
21 . CSS Layout - The position Property	75
The position Property	75
position: static;	75
position: relative;	75
position: fixed;	76
position: absolute;	76
8. Overlapping Elements.....	77
Positioning Text In an Image.....	77
CSS Layout - Overflow	78
Visible	78
edit	79
CSS Layout - float and clear.....	79
The float Property	80

The clear Property.....	80
CSS Layout - float and clear.....	80
The float Property	80
The clear Property.....	81
The clearfix Hack - overflow: auto;.....	81
Web Layout Example	81
Edit.....	82
Demo.....	84
nav.....	84
section	84
section	84
CSS Layout - inline-block	85
The inline-block Value	85
Demo.....	87
The New Way - using inline-block.....	87
CSS Layout - Horizontal & Vertical Align.....	87
Center Align Elements	87
Center Align Text	88
Center an Image.....	88
Left and Right Align - Using position	89
Left and Right Align - Using float.....	89
Center Vertically - Using padding.....	90
Center Vertically - Using line-height	91
Center Vertically - Using position & transform	91
Edit.....	92
Centering.....	93
CSS Combinators	93
Descendant Selector	94
Child Selector.....	94
Adjacent Sibling Selector	94
General Sibling Selector.....	95
edit	95
Demo.....	95
CSS Pseudo-classes	96
Syntax.....	96
Anchor Pseudo-classes	96

Pseudo-classes and CSS Classes	97
Hover on <div>	97
Simple Tooltip Hover.....	97
CSS - The :first-child Pseudo-class.....	98
Match the first <p> element	98
Match the first <i> element in all <p> elements	98
Match all <i> elements in all first child <p> elements.....	98
CSS - The :lang Pseudo-class	98
Edit.....	99
CSS Pseudo-elements	100
Syntax.....	100
The ::first-line Pseudo-element	100
The ::first-letter Pseudo-element	101
Pseudo-elements and CSS Classes.....	101
Multiple Pseudo-elements	102
CSS - The ::before Pseudo-element	102
CSS - The ::after Pseudo-element	102
CSS - The ::selection Pseudo-element	103
CSS Opacity / Transparency	103
Transparent Image	103
Transparent Hover Effect	104
Transparent Box	105
Transparency using RGBA.....	105
Text in Transparent Box	106
Edit.....	107
CSS Navigation Bar	108
Demo:.....	108
Navigation Bars.....	109
Navigation Bar = List of Links	109
Vertical Navigation Bar.....	110
Vertical Navigation Bar Examples	111
Active/Current Navigation Link	113
Center Links & Add Borders	113
Full-height Fixed Vertical Navbar.....	114
Horizontal Navigation Bar	114
Inline List Items.....	114

Example.....	114
Floating List Items	114
Horizontal Navigation Bar Examples	115
Example.....	115
Active/Current Navigation Link	117
Right-Align Links	117
Border Dividers	118
Fixed Navigation Bar.....	118
Fixed Top.....	118
Fixed Bottom.....	118
Gray Horizontal Navbar	119
Responsive Topnav	119
Responsive Sidenav.....	120
Dropdown Navbar	122
CSS Dropdowns	123
Demo: Dropdown Examples.....	123
Basic Dropdown.....	123
Dropdown Menu	124
Right-aligned Dropdown Content	125
Dropdown Image	126
Demo:.....	126
Basic Tooltip	126
Positioning Tooltips	127
Right Tooltip	128
Left Tooltip.....	128
Fade In Tooltips (Animation)	130
Image Gallery	131
Example.....	131
CSS Image Sprites	133
Image Sprites - Simple Example.....	134
Image Sprites - Create a Navigation List.....	134
Image Sprites - Hover Effect.....	135
CSS Attribute Selectors	136
Style HTML Elements.....	136
CSS [attribute] Selector	136
CSS [attribute="value"] Selector	137

CSS [attribute~="value"] Selector	137
CSS [attribute = "value"] Selector	137
CSS [attribute^="value"] Selector	138
CSS [attribute\$="value"] Selector	138
CSS [attribute*="value"] Selector	138
Styling Forms	138
Bottom of Form.....	139
Styling Input Fields.....	139
Padded Inputs.....	139
Bordered Inputs.....	140
Colored Inputs.....	140
Focused Inputs.....	141
Input with icon/image	141
Animated Search Input	142
Styling Textareas	142
Styling Select Menus	142
Styling Input Buttons.....	143
CSS Counters	143
Automatic Numbering With Counters	143
Nesting Counters.....	144
CSS Counter Properties.....	145
Chapter 2 vision three.....	145
CSS3 Introduction	145
CSS3 Rounded Corners	146
Browser Support	146
CSS3 Rounded Corners.....	146
CSS3 border-radius Property	146
CSS3 border-radius - Specify Each Corner	147
CSS3 Border Images	149
CSS3 border-image Property	149
CSS3 border-image - Different Slice Values	150
Edit.....	151
Demo	152
CSS3 Backgrounds.....	153
CSS3 Multiple Backgrounds.....	153
CSS3 Background Size	153

Lorem Ipsum Dolor.....	154
Lorem Ipsum Dolor.....	154
Define Sizes of Multiple Background Images	155
Full Size Background Image.....	155
CSS3 background-origin Property	155
CSS3 background-clip Property	156
Demo	158
Lorem Ipsum Dolor.....	158
CSS3 Colors	159
RGBA Colors	159
Lorem Ipsum Dolor.....	158
Lorem Ipsum Dolor.....	159
HSL Colors.....	160
HSLA Colors.....	160
Opacity	161
Edit	161
CSS3 Gradients	163
CSS3 Linear Gradients	163
Using Angles.....	164
Using Multiple Color Stops	164
Gradient Background.....	165
Using Transparency	165
Repeating a linear-gradient	166
CSS3 Radial Gradients	166
Set Shape.....	167
Use of Different Size Keywords.....	167
Repeating a radial-gradient	168
CSS3 Shadow Effects.....	170
CSS3 Shadow Effects.....	170
CSS3 Shadow Effects.....	170
CSS3 Shadow Effects.....	171
CSS3 Text Shadow.....	171
Text shadow effect!	171
Text shadow effect!	171
Text shadow effect!	171
Text shadow effect!	171

Text shadow effect!	172
Multiple Shadows.....	172
Text shadow effect!	172
Text shadow effect!	172
CSS3 box-shadow Property	172
Cards	174
CSS3 Text.....	174
CSS3 Text Overflow.....	175
CSS3 Word Wrapping	175
CSS3 Word Breaking.....	176
CSS3 Web Fonts	176
Different Font Formats	177
The Font You Want	177
Using Bold Text.....	178
CSS3 2D Transforms.....	178
The translate() Method	179
The rotate() Method.....	179
The scale() Method.....	180
The skewX() Method.....	180
The skewY() Method	181
The skew() Method	181
The matrix() Method.....	182
CSS3 3D Transforms.....	184
Browser Support for 3D Transforms	184
CSS3 3D Transforms	184
The rotateX() Method	184
The rotateY() Method.....	185
The rotateZ() Method.....	185
CSS3 Transitions.....	186
How to Use CSS3 Transitions?	187
Delay the Transition Effect	188
CSS3 Animations	189
The @keyframes Rule	189
Delay an Animation	191
Specify the Speed Curve of the Animation.....	193
Animation Shorthand Property	193

Demo	196
CSS Images	196
Image Text.....	199
CSS Buttons	200
Basic Button Styling	200
Button Colors	200
Button Sizes.....	201
Rounded Buttons	201
Colored Button Borders	201
Shadow Buttons	202
Disabled Buttons	202
Button Width.....	203
Button Groups.....	203
CSS3 Multi-column Layout.....	203
Specify How Many Columns an Element Should Span	205
CSS3 User Interface	206
Flex Direction.....	212
The justify-content Property	213
The align-items Property	215
The flex-wrap Property	216
The align-content Property	217 ¹
Flex Item Properties.....	218
CSS3 Media Queries.....	223
Width above 1151px - Add icon as we used before.....	229
Responsive Web Design - Introduction	230
Designing For The Best Experience For All Users	230
Responsive Web Design - The Viewport	231
Setting The Viewport.....	231
Size Content to the Viewport.....	233
What is a Grid-View?	233
Building a Responsive Grid-View	233
Responsive Web Design - Media Queries	236
Add a Breakpoint	236
Always Design for Mobile First.....	237
Another Breakpoint.....	238

Responsive Web Design - Images	244
Using The width Property	244
Using The max-width Property	245
Add an Image to The Example Web Page	245
Background Images.....	245
Different Images for Different Devices	246
HTML5 <picture> Element	248
Responsive Web Design - Videos.....	248
Using The max-width Property	249
Add a Video to the Example Web Page	249
Responsive Web Design - Frameworks	254
Using W3.CSS	254
W3.CSS Demo	255
CSS References.....	257

CHAPTER 1

1. What is CSS?

What is CSS?

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects

Advantages of CSS CSS saves time – You can write CSS once and then reuse same sheet in multiple HTML pages.

You can define a style for each HTML element and apply it to as many Web pages as you want. Pages load faster – If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times. Easy maintenance – To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.

Superior styles to HTML – CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.

Multiple Device Compatibility – Style sheets allow content to be optimized for more than one type of device.

By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.

Global web standards – Now HTML attributes are being deprecated and it is being recommended to use CSS.

So it's a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

Offline Browsing – CSS can store web applications locally with the help of an offline cache. Using this, we can view offline websites.

The cache also ensures faster loading and better overall performance of the website.

Platform Independence – The Script offer consistent platform independence and can support latest browsers as well.

Who Creates and Maintains CSS?

CSS was invited by Haakon Wium Lie on October 10, 1994 and maintained through a group of people within the W3C called the CSS Working Group.

The CSS Working Group creates documents called specifications. When a specification has been discussed and officially ratified by W3C members, it becomes a recommendation.

These ratified specifications are called recommendations because the W3C has no control over the actual implementation of the language. Independent companies and organizations create that software CSS stands for Cascading Style Sheets

CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**

CSS saves **a lot of work**. It can control the layout of multiple web pages all at once external style sheets are stored in **CSS files**.

CSS Versions

Cascading Style Sheets, level 1 (CSS1) was came out of W3C as a recommendation in December 1996.

This version describes the CSS language as well as a simple visual formatting model for all the HTML tags.

CSS2 was became a W3C recommendation in May 1998 and builds on CSS1. This version adds Support for media-specific style sheets e.g. printers and aural devices, downloadable fonts, element Positioning and tables.

CSS3 was became a W3C recommendation in June 1999 and builds on older versions CSS. it hasDivided into documentations is called as Modules and here each module having new extension featuresDefined in CSS2.

CSS3 Modules

CSS3 Modules are having old CSS specifications as well as extension features.

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

CSS Demo - One HTML Page - Multiple Styles!

Here we will show one HTML page displayed with four different stylesheets. Click on the "Stylesheet 1", "Stylesheet 2", "Stylesheet 3", "Stylesheet 4" links below to see the different styles:

```
<div class="container wrapper">
  <div id="top">
    <h1>Welcome to My Homepage</h1>
    <p>Use the menu to select different Stylesheets</p>
  </div>
  <div class="wrapper">
    <div id="menubar">
      <ul id="menulist">
        <li class="menuitem" onclick="reStyle(0)">Stylesheet 1
        </li><li class="menuitem" onclick="reStyle(1)">Stylesheet 2
        </li><li class="menuitem" onclick="reStyle(2)">Stylesheet 3
        </li><li class="menuitem" onclick="reStyle(3)">Stylesheet 4
        </li><li class="menuitem" onclick="noStyles()">No Stylesheet
        </li></ul>
      </div>
      <div id="main">
        <h1>Same Page Different Stylesheets</h1>
        <p>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:<br>
        <a href="#" onclick="reStyle(0);return false">Stylesheet1</a>,
        <a href="#" onclick="reStyle(1);return false">Stylesheet2</a>,
        <a href="#" onclick="reStyle(2);return false">Stylesheet3</a>,
        <a href="#" onclick="reStyle(3);return false">Stylesheet4</a>.
        </p>
        <h2>No Styles</h2>
        <p>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:<br><a href="#"
onclick="noStyles();return false">No Stylesheet</a>.</p>
      </div>
      <div id="sidebar">
        <h3>Side-Bar</h3>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy
nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
      </div>
    </div>

    <div id="bottom">
      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 feugait nulla facilisi.
    </div>
  </div>
```


Demo

Welcome to My Homepage

Use the menu to select different Stylesheets

Sheet 1

Sheet 1

Sheet 1

Sheet 1

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.

Sheet 1

Sheet 1

Sheet 1

Sheet 1

Side-Bar

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

Why Use CSS?

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

CSS Solved a Big Problem

HTML was NEVER intended to contain tags for formatting a web page!

HTML was created to **describe the content** of a web page, like:

```
<h1>This is a heading</h1>
```

`<p>This is a paragraph.</p>`

When tags like ``, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS removed the style formatting from the HTML page!

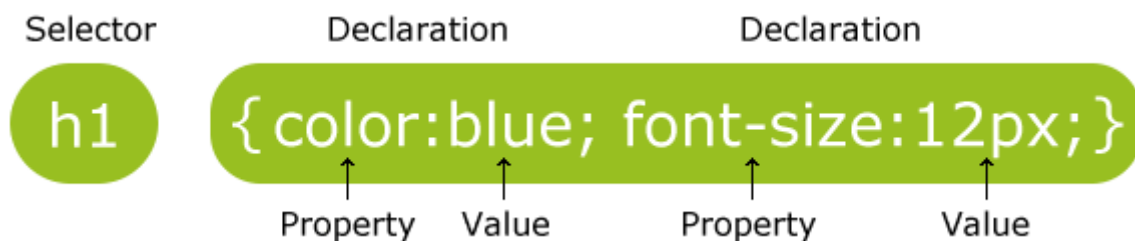
CSS Saves a Lot of Work!

The style definitions are normally saved in external .css files.

With an external stylesheet file, you can change the look of an entire website by changing just one file!

CSS Syntax

A CSS rule-set consists of a selector and a declaration block:



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

In the following example all `<p>` elements will be center-aligned, with a red text color:

Example

```
p {  
  color: red;  
  text-align: center;  
}
```

CSS Selectors

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

The element Selector

The element selector selects elements based on the element name.

You can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color):

Example

```
p {  
  text-align: center;  
  color: red;  
}
```

The id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element should be unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

The style rule below will be applied to the HTML element with id="para1":

Example

```
#para1 {  
  text-align: center;  
  color: red;  
}
```

Note: An id name cannot start with a number!

The class Selector

The class selector selects elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the name of the class.

In the example below, all HTML elements with class="center" will be red and center-aligned:

Example

```
.center {  
  text-align: center;  
  color: red;  
}
```

HTML elements can also refer to more than one class.

In the example below, the <p> element will be styled according to class="center" and to class="large":

Example

```
<p class="center large">This paragraph refers to two classes.</p>
```

Note: A class name cannot start with a number!

Grouping Selectors

If you have elements with the same style definitions, like this:

```
h1 {  
  text-align: center;  
  color: red;  
}
```

```
h2 {  
  text-align: center;  
  color: red;  
}
```

```
p {  
  text-align: center;  
  color: red; }
```

It will be better to group the selectors, to minimize the code.

To group selectors, separate each selector with a comma.

In the example below we have grouped the selectors from the code above:

Example

```
h1, h2, p {  
  text-align: center;  
  color: red;  
}
```

CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

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

Example

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

CSS How to...

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

External Style Sheet

With an external style sheet, you can change the look of an entire website by changing just one file!

Each page must include a reference to the external style sheet file inside the `<link>` element. The `<link>` element goes inside the `<head>` section:

Example

```
<head>  
<link rel="stylesheet" type="text/css" href="mystyle.css">  
</head>
```

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a `.css` extension.

Here is how the "myStyle.css" looks:

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

```
h1 {  
  color: navy;  
  margin-left: 20px;  
}
```

Note: Do not add a space between the property value and the unit (such as `margin-left: 20 px;`). The correct way is: `margin-left: 20`

Internal Style Sheet

An internal style sheet may be used if one single page has a unique style.

Internal styles are defined within the `<style>` element, inside the `<head>` section of an HTML page:
Example

```
<head>  
<style>  
body {  
  background-color: linen;  
}
```

```
h1 {  
  color: maroon;  
  margin-left: 40px;  
}  
</style>
```

`</head>`
Example

```
<head>  
<style>  
body {  
  background-color: linen;  
}
```

```
h1 {  
  color: maroon;  
  margin-left: 40px;  
}  
</style>  
</head>
```

Inline Styles

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

The example below shows how to change the color and the left margin of a `<h1>` element:

Example

```
<h1 style="color:blue;margin-left:30px;">This is a heading.</h1>
```

Tip: An inline style loses many of the advantages of a style sheet (by mixing content with presentation). Use this method sparingly

Multiple Style Sheets

If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

Example

Assume that an external style sheet has the following style for the <h1> element:

```
h1 {  
  color: navy;  
}
```

then, assume that an internal style sheet also has the following style for the <h1> element:

```
h1 {  
  color: orange;  
}
```

Multiple Style Sheets

If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

Example

Assume that an external style sheet has the following style for the <h1> element:

```
h1 {  
  color: navy;  
}
```

then, assume that an internal style sheet also has the following style for the <h1> element:

```
h1 {  
  color: orange;  
}
```

If the internal style is defined after the link to the external style sheet, the <h1> elements will be "orange":

Example

```

<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<style>
h1 {
  color: orange;
}
</style>
</head>

```

However, if the internal style is defined before the link to the external style sheet, the <h1> elements will be "navy"

Example

```

<head>
<style>
h1 {
  color: orange;
}
</style>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>

```








Cascading Order

What style will be used when there is more than one style specified for an HTML element?

Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style (inside a specific HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or a browser default value.

Color	Name
	Red
	Green
	Blue
	Orange
	Yellow
	Cyan
	Black

Note: Color names are case-insensitive: "Red" is the same as "red" or "RED".













HTML and CSS supports [140 standard color names](#).

GB (Red, Green, Blue)

RGB color values can be specified using this formula: `rgb(red, green, blue)`.

Each parameter (red, green, blue) defines the intensity of the color between 0 and 255.







For example, `rgb(255, 0, 0)` is displayed as red, because red is set to its highest value (255) and the others are set to 0. Experiment by mixing the RGB values below:

Red	Green	Blue
255	0	0
		
Color	RGB	
	<code>rgb(255,0,0)</code>	
	<code>rgb(0,255,0)</code>	
	<code>rgb(0,0,255)</code>	
	<code>rgb(255,165,0)</code>	
	<code>rgb(255,255,0)</code>	
	<code>rgb(0,255,255)</code>	
Color	RGB	
	<code>rgb(0,0,0)</code>	
	<code>rgb(128,128,128)</code>	
	<code>rgb(255,255,255)</code>	

Hexadecimal Colors

RGB values can also be specified using **hexadecimal** color values in the form: `#RRGGBB`, where RR (red), GG (green) and BB (blue) are hexadecimal values between 00 and FF (same as decimal 0-255).

For example, `#FF0000` is displayed as red, because red is set to its highest value (FF) and the others are set to the lowest value (00). **Note:** HEX values are case-insensitive: `"#ff0000"` is the same as `"#FF0000"`.

Color	HEX
	<code>#FF0000</code>
	<code>#00FF00</code>
	<code>#0000FF</code>
	<code>#FFA500</code>
	<code>#FFFF00</code>
	<code>#00FFFF</code>

Shades of grey are often defined using equal values for all the 3 light sources:

Example

Color	HEX
	#000000
	#808080
	#FFFFFF

2. CSS Backgrounds

The 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 Color

The `background-color` property specifies the background color of an element.

The background color of a page is set like this:

Example

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

With CSS, 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)"

CSS Backgrounds

The 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 Color

The `background-color` property specifies the background color of an element.

The background color of a page is set like this:

Example

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

With CSS, 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)"

Look at CSS Color Values for a complete list of possible color values.

In the example below, the `<h1>`, `<p>`, and `<div>` elements have different background colors:

Example

```
h1 {  
  background-color: green;  
}  
  
div {  
  background-color: lightblue;  
}  
  
p {  
  background-color: yellow;  
}
```

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.

The background image for a page can be set like this:

Example

```
body {  
  background-image: url("paper.gif");  
}
```

Below is an example of a **bad** combination of text and background image. The text is hardly readable:

Example

```
body {  
  background-image: url("bgdesert.jpg");  
}
```

Note: When using a background image, use an image that does not disturb the text.

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, like this:

Example

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

If the image above is repeated only horizontally (`background-repeat: repeat-x;`), the background will look better:

Example

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

Tip: To repeat an image vertically, set `background-repeat: repeat-y;`

Background Image - Set position and no-repeat

Showing the background image only once is also specified by the `background-repeat` property:

Example

```
body {  
  background-image: url("img_tree.png");  
  background-repeat: no-repeat;  
}
```

In the example above, the background image is shown in the same place as the text. We want to change the position of the image, so that it does not disturb the text too much.

The position of the image is specified by the `background-position` property:

Example

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

Background Image - Fixed position

To specify that the background image should be fixed (will not scroll with the rest of the page), use the `background-attachment` property:

Example

```
body {  
  background-image: url("img_tree.png");  
  background-repeat: no-repeat;  
  background-position: right top;  
  background-attachment: fixed;  
}
```

Background - Shorthand property

To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property.

The shorthand property for background is `background`:

Example

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

When using the shorthand property the order of the property values is:

When using the shorthand property the order of the property values is:

- `background-color`
- `background-image`
- `background-repeat`
- `background-attachment`
- `background-position`

It does not matter if one of the property values is missing, as long as the other ones are in this order.

CSS Borders

CSS Border Properties

The CSS `border` properties allow you to specify the style, width, and color of an element's border.

have borders on all sides. I have a red bottom border I have rounded borders. I have a blue left border.

Border Style

The `border-style` property specifies what kind of border to display.

The following values are allowed:

- `dotted` - Defines a dotted border
- `dashed` - Defines a dashed border
- `solid` - Defines a solid border
- `double` - Defines a double border
- `groove` - Defines a 3D grooved border. The effect depends on the border-color value
- `ridge` - Defines a 3D ridged border. The effect depends on the border-color value
- `inset` - Defines a 3D inset border. The effect depends on the border-color value
- `outset` - Defines a 3D outset border. The effect depends on the border-color value
- `none` - Defines no border
- `hidden` - Defines a hidden border

The `border-style` property can have from one to four values (for the top border, right border, bottom border, and the left

Example

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
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.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

Result:

.....
A dotted border.
.....

A dashed border.

—————
A solid border.
—————

A double border.

A groove border. The effect depends on the border-color value.

A ridge border. The effect depends on the border-color value.

An inset border. The effect depends on the border-color value.

An outset border. The effect depends on the border-color value.

No border.

A hidden border.

A mixed border.

Result:

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border. The effect depends on the border-color value.

A ridge border. The effect depends on the border-color value.

An inset border. The effect depends on the border-color value.

An outset border. The effect depends on the border-color value.

No border.

A hidden border.

A mixed border.

Note: None of the OTHER CSS border properties described below will have ANY effect unless the border-style property is set!

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 border, right border, bottom border, and the left border).

5px border-width

Example

```
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 color can be set by:

- name - specify a color name, like "red"
- Hex - specify a hex value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- transparent

The `border-color` property can have from one to four values (for the top border, right border, bottom border, and the left border).

If `border-color` is not set, it inherits the color of the element.

Red border

Example


```

p.one {
  border-style: solid;
  border-color: red;
}

p.two {
  border-style: solid;
  border-color: green;
}

p.three {
  border-style: solid;
  border-color: red green blue yellow;
}

```

Border - Individual Sides

From the examples above you have seen that it is possible to specify a different border for each side.

In CSS, there is also properties for specifying each of the borders (top, right, bottom, and left):

Different Border Styles

Example

```

p {
  border-top-style: dotted;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: solid;
}

```

The example above gives the same result as this:

Example

```

p {
  border-style: dotted solid;
}

```

Border - Individual Sides

From the examples above you have seen that it is possible to specify a different border for each side.

In CSS, there is also properties for specifying each of the borders (top, right, bottom, and left):

Different Border Styles

Example

```
p {  
  border-top-style: dotted;  
  border-right-style: solid;  
  border-bottom-style: dotted;  
  border-left-style: solid;  
}
```

The example above gives the same result as this:

Example

```
p {  
  border-style: dotted solid;  
}
```

So, here is how it works:

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 three values:

- **border-style: dotted solid double;**
 - top border is dotted
 - right and left borders are solid
 - bottom border is double

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 `border-style` property is used in the example above. However, it also works with `border-width` and `border-color`.

Border - Shorthand Property

As you can see from the examples above, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

The `border` property is a shorthand property for the following individual border properties:

- `border-width`
- `border-style` (required)
- `border-color`

Example

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

Result:

Some text

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

Left Border

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

Result:

Some text

Bottom Border

```
p {  
  border-bottom: 6px solid red;  
  background-color: lightgrey;  
}
```

Result:

Some text

Rounded Borders

The `border-radius` property is used to add rounded borders to an element:

Normal border

Round border

Rounder border

Roudest border

Example

```
p {  
  border: 2px solid red;  
  border-radius: 5px;  
}
```

Note: The `border-radius` property is not supported in IE8 and earlier versions.

3. CSS Margins

This element has a margin of 70px.

CSS Margins

The CSS `margin` properties are used to generate space around elements.

The `margin` properties set the size of the white space outside the border.

With CSS, you have full control over the margins. There are CSS properties for setting the margin for each side of an element (top, right, bottom, and left)

Margin - Individual Sides

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.

The following example sets different margins for all four sides of a `<p>` element:

Example

```
p {  
  margin-top: 100px;
```

```
margin-bottom: 100px;
margin-right: 150px;
margin-left: 80px;
}
```

Margin - Shorthand Property

To shorten the code, it is possible to specify all the margin properties in one property.

The `margin` property is a shorthand property for the following individual margin properties:

- `margin-top`
- `margin-right`
- `margin-bottom`
- `margin-left`

Example

```
p {
  margin: 100px 150px 100px 80px;
}
```

So, here is how it works:

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 three values:

- **`margin: 25px 50px 75px;`**
 - top margin is 25px
 - right and left margins are 50px
 - bottom margin is 75px

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

The auto Value

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:

Example

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

The inherit Value

This example lets the left margin be inherited from the parent element:

Example

```
div.container {  
  border: 1px solid red;  
  margin-left: 100px;  
}  
  
p.one {  
  margin-left: inherit;  
}
```

Margin Collapse

Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.

This does not happen on left and right margins! Only top and bottom margins!

Look at the following example:

Example

```
h1 {  
  margin: 0 0 50px 0;  
}  
  
h2 {  
  margin: 20px 0 0 0;  
}
```

In the example above, the `<h1>` element has a bottom margin of 50px. 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 70px (50px + 20px). But due to margin collapse, the actual margin ends up being 50px.

CSS Padding

This element has a padding of 70px.

CSS Padding

The CSS `padding` properties are used to generate space around content.

The padding clears an area around the content (inside the border) of an element.

With CSS, you have full control over the padding. There are CSS properties for setting the padding for each side of an element (top, right, bottom, and left).

Padding - Individual Sides

CSS has properties for specifying the padding for each side of an element:

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

All the padding 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

The following example sets different padding for all four sides of a `<p>` element:

Example

```
p {  
  padding-top: 50px;  
  padding-right: 30px;  
  padding-bottom: 50px;  
  padding-left: 80px;  
}
```

Padding - Shorthand Property

To shorten the code, it is possible to specify all the padding properties in one property.

The `padding` property is a shorthand property for the following individual padding properties:

- padding-top
- padding-right
- padding-bottom
- padding-left

Example

```
p {
  padding: 50px 30px 50px 80px;
}
```

So, here is how it works:

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 three values:

- **padding: 25px 50px 75px;**
 - top padding is 25px
 - right and left paddings are 50px
 - bottom padding is 75px

If the `padding` property has two values:

- **padding: 25px 50px;**
 - top and bottom paddings are 25px
 - right and left paddings are 50px

If the `padding` property has one value:

- **padding: 25px;**
 - all four paddings are 25px

Example

```
div.ex1 {
  padding: 25px 50px 75px 100px;
}
```

```
div.ex2 {
  padding: 25px 50px 75px;
}
```

```
div.ex3 {
  padding: 25px 50px;
```



```
}
```

```
div.ex4 {  
  padding: 25px;  
}
```

This element has a height of 100 pixels and a width of 500 pixels.

Example

```
div {  
  height: 100px;  
  width: 500px;  
  background-color: powderblue;  
}
```

Note: The `height` and `width` properties do not include padding, borders, or margins; they set the height/width of the area inside the padding, border, and margin of the element!

Setting max-width

The `max-width` property is used to set the maximum width of an element.

The `max-width` can be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).

The problem with the `<div>` above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.

Using `max-width` instead, in this situation, will improve the browser's handling of small windows.

Tip: Drag the browser window to smaller than 500px wide, to see the difference between the two divs!

This element has a height of 100 pixels and a max-width of 500 pixels.

Note: The value of the `max-width` property overrides `width`.

The following example shows a `<div>` element with a height of 100 pixels and a max-width of 500 pixels:

Example

```
div {  
  max-width: 500px;  
  height: 100px;  
  background-color: powderblue;  
}
```

This element has a width of 100%.

Setting height and width

The `height` and `width` properties are used to set the height and width of an element.

The `height` and `width` can be set to `auto` (this is default. Means that the browser calculates the height and width), or be specified in *length values*, like `px`, `cm`, etc., or in percent (%) of the containing block.

This element has a height of 200 pixels and a width of 50%

Example

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

CSS Box Model

The CSS Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:

- Explanation of the different parts: **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



The box model allows us to add a border around elements, and to define space between elements.

Example

```
div {  
  width: 300px;
```

```
border: 25px solid green;
padding: 25px;
margin: 25px;
}
```

Width and Height of an Element

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

Important: When you set the width and height properties of an element with CSS, you just set the width and height of the **content area**. To calculate the full size of an element, you must also add padding, borders and margins.

Assume we want to style a <div> element to have a total width of 350px:

Example

```
div {
  width: 320px;
  padding: 10px;
  border: 5px solid gray;
  margin: 0;
```

Here is the math:

```
320px (width)
+ 20px (left + right padding)
+ 10px (left + right border)
+ 0px (left + right margin)
= 350px
```

The total width of an element should be calculated like this:

Total element width = width + left padding + right padding + left border + right border + left margin + right margin

The total height of an element should be calculated like this:

Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

Note for old IE: Internet Explorer 8 and earlier versions, include padding and border in the width property. To fix this problem, add a <!DOCTYPE html> to the HTML page.

CSS Outline

CSS Outline

The CSS `outline` properties specify the style, color, and width of an outline.

An outline is a line that is drawn around elements (outside the borders) to make the element "stand out".

However, the outline property is different from the border property - The outline is NOT a part of an element's dimensions; the element's total width and height is not affected by the width of the outline.

This element has a thin black border and a double outline that is 10px wide and green.

Outline Style

The `outline-style` property specifies the style of the outline.

The `outline-style` property can have one of the following values:

- `dotted` - Defines a dotted outline
- `dashed` - Defines a dashed outline
- `solid` - Defines a solid outline
- `double` - Defines a double outline
- `groove` - Defines a 3D grooved outline. The effect depends on the `outline-color` value
- `ridge` - Defines a 3D ridged outline. The effect depends on the `outline-color` value
- `inset` - Defines a 3D inset outline. The effect depends on the `outline-color` value
- `outset` - Defines a 3D outset outline. The effect depends on the `outline-color` value
- `none` - Defines no outline
- `hidden` - Defines a hidden outline

The following example first sets a thin black border around each `<p>` element, then it shows the different `outline-style` values:

Example

```
p {  
  border: 1px solid black;  
  outline-color: red;  
}  
  
p.dotted {outline-style: dotted;}  
p.dashed {outline-style: dashed;}  
p.solid {outline-style: solid;}  
p.double {outline-style: double;}  
p.groove {outline-style: groove;}  
p.ridge {outline-style: ridge;}  
p.inset {outline-style: inset;}  
p.outset {outline-style: outset;}
```

Result:

A dotted outline.....	<input type="text"/>
A dashed outline.....	<input type="text"/>
A solid outline.....	<input type="text"/>
A double outline.....	<input type="text"/>
A groove outline. The effect depends on the outline-color value	<input type="text" value="red"/>
A ridge outline. The effect depends on the outline-color value.	
An inset outline. The effect depends on the outline-color value.	
An outset outline. The effect depends on the outline-color value	<input type="text"/>

Note: None of the OTHER CSS outline properties described below will have ANY effect unless the `outline-style` property is set!

Outline Color

The `outline-color` property is used to set the color of the outline.

The color can be set by:

- name - specify a color name, like "red"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- Hex - specify a hex value, like "#ff0000"
- invert - performs a color inversion (which ensures that the outline is visible, regardless of color background)

Example

```
p {  
  border: 1px solid black;  
  outline-style: double;  
  outline-color: red;  
}
```

Result:

A colored outline.

Outline Width

The `outline-width` property specifies the width of the outline.

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.

Example

```
p {border: 1px solid black;}
```

```
p.one {  
  outline-style: double;  
  outline-color: red;  
  outline-width: thick;  
}
```

```
p.two {  
  outline-style: double;  
  outline-color: green;  
  outline-width: 3px;  
}
```

Result:

A thick outline.
A thinner outline.

Outline - Shorthand property

To shorten the code, it is also possible to specify all the individual outline properties in one property.

The `outline` property is a shorthand property for the following individual outline properties:

- `outline-width`
- `outline-style` (required)
- `outline-color`

Example

```
p {  
  border: 1px solid black;  
  outline: 5px dotted red;  
}
```

Result:

An outline.

4. CSS Text

TEXT FORMATTING

This text is styled with some of the text formatting properties. The heading uses the `text-align`, `text-transform`, and `color` properties. The paragraph is indented, aligned, and the space between characters is specified. The underline is removed from this colored

Text Color

The `color` property is used to set the color of the text.

With CSS, a color is most often specified by:

- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

Look at [CSS Color Values](#) for a complete list of possible color values.

The default text color for a page is defined in the body selector.

Example

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

Note: For W3C compliant CSS: If you define the `color` property, you must also define the `background-color`

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.

The following example shows center aligned, and left and right aligned text (left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left):

Example

```
h1 {  
  text-align: center;  
}
```

```
h2 {  
  text-align: left;  
}
```

```
h3 {  
  text-align: right;  
}
```

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):

Example

```
div {  
  text-align: justify;  
}
```

Text Decoration

The `text-decoration` property is used to set or remove decorations from text.

The value `text-decoration: none;` is often used to remove underlines from links:

Example

```
a {  
  text-decoration: none;  
}
```

The other `text-decoration` values are used to decorate text:

Example

```
h1 {  
  text-decoration: overline;  
}
```

```
h2 {  
  text-decoration: line-through;  
}
```

```
h3 {  
  text-decoration: underline;  
}
```


Note: It is not recommended to underline text that is not a link, as this often confuses the reader.

Text Transformation

The `text-transform` property is used to specify uppercase and lowercase letters in a text.

It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

Example

```
p.uppercase {  
  text-transform: uppercase;  
}
```

```
p.lowercase {  
  text-transform: lowercase;  
}
```

```
p.capitalize {  
  text-transform: capitalize;  
}
```

Text Indentation

The `text-indent` property is used to specify the indentation of the first line of a text:

Example

```
p {  
  text-indent: 50px;  
}
```

Letter Spacing

The `letter-spacing` property is used to specify the space between the characters in a text.

The following example demonstrates how to increase or decrease the space between characters:

Example

```
h1 {  
  letter-spacing: 3px;  
}
```

```
h2 {  
  letter-spacing: -3px;  
}
```

Line Height

The `line-height` property is used to specify the space between lines:

Example

```
p.small {  
  line-height: 0.8;  
}
```

```
p.big {  
  line-height: 1.8;  
}
```

Text Direction

The `direction` property is used to change the text direction of an element:

Example

```
div {  
  direction: rtl;  
}
```

Word Spacing

The `word-spacing` property is used to specify the space between the words in a text.

The following example demonstrates how to increase or decrease the space between words:

Example

```
h1 {  
  word-spacing: 10px;  
}
```

```
h2 {  
  word-spacing: -5px;  
}
```

how to set the vertical align of an image in a text.

CSS Fonts

The CSS font properties define the font family, boldness, size, and the style of a text.

Difference Between Serif and Sans-serif Fonts



CSS Font Families

In CSS, 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 Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

Note: 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".

More than one font family is specified in a comma-separated list:

Example

```
p {  
  font-family: "Times New Roman", Times, serif;  
}
```

For commonly used font combinations, look at our [Web Safe Font Combinations](#).

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)

Example

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

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

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

Font Size

The `font-size` property sets the size of the text.

Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.

Always use the proper HTML tags, like `<h1>` - `<h6>` for headings and `<p>` for paragraphs.

The font-size value can be an absolute, or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Note: If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

Set Font Size With Pixels

Setting the text size with pixels gives you full control over the text size:

Example

```
h1 {  
  font-size: 40px;  
}
```

```
h2 {  
  font-size: 30px;  
}
```

```
p {  
  font-size: 14px;  
}
```

ip: If you use pixels, you can still use the zoom tool to resize the entire page.

Set Font Size With Em

To allow users to resize the text (in the browser menu), many developers use em instead of pixels.

The em size unit is recommended by the W3C.

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: $pixels/16=em$

Example

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

```
h2 {  
  font-size: 1.875em; /* 30px/16=1.875em */  
}
```

```
p {
```

```
font-size: 0.875em; /* 14px/16=0.875em */
}
```

In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers.

Unfortunately, there is still a problem with older versions of IE. The text becomes larger than it should when made larger, and smaller than it should when made smaller.

Use a Combination of Percent and Em

The solution that works in all browsers, is to set a default font-size in percent for the <body> element:

Example

```
body {
  font-size: 100%;
}

h1 {
  font-size: 2.5em;
}

h2 {
  font-size: 1.875em;
}

p {
  font-size: 0.875em;
}
```

Our code now works great! It shows the same text size in all browsers, and allows all browsers to zoom or resize the text!

Font Weight

The `font-weight` property specifies the weight of a font:

Example

```
p.normal {
  font-weight: normal;
}

p.thick {
  font-weight: bold;
}
```

Font Variant

The `font-variant` property specifies whether or not a text should be displayed in a small-caps font.

In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

Example

```
p.normal {  
    font-variant: normal;  
}  
  
p.small {  
    font-variant: small-caps;  
}
```

How To Add Icons

The simplest way to add an icon to your HTML page, is with an icon library, such as Font Awesome.

Add the name of the specified icon class to any inline HTML element (like `<i>` or ``).

All the icons in the icon libraries below, are scalable vectors that can be customized with CSS (size, color, shadow, etc.)

Font Awesome Icons

To use the Font Awesome icons, add the following line inside the `<head>` section of your HTML page:

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.6.3/css/font-awesome.min.css">
```

No downloading or installation is required!

Example

```
<!DOCTYPE html>  
<html>  
<head>  
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.6.3/css/font-awesome.min.css">  
</head>  
<body>
```

```

<i class="fa fa-cloud"></i> it will show cloud icon/imag respectively
<i class="fa fa-heart"></i>
<i class="fa fa-car"></i>
<i class="fa fa-file"></i>
<i class="fa fa-bars"></i>

</body>
</html>

```

Bootstrap Icons

To use the Bootstrap glyphs, add the following line inside the <head> section of your HTML page:

```

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

```

Note: No downloading or installation is required!

Example

```

<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body>

<i class="glyphicon glyphicon-cloud"></i>
<i class="glyphicon glyphicon-remove"></i>
<i class="glyphicon glyphicon-user"></i>
<i class="glyphicon glyphicon-envelope"></i>
<i class="glyphicon glyphicon-thumbs-up"></i>

</body>
</html>

```

Result:

Google Icons

To use the Google icons, add the following line inside the <head> section of your HTML page:

```

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

```

Note: No downloading or installation is required!

Example

```

<!DOCTYPE html>
<html>

```



```

<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>
<body>

<i class="material-icons">cloud</i>
<i class="material-icons">favorite</i>
<i class="material-icons">attachment</i>
<i class="material-icons">computer</i>
<i class="material-icons">traffic</i>

</body>
</html>

```

Result:

Small image here fail to show but consinder the inages of google icona o text icon

CSS Links

With CSS, links can be styled in different ways.

[Text Link](#) [Text Link](#) [Link Button](#) [Link Button](#)

Styling Links

Links can be styled with any CSS property (e.g. color, font-family, background, etc.).

Example

```

a {
  color: hotpink;
}

```

In addition, links can be styled differently depending on what **state** they are in.

The four links states are:

- `a:link` - a normal, unvisited link
- `a:visited` - a link the user has visited
- `a:hover` - a link when the user mouses over it
- `a:active` - a link the moment it is clicked

Example

```

/* 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;
}
```

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

Text Decoration

The `text-decoration` property is mostly used to remove underlines from links:

Example

```
a:link {
  text-decoration: none;
}

a:visited {
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}

a:active {
  text-decoration: underline;
}
```

Background Color

The `background-color` property can be used to specify a background color for links:

Example

```

a:link {
  background-color: yellow;
}

a:visited {
  background-color: cyan;
}

a:hover {
  background-color: lightgreen;
}

a:active {
  background-color: hotpink;
}

```

Advanced - Link Buttons

This example demonstrates a more advanced example where we combine several CSS properties to display links as boxes/buttons:

Example

```

a:link, a:visited {
  background-color: #f44336;
  color: white;
  padding: 14px 25px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
}

a:hover, a:active {
  background-color: red;
}

```

6. CSS Lists

Coffee

1. Tea
 2. Coca Cola
- Coffee
 - Tea
 - Coca Cola

HTML Lists and CSS List Properties

In HTML, there are two main types of lists:

- unordered lists () - the list items are marked with bullets
- ordered lists () - the list items are marked with numbers or letters

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker
- Add background colors to lists and list items

Different List Item Markers

The `list-style-type` property specifies the type of list item marker.

The following example shows some of the available list item markers:

Example

```
ul.a {  
  list-style-type: circle;  
}  
  
ul.b {  
  list-style-type: square;  
}  
  
ol.c {  
  list-style-type: upper-roman;  
}  
  
ol.d {  
  list-style-type: lower-alpha;  
}
```

Note: Some of the values are for unordered lists, and some for ordered lists.

An Image as The List Item Marker

The `list-style-image` property specifies an image as the list item marker:

Example

```
ul {  
  list-style-image: url('sqpurple.gif');  
}
```

Position The List Item Markers

The `list-style-position` property specifies whether the list-item markers should appear inside or outside the content flow:

Example

```
ul {  
  list-style-position: inside;  
}
```

List - Shorthand property

The `list-style` property is a shorthand property. It is used to set all the list properties in one declaration:

Example

```
ul {  
  list-style: square inside url("sqpurple.gif");  
}
```

When using the shorthand property, the order of the property values are:

- `list-style-type` (if a `list-style-image` is specified, the value of this property will be displayed if the image for some reason cannot be displayed)
- `list-style-position` (specifies whether the list-item markers should appear inside or outside the content flow)
- `list-style-image` (specifies an image as the list item marker)

If one of the property values above are missing, the default value for the missing property will be inserted, if any.

Styling List With Colors

We can also style lists with colors, to make them look a little more interesting.

Anything added to the `` or `` tag, affects the entire list, while properties added to the `` tag will affect the individual list items:

Example

```
ol {  
  background: #ff9999;  
  padding: 20px;  
}  
  
ul {  
  background: #3399ff;  
  padding: 20px;  
}
```

```
ol li {
  background: #ffe5e5;
  padding: 5px;
  margin-left: 35px;
}
```

```
ul li {
  background: #cce5ff;
  margin: 5px;
}
```

Result:



- Coffee
- Tea
- Coca Cola

Coffee
Tea
Coca cola

7. CSS Tables

CSS Tables

The look of an HTML table can be greatly improved with CSS:

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Berglunds snabbköp	Christina Berglund	Sweden
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Königlich Essen	Philip Cramer	Germany
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy
:		

Table Borders

To specify table borders in CSS, use the `border` property.

The example below specifies a black border for `<table>`, `<th>`, and `<td>` elements:

Firstname	Lastname
Peter	Griffin
Lois	Griffin

Example

```
table, th, td {  
  border: 1px solid black;  
}
```

Notice that the table in the example above has double borders. This is because both the `<table>` and the `<th>` and `<td>` elements have separate borders.

Collapse Table Borders

The `border-collapse` property sets whether the table borders should be collapsed into a single border:

Example

```
table {  
  border-collapse: collapse;  
}  
  
table, th, td {  
  border: 1px solid black;  
}
```

If you only want a border around the table, only specify the `border` property for `<table>`:

Firstname	Lastname
Peter	Griffin
Lois	Griffin

Example

```
table {  
  border: 1px solid black;  
}
```

Table Width and Height

Width and height of a table are defined by the `width` and `height` properties.

The example below sets the width of the table to 100%, and the height of the `<th>` elements to 50px:

Firstname	Lastname	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

Example

```
table {  
  width: 100%;  
}
```

```
th {  
  height: 50px;  
}
```

Horizontal Alignment

The `text-align` property sets the horizontal alignment (like left, right, or center) of the content in `<th>` or `<td>`.

By default, the content of `<th>` elements are center-aligned and the content of `<td>` elements are left-aligned.

The following example left-aligns the text in `<th>` elements:

Firstname	Lastname	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

Horizontal Alignment

The `text-align` property sets the horizontal alignment (like left, right, or center) of the content in `<th>` or `<td>`.

By default, the content of `<th>` elements are center-aligned and the content of `<td>` elements are left-aligned.

The following example left-aligns the text in `<th>` elements:

Example

```
th {  
  text-align: left;  
}
```


Vertical Alignment

The `vertical-align` property sets the vertical alignment (like top, bottom, or middle) of the content in `<th>` or `<td>`.

By default, the vertical alignment of the content in a table is middle (for both `<th>` and `<td>` elements).

The following example sets the vertical text alignment to bottom for `<td>` elements:

Firstname	Lastname	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

Example

```
td {  
    height: 50px;  
    vertical-align: bottom;  
}
```

Table Padding

To control the space between the border and the content in a table, use the `padding` property on `<td>` and `<th>` elements:

Firstname	Lastname	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

Example

```
th, td {  
    padding: 15px;  
    text-align: left;  
}
```

Horizontal Dividers

First Name	Last Name	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

Hoverable Table

Use the `:hover` selector on `<tr>` to highlight table rows on mouse over:

First Name	Last Name	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

Example

```
tr:hover {background-color: #f5f5f5}
```

Striped Tables

First Name	Last Name	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe		

For zebra-striped tables, use the `nth-child()` selector and add a `background-color` to all even (or odd) table rows

Example

```
tr:nth-child(even) {background-color: #f2f2f2}
```

Table Color

The example below specifies the background color and text color of `<th>` elements

First Name	Last Name	Savings
Peter	Griffin	\$100
Lois	Griffin	\$150
Joe	Swanson	\$300

Example

```
th {  
  background-color: #4CAF50;  
  color: white;  
}
```

Responsive Table

A responsive table will display a horizontal scroll bar if the screen is too small to display the full content:

First Name	Last Name	Points	Points	Points	Points	Points	Points	Points	Points	Points	Points	Points	Points
Jill	Smith	50	50	50	50	50	50	50	50	50	50	50	50
Eve	Jackson	94	94	94	94	94	94	94	94	94	94	94	94
Adam	Johnson	67	67	67	67	67	67	67	67	67	67	67	67

add a container element (like `<div>`) with `overflow-x:auto` around the `<table>` element to make it responsive:

EXAMPLE SOURCE OF THE TABLE ABOVE

```
<div style="overflow-x:auto;">
```

```
<table>
```

```
... table content ...
```

```
</table>
```

```
</div>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
table {
```

```
    border-collapse: collapse;
```

```
    width: 100%;
```

```
}
```

```
th, td {
```

```
    text-align: left;
```

```
    padding: 8px;
```

```
}
```

```
tr:nth-child(even){ background-color: #f2f2f2}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h2>Responsive Table</h2>
```

```
<p>A responsive table will display a horizontal scroll bar if the screen is too
```

```
small to display the full content. Resize the browser window to see the effect:</p>
```

```
<p>To create a responsive table, add a container element (like div) with <strong>overflow-x:auto</strong> around the table element:</p>
```

```
<div style="overflow-x:auto;">
```

```
<table>
```

```
<tr>
```

```
<th>First Name</th>
```

```
<th>Last Name</th>
```

```
<th>Points</th>
```

```
<th>Points</th>
```

```
<th>Points</th>
```

```
<th>Points</th>
```

```
<th>Points</th>
```

```
<th>Points</th>
```

```
<th>Points</th>
```

```
<th>Points</th>
```

```
<th>Points</th>
```

```
<th>Points</th>
```

```
</tr>
```

```
<tr>

<td>Jill</td>

<td>Smith</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

<td>50</td>

</tr>

<tr>

<td>Eve</td>

<td>Jackson</td>

<td>94</td>

<td>94</td>

<td>94</td>

<td>94</td>

<td>94</td>

<td>94</td>

<td>94</td>

<td>94</td>

<td>94</td>
```

```

</tr>

<tr>

  <td>Adam</td>

  <td>Johnson</td>

  <td>67</td>

  <td>67</td>

  <td>67</td>

  <td>67</td>

  <td>67</td>

  <td>67</td>

  <td>67</td>

  <td>67</td>

  <td>67</td>

  <td>67</td>

  <td>67</td>

</tr>

</table>

</div>

</body>

</html>

```

CSS Layout - The display Property

The `display` property is the most important CSS property for controlling layout.

The display Property

The `display` property specifies if/how an element is displayed.

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is `block` or `inline`.

Block-level Elements

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).

The `<div>` element is a block-level element.

Examples of block-level elements:

- `<div>`
- `<h1>` - `<h6>`
- `<p>`
- `<form>`
- `<header>`
- `<footer>`
- `<section>`

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline `` element inside a paragraph.

Examples of inline elements:

- ``
- `<a>`
- ``

Display: none;

`display: none;` is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.

The `<script>` element use `display: none;` as its default.

Override The Default Display Value

As mentioned, every element has a default display value. However, you can override this.

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:

Example

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

N

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

The following example displays `` elements as block elements:

Example

```
span {display: block;  
}
```

The following example displays `<a>` elements as block elements:

Example

```
a {  
  display: block;  
}
```

Hide an Element - `display:none` or `visibility:hidden`?

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:

Example

```
h1.hidden {  
  display: none;  
}
```

`visibility:hidden;` also hides an element.

However, the element will still take up the same space as before. The element will be hidden, but still affect the layout:

Example

```
h1.hidden {  
  visibility: hidden;  
}
```


a. CSS Layout - width and max-width

Using width, max-width and margin: auto;

As mentioned in the previous chapter; a block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Setting the `width` of a block-level element will prevent it from stretching out to the edges of its container. Then, you can set the margins to `auto`, to horizontally center the element within its container. The element will take up the specified width, and the remaining space will be split equally between the two margins:

This `<div>` element has a width of 500px, and margin set to `auto`.

Note: The problem with the `<div>` above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page.

Using `max-width` instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices:

This `<div>` element has a max-width of 500px, and margin set to `auto`.

Tip: Resize the browser window to less than 500px wide, to see the difference between the two divs!

Here is an example of the two divs above:

Example

```
div.ex1 {  
  width: 500px;  
  margin: auto;  
  border: 3px solid #73AD21;  
}
```

```
div.ex2 {  
  max-width: 500px;  
  margin: auto;  
  border: 3px solid #73AD21;  
}
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
div.ex1 {  
    width:500px;  
    margin: auto;  
    border: 3px solid #73AD21;  
}
```

```
div.ex2 {  
    max-width:500px;  
    margin: auto;  
    border: 3px solid #73AD21;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div class="ex1">This div element has width: 500px;</div>
```

```
<br>
```

```
<div class="ex2">This div element has max-width: 500px;</div>
```

```
<p><strong>Tip:</strong> Drag the browser window to smaller than 500px wide, to see the difference  
between
```

```
the two divs!</p>
```

```
</body>
```

```
</html>
```

21 . CSS Layout - The position Property

The `position` property specifies the type of positioning method used for an element (static, relative, fixed or absolute).

The position Property

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

There are four different position values:

- `static`
- `relative`
- `fixed`
- `absolute`

Elements are then positioned using the `top`, `bottom`, `left`, and `right` properties. However, these properties will not work unless the `position` property is set first. They also work differently depending on the position value.

`position: static;`

HTML elements are positioned static by default.

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

An element with `position: static;` is not positioned in any special way; it is always positioned according to the normal flow of the page:

This `<div>` element has `position: static;`

Here is the CSS that is used:

Example

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

`position: relative;`

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.

This `<div>` element has `position: relative;`

Here is the CSS that is used:

Example

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

position: fixed;

An element with `position: fixed;` is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

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

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

Example

```
div.fixed {  
  position: fixed;  
  bottom: 0;  
  right: 0;  
  width: 300px;  
  border: 3px solid #73AD21;  
}
```

position: absolute;

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However, if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Note: A "positioned" element is one whose position is anything except `static`.

Here is a simple example:

This `<div>` element has `position: relative;`

This `<div>` element has `position: absolute;`

Here is the CSS that is used:

Example

```
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;  
}
```

8. 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:

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

Example

```
img {  
}
```

An element with greater stack order is always in front of an element with a lower stack order.

Note: If two positioned elements overlap without a `z-index` specified, the element positioned last in the HTML code will be shown on top.

Positioning Text In an Image

How to position text over an image:

Example



|}”{:PUYH UJIKL;/”|

Bottom Left

Top Left

Top Right

Bottom Right

Centered

CSS Layout - Overflow

The CSS `overflow` property specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area.

The `overflow` property 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

Note: The `overflow` property only works for block elements with a specified height.

Note: In OS X Lion (on Mac), scrollbars are hidden by default and only shown when being used (even though "overflow:scroll" is set).

Visible

By default, the overflow is `visible`, meaning that it is not clipped and it renders outside the element's box:

You can use the `overflow` property when you want to have better control of the layout. The `overflow` property specifies what happens if content overflows an element's box.

Example

```
div {  
  width: 200px;  
  height: 50px;  
  background-color: #eee;  
  overflow: visible;  
}
```

edit

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
div {
```

```
background-color: #eee;
```

```
width: 200px;
```

```
height: 50px;
```

```
border: 1px dotted black;
```

```
overflow: visible;
```

```
</style>
```

```
</head><body>
```

```
<h2>CSS Overflow</h2>
```

```
<p>By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box:</p>
```

```
<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>
```

```
</body>
```

```
</html>
```

CSS Layout - float and clear

The `float` property specifies whether or not an element should float.

The `clear` property is used to control the behavior of floating elements.

The float Property

In its simplest use, the `float` property can be used to wrap text around images.

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

Example

```
img {  
  float: right;  
  margin: 0 0 10px 10px;  
}
```

The clear Property

The `clear` property is used to control the behavior of floating elements.

Elements after a floating element will flow around it. To avoid this, use the `clear` property.

The `clear` property specifies on which sides of an element floating elements are not allowed to float:

Example

```
div {  
  clear: left;  
}
```

CSS Layout - float and clear

The `float` property specifies whether or not an element should float.

The `clear` property is used to control the behavior of floating elements.

The float Property

In its simplest use, the `float` property can be used to wrap text around images.

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

Example

```
img {  
  float: right;  
  margin: 0 0 10px 10px;  
}
```


The clear Property

The `clear` property is used to control the behavior of floating elements.

Elements after a floating element will flow around it. To avoid this, use the `clear` property.

The `clear` property specifies on which sides of an element floating elements are not allowed to float:

Example

```
div {  
  clear: left;  
}
```

The clearfix Hack - overflow: auto;

If an element is taller than the element containing it, and it is floated, it will overflow outside of its container.

Then we can add `overflow: auto;` to the containing element to fix this problem:

Example

```
.clearfix {  
  overflow: auto;  
}
```

Web Layout Example

It is common to do entire web layouts using the `float` property:

Example

```
div {  
  border: 3px solid blue;  
}  
  
.clearfix {  
  overflow: auto;  
}  
  
nav {  
  float: left;  
  width: 200px;  
  border: 3px solid #73AD21;  
}  
  
section {  
  margin-left: 206px;
```

```
border: 3px solid red;
}
```

Edit

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
div {
```

```
    border: 3px solid blue;
```

```
}
```

```
.clearfix {
```

```
    overflow: auto;
```

```
}
```

```
nav {
```

```
    float: left;
```

```
    width: 200px;
```

```
    border: 3px solid #73AD21;
```

```
}
```

```
section {
```

```
    margin-left: 206px;
```

```
border: 3px solid red;

}

</style>

</head>

<body>

<div class="clearfix">

<nav>

<span>nav</span>

<ul>

<li><a target="_blank" href="/default.asp">Home</a></li>

<li><a target="_blank" href="default.asp">CSS</a></li>

<li><a target="_blank" href="/html/default.asp">HTML</a></li>

<li><a target="_blank" href="/js/default.asp">JavaScript</a></li>

</ul>

</nav>

<section>

<span>section</span>

<p>Notice we have put a clearfix on the div container. It is not needed in this example, but it would be if the nav element
was longer than the non-floated section content.</p>
```

</section>

<section>

section

<p>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.</p>

</section>

</div>

</body>

</html>

Demo

nav

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

section

Notice we have put a clearfix on the div container. It is not needed in this example, but it would be if the nav element was longer than the non-floated section content

section

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.

The inline-block Value

It has been possible for a long time to create a grid of boxes that fills the browser width and wraps nicely (when the browser is resized), by using the `float` property.

However, the `inline-block` value of the `display` property makes this even easier.

inline-block elements are like inline elements but they can have a width and a height.

Examples

The old way - using `float` (notice that we also need to specify a `clear` property for the element after the floating boxes):

Example

```
.floating-box {  
  float: left;  
  width: 150px;  
  height: 75px;  
  margin: 10px;  
  border: 3px solid #73AD21;  
}
```

```
.after-box {  
  clear: left;
```

The same effect can be achieved by using the `inline-block` value of the `display` property (notice that no `clear` property is needed):

Example

```
.floating-box {  
  display: inline-block;  
  width: 150px;  
  height: 75px;  
  margin: 10px;  
  border: 3px solid #73AD21;  
}
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
.floating-box {  
  
    display: inline-block;  
  
    width: 150px;  
  
    height: 75px;  
  
    margin: 10px;  
  
    border: 3px solid #73AD21;  
  
}
```

```
.after-box {  
  
    border: 3px solid red;  
  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h2>The New Way - using inline-block</h2>
```

```
<div class="floating-box">Floating box</div>
```

```
<div class="floating-box">Floating box</div>
```

```
<div class="floating-box">Floating box</div>
```

```
<div class="floating-box">Floating box</div>
```

```
<div class="floating-box">Floating box</div>
```

```
<div class="floating-box">Floating box</div>
```

```
<div class="floating-box">Floating box</div>
```

```
<div class="floating-box">Floating box</div>
```

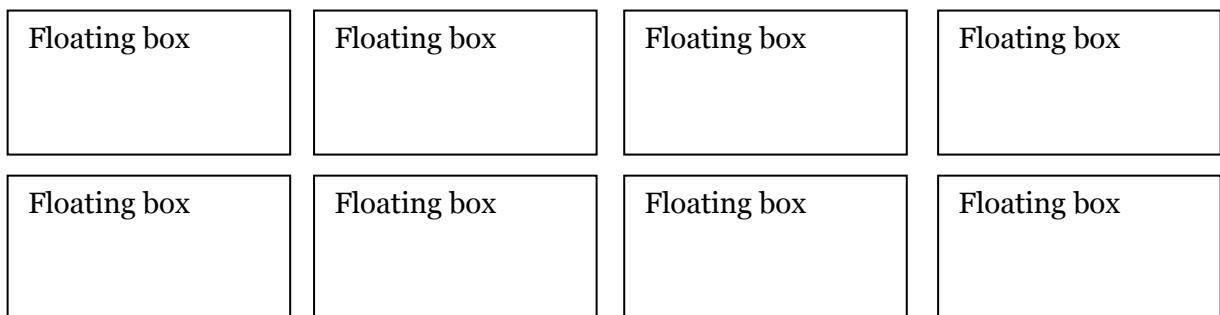
```
<div class="after-box">Another box, after the floating boxes...</div>
```

```
</body>
```

```
</html>
```

Demo

The New Way - using inline-block



CSS Layout - Horizontal & Vertical Align

Center Align Elements

To horizontally center a block element (like `<div>`), use `margin: auto;`

Setting the width of the element will prevent it from stretching out to the edges of its container.

The element will then take up the specified width, and the remaining space will be split equally between the two margins:

This div element is centered.

Example

```
.center {  
  margin: auto;  
  width: 50%;  
  border: 3px solid green;
```

```
padding: 10px;  
}
```

Note: Center aligning has no effect if the `width` property is not set (or set to 100%).

Center Align Text

To just center the text inside an element, use `text-align: center;`

This text is centered.

Example

```
.center {  
  text-align: center;  
  border: 3px solid green;  
}.
```

Center an Image

To center an image, use `margin: auto;` and make it into a **block** element:



Example

```
img {  
  display: block;  
  margin: auto;  
  width: 40%;  
}
```


Left and Right Align - Using position

One method for aligning elements is to use `position: absolute;`

In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since.

Example

```
.right {  
  position: absolute;  
  right: 0px;  
  width: 300px;  
  border: 3px solid #73AD21;  
  padding: 10px;  
}
```

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

Tip: When aligning elements with `position`, always define `margin` and `padding` for the `<body>` element. This is to avoid visual differences in different browsers.

There is also a problem with IE8 and earlier, when using `position`. If a container element (in our case `<div class="container">`) has a specified width, and the `!DOCTYPE` declaration is missing, IE8 and earlier versions will add a 17px margin on the right side. This seems to be space reserved for a scrollbar. So, always set the `!DOCTYPE` declaration when using `position`:

Example

```
body {  
  margin: 0;  
  padding: 0;  
}  
  
.container {  
  position: relative;  
  width: 100%;  
}  
  
.right {  
  position: absolute;  
  right: 0px;  
  width: 300px;  
  background-color: #b0e0e6;  
}
```

Left and Right Align - Using float

Another method for aligning elements is to use the `float` property:

Example

```
.right {  
  float: right;  
  width: 300px;  
  border: 3px solid #73AD21;  
  padding: 10px;  
}
```

Tip: When aligning elements with `float`, always define `margin` and `padding` for the `<body>` element. This is to avoid visual differences in different browsers.

There is also a problem with IE8 and earlier, when using `float`. If the `!DOCTYPE` declaration is missing, IE8 and earlier versions will add a 17px margin on the right side. This seems to be space reserved for a scrollbar. So, always set the `!DOCTYPE` declaration when using `float`:

Example

```
body {  
  margin: 0;  
  padding: 0;  
}  
  
.right {  
  float: right;  
  width: 300px;  
  background-color: #b0e0e6;  
}
```

Center Vertically - Using padding

There are many ways to center an element vertically in CSS. A simple solution is to use top and bottom padding:

I am vertically centered.

Example

```
.center {  
  padding: 70px 0;  
  border: 3px solid green;  
}
```

To center both vertically and horizontally, use `padding` and `text-align: center`:

I am vertically and horizontally centered.

Example

```
.center {  
  padding: 70px 0;  
  border: 3px solid green;  
  text-align: center;  
}
```

Center Vertically - Using line-height

Another trick is to use the `line-height` property with a value that is equal to the `height` property.

I am vertically and horizontally centered.

Example

```
.center {  
  line-height: 200px;  
  height: 200px;  
  border: 3px solid green;  
  text-align: center;  
}
```

/ If the text has multiple lines, add the following: */*

```
.center p {  
  line-height: 1.5;  
  display: inline-block;  
  vertical-align: middle;  
}
```

Center Vertically - Using position & transform

If `padding` and `line-height` is not an option, a third solution is to use positioning and the `transform` property:

I am vertically and horizontally centered.

Example

```
.center {  
  height: 200px;  
  position: relative;  
  border: 3px solid green;  
}
```

```
.center p {  
  margin: 0;  
  position: absolute;  
  top: 50%;  
  left: 50%;
```

```
    transform: translate(-50%, -50%);  
}
```

Edit

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
.center {
```

```
    height: 200px;
```

```
    position: relative;
```

```
    border: 3px solid green;
```

```
}
```

```
.center p {
```

```
    margin: 0;
```

```
    position: absolute;
```

```
    top: 50%;
```

```
    left: 50%;
```

```
    -ms-transform: translate(-50%, -50%);
```

```
    transform: translate(-50%, -50%);
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h2>Centering</h2>
```

```
<p>In this example, we use positioning and the transform property to vertically and horizontally center the div element:</p>
```

```
<div class="center">
```

```
<p>I am vertically and horizontally centered.</p>
```

```
</div>
```

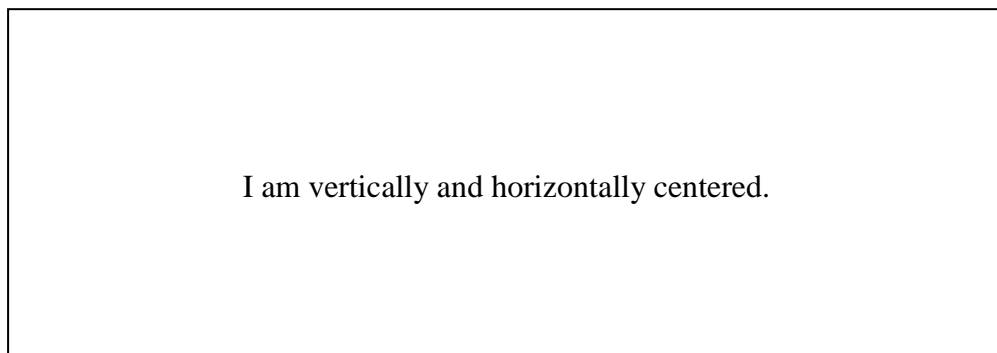
```
<p>Note: The transform property is not supported in IE8 and earlier versions.</p>
```

```
</body>
```

```
</html>
```

Centering

In this example, we use positioning and the transform property to vertically and horizontally center the div element:



Note: The transform property is not supported in IE8 and earlier versions.

CSS Combinators

A combinator is something that explains the relationship between the selectors.

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS3:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

Descendant Selector

The descendant selector matches all elements that are descendants of a specified element.

The following example selects all <p> elements inside <div> elements:

Example

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

Child Selector

The child selector selects all elements that are the immediate children of a specified element.

The following example selects all <p> elements that are immediate children of a <div> element:

Example

```
div > p {  
  background-color: yellow;  
}
```

Adjacent Sibling Selector

The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

The following example selects all <p> elements that are placed immediately after <div> elements:

Example

```
div + p {  
  background-color: yellow;  
}
```

General Sibling Selector

The general sibling selector selects all elements that are siblings of a specified element.

The following example selects all <p> elements that are siblings of <div> elements:

Example

```
div ~ p {  
    background-color: yellow;  
}
```

edit

```
<!DOCTYPE html>  
  
<html>  
  
<head>  
  
<style>  
div ~ p {  
    background-color: yellow;  
}  
  
</style>  
  
</head>  
  
<body>  
  
<div>  
    <p>Paragraph 1 in the div.</p>  
    <p>Paragraph 2 in the div.</p>  
</div>  
  
<p>Paragraph 3. Not in a div.</p>  
<p>Paragraph 4. Not in a div.</p>  
  
</body>  
  
</html>
```

Demo

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3. Not in a div.

Paragraph 4. Not in a div.

CSS 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

Mouse Over Me



Syntax

The syntax of pseudo-classes:

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

Anchor Pseudo-classes

Links can be displayed in different ways:

Example

```
/* unvisited link */  
a:link {  
  color: #FF0000;  
}  
  
/* visited link */  
a:visited {  
  color: #0000FF;  
}  
  
/* mouse over link */  
a:hover {  
  color: #00FF00;  
}  
  
/* selected link */
```



```
a:active {  
  color: #0000FF;  
}
```

Note: `a:hover` MUST come after `a:link` and `a:visited` in the CSS definition in order to be effective!
`a:active` MUST come after `a:hover` in the CSS definition in order to be effective! Pseudo-class names are not case-sensitive.

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:

Example

```
a.highlight:hover {  
  color: #ff0000;  
}
```

Hover on <div>

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

Example

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

Simple Tooltip Hover

Hover over a `<div>` element to show a `<p>` element (like a tooltip):

Hover over me to show the `<p>` element.

Example

```
p {  
  display: none;  
  background-color: yellow;  
  padding: 20px;  
}  
  
div:hover p {  
  display: block;  
}
```

CSS - The :first-child Pseudo-class

The `:first-child` pseudo-class matches a specified element that is the first child of another element.

Match the first <p> element

In the following example, the selector matches any `<p>` element that is the first child of any element:

Example

```
p:first-child {  
  color: blue;  
}
```

Match the first <i> element in all <p> elements

In the following example, the selector matches the first `<i>` element in all `<p>` elements:

Example

```
p i:first-child {  
  color: blue;  
}
```

Match all <i> elements in all first child <p> elements

In the following example, the selector matches all `<i>` elements in `<p>` elements that are the first child of another element:

Example

```
p:first-child i {  
  color: blue;  
}
```

CSS - The :lang Pseudo-class

The `:lang` pseudo-class allows you to define special rules for different languages.

In the example below, `:lang` defines the quotation marks for `<q>` elements with `lang="no"`:

Example

```
<html>  
<head>  
<style>  
  q:lang(no) {  
    quotes: "~" "~";  
  }  
</style>
```

</head>

<body>

<p>Some text <q lang="no">A quote in a paragraph</q> Some text.</p>

</body>

</html>

Edit

<!DOCTYPE html>

<html>

<head>

<style>

div {

background-color: green;

color: white;

padding: 25px;

text-align: center;

}

div:hover {

background-color: blue;

}

</style>

</head>

<body>

<p>Mouse over the div element below to change its background color:</p>

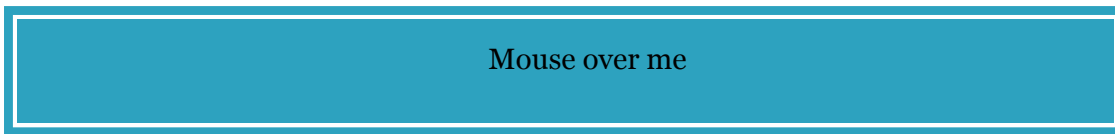
<div>Mouse Over Me</div>

</body>

</html>

Demo

Mouse over the div element below to change its background color:



CSS 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

Syntax

The syntax of pseudo-elements:

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

Notice the double colon notation - `::first-line` versus `:first-line`

The double colon replaced the single-colon notation for pseudo-elements in CSS3. This was an attempt from W3C to distinguish between **pseudo-classes** and **pseudo-elements**.

The single-colon syntax was used for both pseudo-classes and pseudo-elements in CSS2 and CSS1.

For backward compatibility, the single-colon syntax is acceptable for CSS2 and CSS1 pseudo-elements

The ::first-line Pseudo-element

The `::first-line` pseudo-element is used to add a special style to the first line of a text.

The following example formats the first line of the text in all `<p>` elements:

Example

```
p::first-line {  
  color: #ff0000;  
  font-variant: small-caps;  
}
```

Note: The `::first-line` pseudo-element can only be applied to block-level elements.

The following properties apply to the `::first-line` pseudo-element:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

The `::first-letter` Pseudo-element

The `::first-letter` pseudo-element is used to add a special style to the first letter of a text.

The following example formats the first letter of the text in all `<p>` elements:

Example

```
p::first-letter {  
  color: #ff0000;  
  font-size: xx-large;  
}
```

Note: The `::first-letter` pseudo-element can only be applied to block-level elements.

The following properties apply to the `::first-letter` pseudo- element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

Pseudo-elements and CSS Classes

Pseudo-elements can be combined with CSS classes:

Example

```
p.intro::first-letter {  
  color: #ff0000;  
  font-size: 200%;  
}
```

The example above will display the first letter of paragraphs with class="intro", in red and in a larger size.

Multiple Pseudo-elements

Several pseudo-elements can also be combined.

In the following example, the first letter of a paragraph will be red, in an xx-large font size. The rest of the first line will be blue, and in small-caps. The rest of the paragraph will be the default font size and color:

Example

```
p::first-letter {  
  color: #ff0000;  
  font-size: xx-large;  
}  
  
p::first-line {  
  color: #0000ff;  
  font-variant: small-caps;  
}
```

CSS - The ::before Pseudo-element

The ::before pseudo-element can be used to insert some content before the content of an element.

The following example inserts an image before the content of each <h1> element:

Example

```
h1::before {  
  content: url(smiley.gif);  
}
```

CSS - The ::after Pseudo-element

The ::after pseudo-element can be used to insert some content after the content of an element.

The following example inserts an image after the content of each <h1> element:

Example

```
h1::after {  
  content: url(smiley.gif);  
}
```

CSS - The ::selection Pseudo-element

The `::selection` pseudo-element matches the portion of an element that is selected by a user.

The following CSS properties can be applied to `::selection`: `color`, `background`, `cursor`, and `outline`.

The following example makes the selected text red on a yellow background:

Example

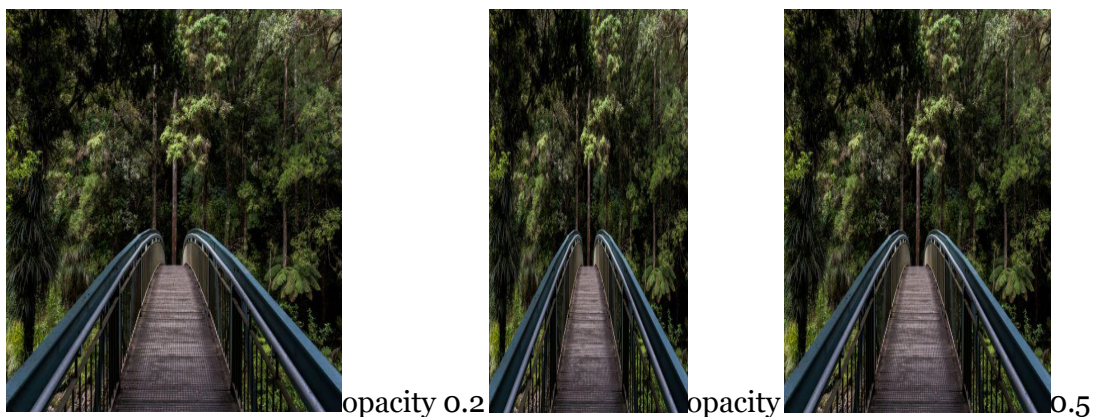
```
::selection {  
  color: red;  
  background: yellow;  
}
```

CSS Opacity / Transparency

The `opacity` property specifies the opacity/transparency of an element.

Transparent Image

The `opacity` property can take a value from 0.0 - 1.0. The lower value, the more transparent:



opacity 1
(default)

Note: IE8 and earlier use `filter:alpha(opacity=x)`. The `x` can take a value from 0 - 100. A lower value makes the element more transparent.

Example

```
img {  
  opacity: 0.5;  
}
```

```
filter: alpha(opacity=50); /* For IE8 and earlier */  
}
```

Transparent Hover Effect

The `opacity` property is often used together with the `:hover` selector to change the opacity on mouse-over:



Example

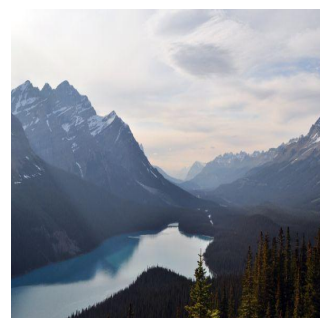
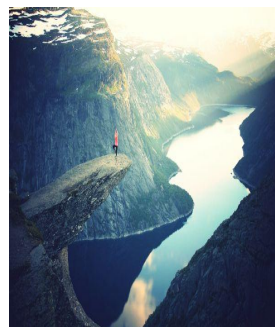
```
img {  
  opacity: 0.5;  
  filter: alpha(opacity=50); /* For IE8 and earlier */  
}  
  
img:hover {  
  opacity: 1.0;  
  filter: alpha(opacity=100); /* For IE8 and earlier */  
}
```

Example explained

The first CSS block is similar to the code in Example 1. In addition, we have added what should happen when a user hovers over one of the images. In this case we want the image to NOT be transparent when the user hovers over it. The CSS for this is `opacity:1;`

When the mouse pointer moves away from the image, the image will be transparent again.

An example of reversed hover effect:




Example

```
img:hover {  
  opacity: 0.5;  
  filter: alpha(opacity=50); /* For IE8 and earlier */  
}
```

Transparent Box

When using the `opacity` property to add transparency to the background of an element, all of its child elements become transparent as well. This can make the text inside a fully transparent element hard to read:



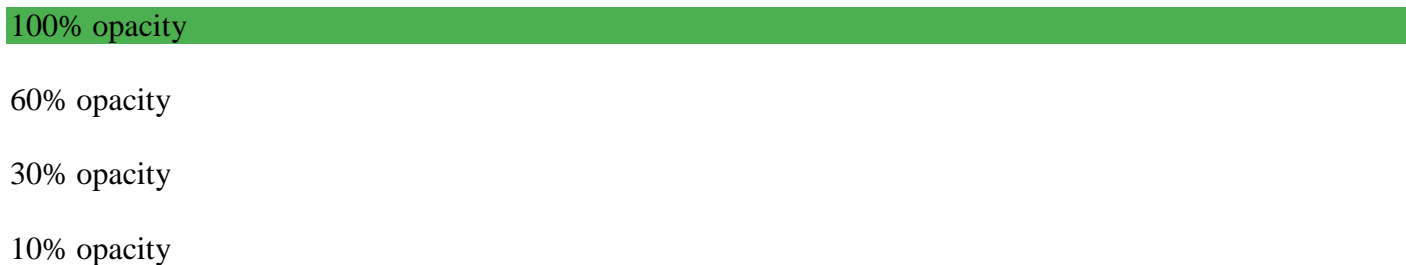
opacity 1
opacity 0.6
opacity 0.3
opacity 0.1

Example

```
div {  
  opacity: 0.3;  
  filter: alpha(opacity=30); /* For IE8 and earlier */  
}
```

Transparency using RGBA

If you do not want to apply opacity to child elements, like in our example above, use **RGBA** color values. The following example sets the opacity for the background color and not the text:



100% opacity
60% opacity
30% opacity
10% opacity

You learned from our [CSS Colors Chapter](#), that you can use RGB as a color value. In addition to RGB, CSS3 introduced an RGB color value with an alpha channel (RGBA) - 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 (fully opaque).

Tip: You will learn more about RGBA Colors in our [CSS3 Colors Chapter](#).

Example

```
div {  
    background: rgba(76, 175, 80, 0.3) /* Green background with 30% opacity */  
}
```

Text in Transparent Box

This is some text that is placed in the transparent box.

Example

```
<html>  
<head>  
<style>  
div.background {  
    background: url(klematis.jpg) repeat;  
    border: 2px solid black;  
}  
  
div.transbox {  
    margin: 30px;  
    background-color: #ffffff;  
    border: 1px solid black;  
    opacity: 0.6;  
    filter: alpha(opacity=60); /* For IE8 and earlier */  
}  
  
div.transbox p {  
    margin: 5%;  
    font-weight: bold;  
    color: #000000;  
}  
</style>  
</head>  
<body>  
  
<div class="background">  
    <div class="transbox">  
        <p>This is some text that is placed in the transparent box.</p>  
    </div>  
</div>  
  
</body>  
</html>
```

Edit

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
div.background {
```

```
    background: url(klematis.jpg) repeat;
```

```
    border: 2px solid black;
```

```
}
```

```
div.transbox {
```

```
    margin: 30px;
```

```
    background-color: #ffffff;
```

```
    border: 1px solid black;
```

```
    opacity: 0.6;
```

```
    filter: alpha(opacity=60); /* For IE8 and earlier */
```

```
}
```

```
div.transbox p {
```

```
    margin: 5%;
```

```
    font-weight: bold;
```

```
    color: #000000;
```

```

}

</style>

</head>

<body>

<div class="background">

  <div class="transbox">

    <p>This is some text that is placed in the transparent box.</p>

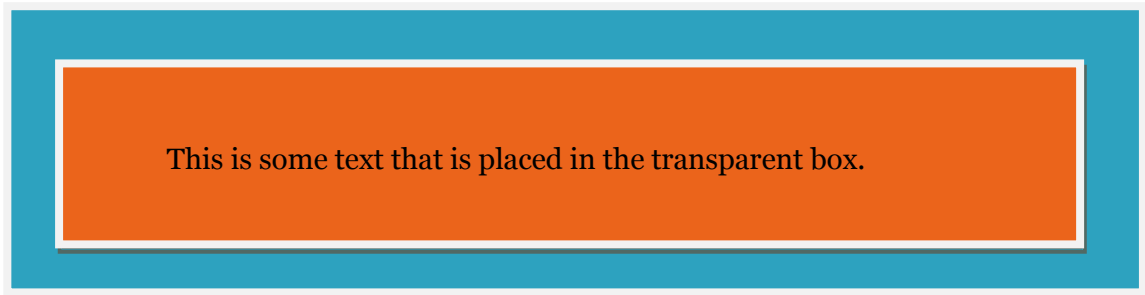
  </div>

</div>

</body>

</html>

```



CSS Navigation Bar

Demo:

Vertical

[Home](#)
[News](#)
[Contact](#)
[About](#)

Horizontal

[Home](#)

[News](#)

[Contact](#)

[About](#)

Navigation Bars

Having easy-to-use navigation is important for any web site.

With CSS you can transform boring HTML menus into good-looking navigation bars.

Navigation Bar = List of Links

A navigation bar needs standard HTML as a base.

In our examples we will build the navigation bar from a standard HTML list.

A navigation bar is basically a list of links, so using the and elements makes perfect sense:

Example

```
<ul>
  <li><a href="default.asp">Home</a></li>
  <li><a href="news.asp">News</a></li>
  <li><a href="contact.asp">Contact</a></li>
  <li><a href="about.asp">About</a></li>
</ul>
```

Now let's remove the bullets and the margins and padding from the list:

Example

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
```

demo

```
<!DOCTYPE html>
```

```
<html>
```

```

<body>

<ul>

<li><a href="#home">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>

<p>Note: We use href="#" for test links. In a real web site this would be URLs.</p>

</body>

</html>

```

Example explained:

- `list-style-type: none;` - Removes the bullets. A navigation bar does not need list markers
- Set `margin: 0;` and `padding: 0;` to remove browser default settings

The code in the example above is the standard code used in both vertical, and horizontal navigation bars.

Vertical Navigation Bar

To build a vertical navigation bar, you can style the `<a>` elements inside the list, in addition to the code above:

Example 2

```

li a {
  display: block;
  width: 60px;
}

```

Example explained:

- `display: block;` - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width (and padding, margin, height, etc. if you want)
- `width: 60px;` - Block elements take up the full width available by default. We want to specify a 60 pixels width

You can also set the width of ``, and remove the width of `<a>`, as they will take up the full width available when displayed as block elements. This will produce the same result as our previous example:

Example

```
ul {  
  list-style-type: none;  
  margin: 0;  
  padding: 0;  
  width: 60px;  
}  
  
li a {  
  display: block;  
}
```

Vertical Navigation Bar Examples

Create a basic vertical navigation bar with a gray background color and change the background color of the links when the user moves the mouse over them:

[Home](#)

[News](#)

[Contact](#)

[About](#)

Example

```
ul {  
  list-style-type: none;  
  margin: 0;  
  padding: 0;  
  width: 200px;  
  background-color: #f1f1f1;  
}  
  
li a {  
  display: block;  
  color: #000;  
  padding: 8px 16px;  
  text-decoration: none;  
}  
  
/* Change the link color on hover */  
li a:hover {  
  background-color: #555;  
  color: white;  
}
```

demo

```
<!DOCTYPE html>

<html>

<head>

<style>

ul {

    list-style-type: none;

    margin: 0;

    padding: 0;

    width: 60px;

}

li a {

    display: block;

    background-color: #dddddd;

}

</style>

</head>

<body>

<ul>

<li><a href="#home">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>

<p>A background color is added to the links to show the link area.</p>
```


<p>Notice that the whole link area is clickable, not just the text.</p>

</body>

</html>

Active/Current Navigation Link

Add an "active" class to the current link to let the user know which page he/she is on:

[Home](#)

[News](#)

[Contact](#)

[About](#)

Example

```
.active {  
  background-color: #4CAF50;  
  color: white;  
}
```

Center Links & Add Borders

Add `text-align:center` to `` or `<a>` to center the links.

Add the `border` property to `` add a border around the navbar. If you also want borders inside the navbar, add a `border-bottom` to all `` elements, except for the last one:

[Home](#)

[News](#)

[Contact](#)

[About](#)

Example

```
ul {  
  border: 1px solid #555;  
}  
  
li {  
  text-align: center;  
  border-bottom: 1px solid #555;
```

```

}

li:last-child {
  border-bottom: none;
}

```

Full-height Fixed Vertical Navbar

Create a full-height, "sticky" side navigation:

Example

```

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 25%;
  background-color: #f1f1f1;
  height: 100%; /* Full height */
  position: fixed; /* Make it stick, even on scroll */
  overflow: auto; /* Enable scrolling if the sidenav has too much content */
}

```

Note: This example might not work properly on mobile devices.

Horizontal Navigation Bar

There are two ways to create a horizontal navigation bar. Using **inline** or **floating** list items.

Inline List Items

One way to build a horizontal navigation bar is to specify the `` elements as inline, in addition to the "standard" code above:

Example

```

li {
  display: inline;
}

```

Example explained:

- `display: inline;` - By default, `` elements are block elements. Here, we remove the line breaks before and after each list item, to display them on one line

Floating List Items

Another way of creating a horizontal navigation bar is to float the `` elements, and specify a layout for the navigation links:

Example

```
li {  
  float: left;  
}  
  
a {  
  display: block;  
  padding: 8px;  
  background-color: #dddddd;  
}
```

Example explained:

- `float: left;` - use float to get block elements to slide next to each other
- `display: block;` - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify padding (and height, width, margins, etc. if you want)
- `padding: 8px;` - Since block elements take up the full width available, they cannot float next to each other. Therefore, specify some padding to make them look good
- `background-color: #dddddd;` - Add a gray background-color to each a element

Tip: Add the background-color to `` instead of each `<a>` element if you want a full-width background color:

Example

```
ul {  
  background-color: #dddddd;  
}
```

Horizontal Navigation Bar Examples

Create a basic horizontal navigation bar with a dark background color and change the background color of the links when the user moves the mouse over them:

Home	News	Contact	About
----------------------	----------------------	-------------------------	-----------------------

Example

```
ul {  
  list-style-type: none;  
  margin: 0;  
  padding: 0;  
  overflow: hidden;  
  background-color: #333;  
}
```

```

}

li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

/* Change the link color to #111 (black) on hover */
li a: hover {
  background-color: #111;
}

```

demo

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
ul {
```

```
  list-style-type: none;
```

```
  margin: 0;
```

```
  padding: 0;
```

```
}
```

```
li {
```

```
  display: inline;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<ul>

<li><a href="#home">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>

</body>

</html>
```

Active/Current Navigation Link

Add an "active" class to the current link to let the user know which page he/she is on:

[Home](#)

[News](#)

[Contact](#)

[About](#)

Example

```
.active {
  background-color: #4CAF50;
}
```

Right-Align Links

Right-align links by floating the list items to the right (`float:right`):

[Home](#)

[News](#)

[Contact](#)

[About](#)

Example

```

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

```

Border Dividers

Add the `border-right` property to `` to create link dividers:

[Home](#)

[News](#)

[Contact](#)

[About](#)

Example

```

/* Add a gray right border to all list items, except the last item (last-child) */
li {
  border-right: 1px solid #bbb;
}

li:last-child {
  border-right: none;
}

```

Fixed Navigation Bar

Make the navigation bar stay at the top or the bottom of the page, even when the user scrolls the page:

Fixed Top

```

ul {
  position: fixed;
  top: 0;
  width: 100%;
}

```

Fixed Bottom

```

ul {
  position: fixed;
  bottom: 0;
}

```

```
width: 100%;  
}
```

Note: These examples might not work properly on mobile devices.

Gray Horizontal Navbar

An example of a gray horizontal navigation bar with a thin gray border:

- [Home](#)
- [News](#)
- [Contact](#)
- [About](#)

Example

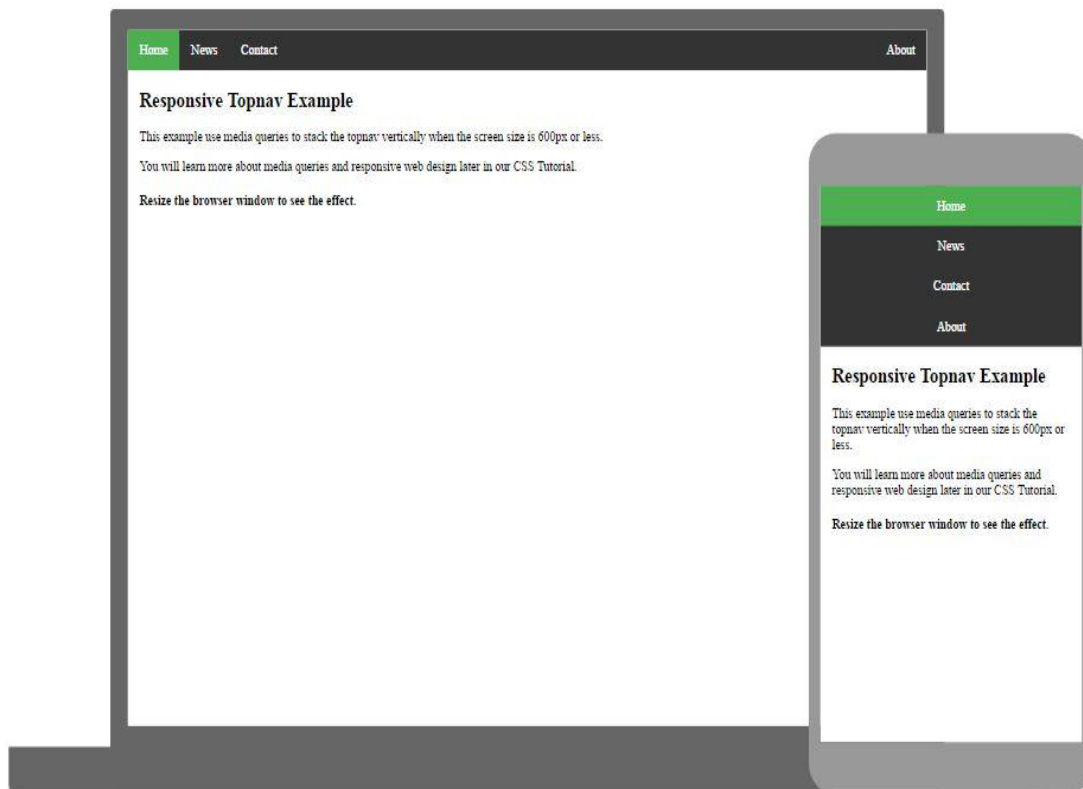
```
ul {  
  border: 1px solid #e7e7e7;  
  background-color: #f3f3f3;  
}
```

```
li a {  
  color: #666;  
}
```

More Examples

Responsive Topnav

How to use CSS3 media queries to create a responsive top navigation.



Responsive Sidenav

How to use CSS3 media queries to create a responsive side navigation.

Demo

```
<!DOCTYPE html>

<html>

<head>

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

<style>

body {margin: 0;}


ul.topnav {

    list-style-type: none;

    margin: 0;

    padding: 0;
```



```

    overflow: hidden;

    background-color: #333;

}

ul.topnav li {float: left;}

ul.topnav li a {

    display: block;

    color: white;

    text-align: center;

    padding: 14px 16px;

    text-decoration: none;

}

ul.topnav li a:hover:not(.active) {background-color: #111;}

ul.topnav li a.active {background-color: #4CAF50;}

ul.topnav li.right {float: right;}

@media screen and (max-width: 600px){

    ul.topnav li.right,

    ul.topnav li {float: none;}

}

</style>

</head>

<body>

<ul class="topnav">

    <li><a class="active" href="#home">Home</a></li>

    <li><a href="#news">News</a></li>

    <li><a href="#contact">Contact</a></li>

    <li class="right"><a href="#about">About</a></li>

```

```
</ul>
```

```
<div style="padding:0 16px;">
```

```
<h2>Responsive Topnav Example</h2>
```

```
<p>This example use media queries to stack the topnav vertically when the screen size is 600px or less.</p>
```

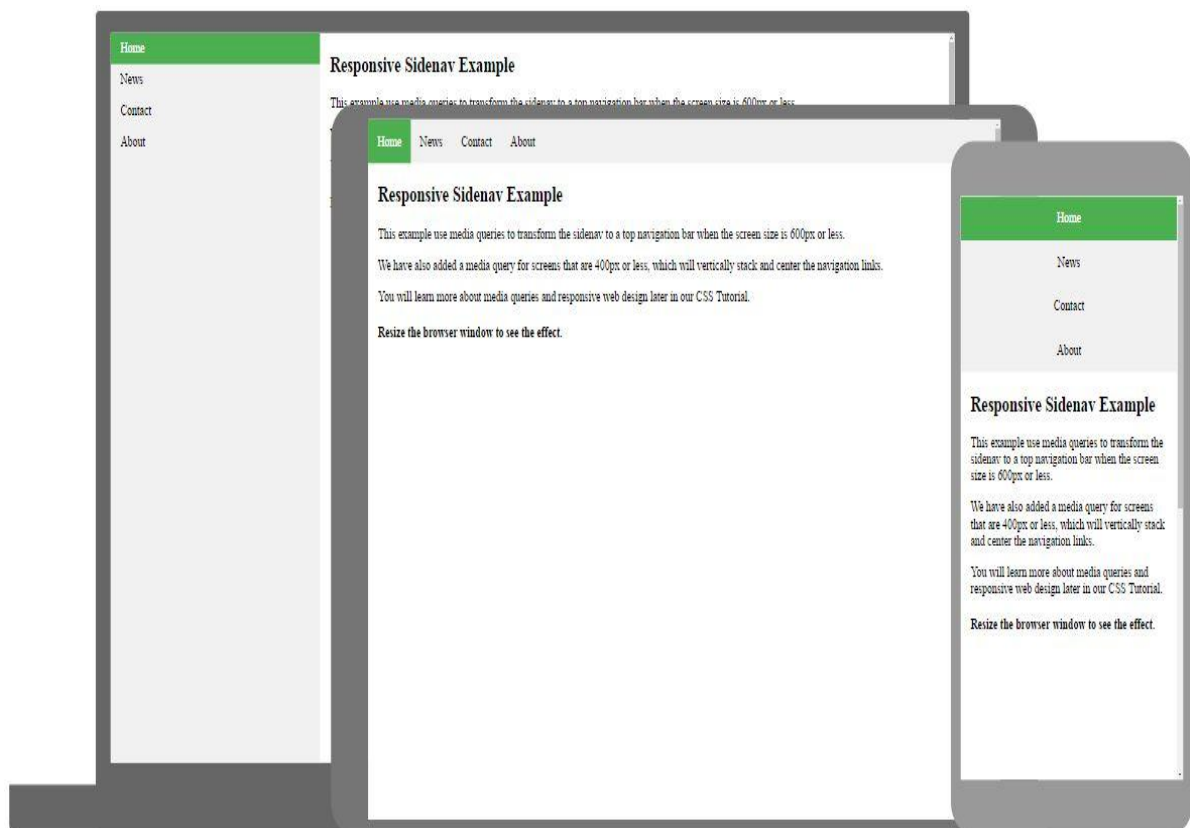
```
<p>You will learn more about media queries and responsive web design later in our CSS Tutorial.</p>
```

```
<h4>Resize the browser window to see the effect.</h4>
```

```
</div>
```

```
</body>
```

```
</html>
```



Dropdown Navbar

How to add a dropdown menu inside a navigation bar.

CSS Dropdowns

Create a hoverable dropdown with CSS.

Demo: Dropdown Examples

Move the mouse over the examples below:

Dropdown Text



Other:

Basic Dropdown

Create a dropdown box that appears when the user moves the mouse over an element.

Example

```
<style>
.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  padding: 12px 16px;
  z-index: 1;
}

.dropdown:hover .dropdown-content {
  display: block;
}
</style>

<div class="dropdown">
  <span>Mouse over me</span>
  <div class="dropdown-content">
    <p>Hello World!</p>
  </div>
</div>
```

```
</div>
</div>
```

Example Explained

HTML) Use any element to open the dropdown content, e.g. a ``, or a `<button>` element.

Use a container element (like `<div>`) to create the dropdown content and add whatever you want inside of it.

Wrap a `<div>` element around the elements to position the dropdown content correctly with CSS.

CSS) The `.dropdown` class use `position:relative`, which is needed when we want the dropdown content to be placed right below the dropdown button (using `position:absolute`).

The `.dropdown-content` class holds the actual dropdown content. It is hidden by default, and will be displayed on hover (see below). Note the `min-width` is set to 160px. Feel free to change this. **Tip:** If you want the width of the dropdown content to be as wide as the dropdown button, set the `width` to 100% (and `overflow:auto` to enable scroll on small screens).

Instead of using a border, we have used the CSS3 `box-shadow` property to make the dropdown menu look like a "card".

The `:hover` selector is used to show the dropdown menu when the user moves the mouse over the dropdown button.

Dropdown Menu

Create a dropdown menu that allows the user to choose an option from a list:

This example is similar to the previous one, except that we add links inside the dropdown box and style them to fit a styled dropdown button:

Example

```
<style>
/* Style The Dropdown Button */
.droptbn {
  background-color: #4CAF50;
  color: white;
  padding: 16px;
  font-size: 16px;
  border: none;
  cursor: pointer;
}

/* The container <div> - needed to position the dropdown content */
.dropdown {
  position: relative;
  display: inline-block;
```

```

}

/* Dropdown Content (Hidden by Default) */
.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
}

/* Links inside the dropdown */
.dropdown-content a {
  color: black;
  padding: 12px 16px;
  text-decoration: none;
  display: block;
}

/* Change color of dropdown links on hover */
.dropdown-content a:hover {background-color: #f1f1f1}

/* Show the dropdown menu on hover */
.dropdown:hover .dropdown-content {
  display: block;
}

/* Change the background color of the dropdown button when the dropdown content is shown */
.dropdown:hover .dropbtn {
  background-color: #3e8e41;
}
</style>

<div class="dropdown">
  <button class="dropbtn">Dropdown</button>
  <div class="dropdown-content">
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
    <a href="#">Link 3</a>
  </div>
</div>

```

Right-aligned Dropdown Content

If you want the dropdown menu to go from right to left, instead of left to right, add `right: 0;`

Example

```
.dropdown-content {  
  right: 0;  
}
```

More Examples

Dropdown Image

How to add an image and other content inside the dropdown box.

Hover over the image:



CSS Tooltip

Create tooltips with CSS.

Demo:

Tooltip Examples

A tooltip is often used to specify extra information about something when the user moves the mouse pointer over an element:

Top

Right

Bottom

Left

Basic Tooltip

Create a tooltip that appears when the user moves the mouse over an element:

Example

```
<style>  
/* Tooltip container */  
.tooltip {  
  position: relative;  
  display: inline-block;  
  border-bottom: 1px dotted black; /* If you want dots under the hoverable text */  
}  
  
/* Tooltip text */  
.tooltip .tooltiptext {  
  visibility: hidden;
```

```

width: 120px;
background-color: black;
color: #fff;
text-align: center;
padding: 5px 0;
border-radius: 6px;

/* Position the tooltip text - see examples below! */
position: absolute;
z-index: 1;
}

/* Show the tooltip text when you mouse over the tooltip container */
.tooltip:hover .tooltiptext {
  visibility: visible;
}
</style>

<div class="tooltip">Hover over me
  <span class="tooltiptext">Tooltip text</span>
</div>

```

Example Explained

HTML) Use a container element (like `<div>`) and add the `"tooltip"` class to it. When the user mouse over this `<div>`, it will show the tooltip text.

The tooltip text is placed inside an inline element (like ``) with `class="tooltiptext"`.

CSS) The `tooltip` class use `position:relative`, which is needed to position the tooltip text (`position:absolute`). **Note:** See examples below on how to position the tooltip.

The `tooltiptext` class holds the actual tooltip text. It is hidden by default, and will be visible on hover (see below). We have also added some basic styles to it: 120px width, black background color, white text color, centered text, and 5px top and bottom padding.

The CSS3 `border-radius` property is used to add rounded corners to the tooltip text.

The `:hover` selector is used to show the tooltip text when the user moves the mouse over the `<div>` with `class="tooltip"`.

Positioning Tooltips

In this example, the tooltip is placed to the right (`left:105%`) of the "hoverable" text (`<div>`). Also note that `top:-5px` is used to place it in the middle of its container element. We use the number **5** because the tooltip text has a top and bottom padding of 5px. If you increase its padding, also increase the value of the `top` property to ensure that it stays in the middle (if this is something you want). The same applies if you want the tooltip placed to the left.

Right Tooltip

```
.tooltip .tooltiptext {  
  top: -5px;  
  left: 105%;  
}
```

Left Tooltip

```
.tooltip .tooltiptext {  
  top: -5px;  
  right: 105%;  
}
```

If you want the tooltip to appear on top or on the bottom, see examples below. Note that we use the `margin-left` property with a value of minus 60 pixels. This is to center the tooltip above/below the hoverable text. It is set to the half of the tooltip's width ($120/2 = 60$).

Top Tooltip

```
.tooltip .tooltiptext {  
  width: 120px;  
  bottom: 100%;  
  left: 50%;  
  margin-left: -60px; /* Use half of the width (120/2 = 60), to center the tooltip */  
}
```

Bottom Tooltip

```
.tooltip .tooltiptext {  
  width: 120px;  
  top: 100%;  
  left: 50%;  
  margin-left: -60px; /* Use half of the width (120/2 = 60), to center the tooltip */  
}
```

Tooltip Arrows

To create an arrow that should appear from a specific side of the tooltip, add "empty" content after tooltip, with the pseudo-element class `::after` together with the `content` property. The arrow itself is created using borders. This will make the tooltip look like a speech bubble.

This example demonstrates how to add an arrow to the bottom of the tooltip:

Bottom Arrow

```
.tooltip .tooltiptext::after {  
  content: " ";  
  position: absolute;  
  top: 100%; /* At the bottom of the tooltip */  
}
```



```

left: 50%;
margin-left: -5px;
border-width: 5px;
border-style: solid;
border-color: black transparent transparent transparent;
}

```

Example Explained

Position the arrow inside the tooltip: `top: 100%` will place the arrow at the bottom of the tooltip. `left: 50%` will center the arrow.

Note: The `border-width` property specifies the size of the arrow. If you change this, also change the `margin-left` value to the same. This will keep the arrow centered.

The `border-color` is used to transform the content into an arrow. We set the top border to black, and the rest to transparent. If all sides were black, you would end up with a black square box.

This example demonstrates how to add an arrow to the top of the tooltip. Notice that we set the bottom border color this time:

Top Arrow

```

.tooltip .tooltiptext::after {
  content: " ";
  position: absolute;
  bottom: 100%; /* At the top of the tooltip */
  left: 50%;
  margin-left: -5px;
  border-width: 5px;
  border-style: solid;
  border-color: transparent transparent black transparent;
}

```

Result:

This example demonstrates how to add an arrow to the left of the tooltip:

Left Arrow

```

.tooltip .tooltiptext::after {
  content: " ";
  position: absolute;
  top: 50%;
  right: 100%; /* To the left of the tooltip */
  margin-top: -5px;
  border-width: 5px;
  border-style: solid;
  border-color: transparent black transparent transparent;
}

```

This example demonstrates how to add an arrow to the right of the tooltip:

Right Arrow

```
.tooltip .tooltiptext::after {  
  content: " ";  
  position: absolute;  
  top: 50%;  
  left: 100%; /* To the right of the tooltip */  
  margin-top: -5px;  
  border-width: 5px;  
  border-style: solid;  
  border-color: transparent transparent transparent black;  
}
```

Result:

Fade In Tooltips (Animation)

If you want to fade in the tooltip text when it is about to be visible, you can use the CSS3 `transition` property together with the `opacity` property, and go from being completely invisible to 100% visible, in a number of specified seconds (1 second in our example):

Example

```
.tooltip .tooltiptext {  
  opacity: 0;  
  transition: opacity 1s;  
}  
  
.tooltip:hover .tooltiptext {  
  opacity: 1;  
}
```

CSS can be used to create an image gallery.



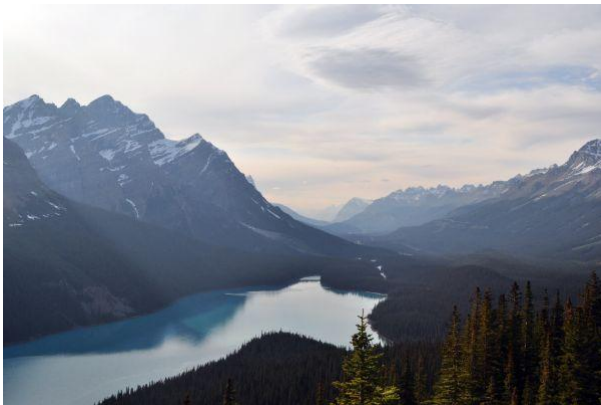
Add a description of the image here



Add a description of the image here



Add a description of the image here



Add a description of the image here

Image Gallery

The following image gallery is created with CSS:

Example

```
<html>  
<head>
```

```

<style>
div.img {
    margin: 5px;
    border: 1px solid #ccc;
    float: left;
    width: 180px;
}

div.img:hover {
    border: 1px solid #777;
}

div.img img {
    width: 100%;
    height: auto;
}

div.desc {
    padding: 15px;
    text-align: center;
}
</style>
</head>
<body>

<div class="img">
  <a target="_blank" href="fjords.jpg">
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="img">
  <a target="_blank" href="forest.jpg">
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="img">
  <a target="_blank" href="lights.jpg">
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="img">
  <a target="_blank" href="mountains.jpg">

```

```

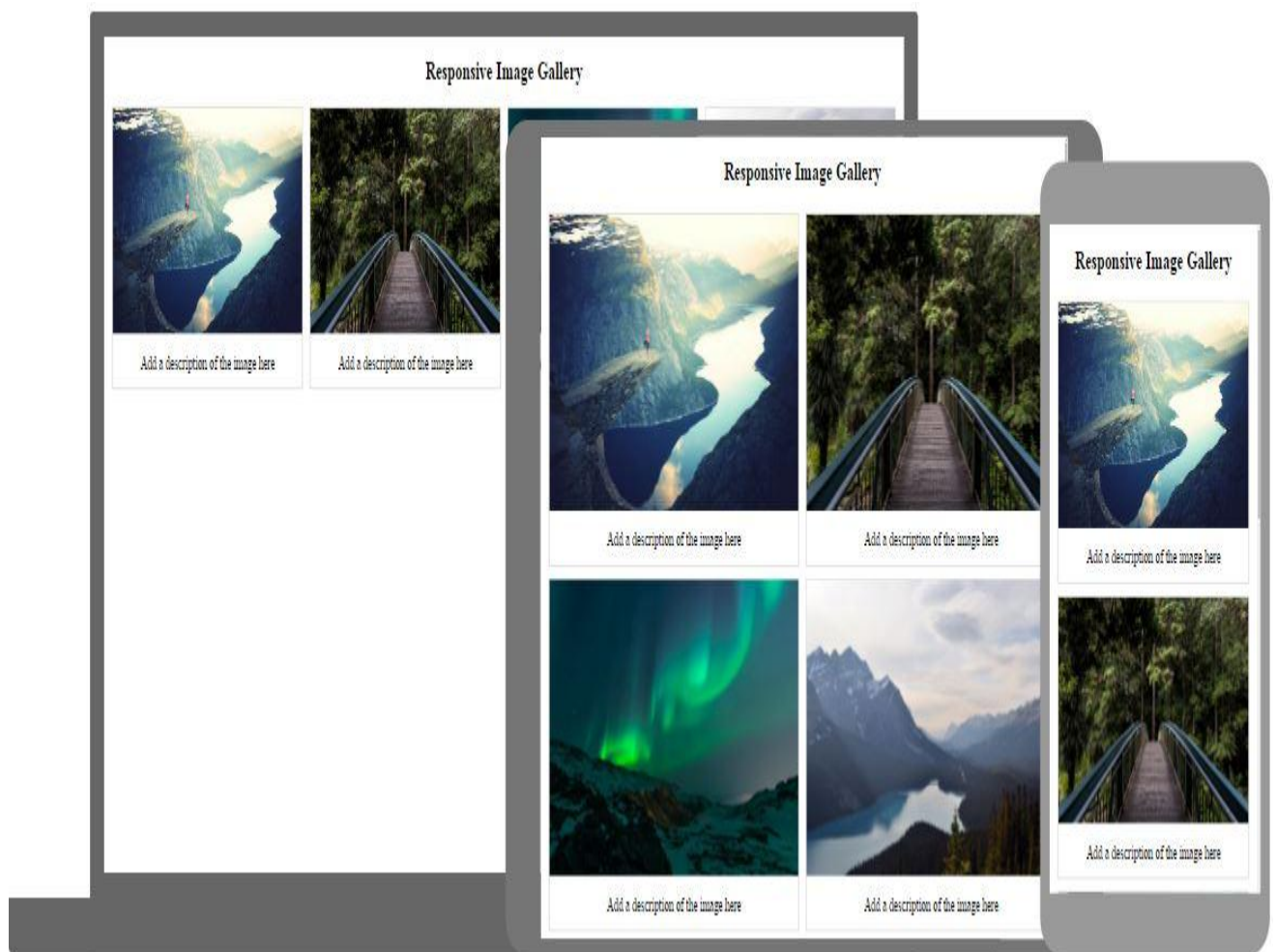
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

</body>
</html>

```

Responsive Image Gallery

How to use CSS3 media queries to create a responsive image gallery.



CSS Image Sprites

An image sprite is a collection of images put into a single image.

A web page with many images can take a long time to load and generates multiple server requests.

Using image sprites will reduce the number of server requests and save bandwidth.

Image Sprites - Simple Example

Instead of using three separate images, we use this single image ("img_navsprites.gif"):



With CSS, we can show just the part of the image we need.

In the following example the CSS specifies which part of the "img_navsprites.gif" image to show:

Example

```
#home {  
  width: 46px;  
  height: 44px;  
  background: url(img_navsprites.gif) 0 0;  
}
```

Example explained:

- `` - Only defines a small transparent image because the src attribute cannot be empty. The displayed image will be the background image we specify in CSS
- `width: 46px; height: 44px;` - Defines the portion of the image we want to use
- `background: url(img_navsprites.gif) 0 0;` - Defines the background image and its position (left opx, top opx)

This is the easiest way to use image sprites, now we want to expand it by using links and hover effects.

Image Sprites - Create a Navigation List

We want to use the sprite image ("img_navsprites.gif") to create a navigation list.

We will use an HTML list, because it can be a link and also supports a background image:

Example

```
#navlist {  
  position: relative;  
}  
  
#navlist li {  
  margin: 0;  
  padding: 0;  
  list-style: none;  
  position: absolute;  
  top: 0;  
}
```

```
#navlist li, #navlist a {
  height: 44px;
  display: block;
}

#home {
  left: 0px;
  width: 46px;
  background: url('img_navsprites.gif') 0 0;
}

#prev {
  left: 63px;
  width: 43px;
  background: url('img_navsprites.gif') -47px 0;
}

#next {
  left: 129px;
  width: 43px;
  background: url('img_navsprites.gif') -91px 0;
}
```

Example explained:

- #navlist {position:relative;} - position is set to relative to allow absolute positioning inside it
- #navlist li {margin:0;padding:0;list-style:none;position:absolute;top:0;} - margin and padding is set to 0, list-style is removed, and all list items are absolute positioned
- #navlist li, #navlist a {height:44px;display:block;} - the height of all the images are 44px

Now start to position and style for each specific part:

- #home {left:0px;width:46px;} - Positioned all the way to the left, and the width of the image is 46px
- #home {background:url(img_navsprites.gif) 0 0;} - Defines the background image and its position (left 0px, top 0px)
- #prev {left:63px;width:43px;} - Positioned 63px to the right (#home width 46px + some extra space between items), and the width is 43px.
- #prev {background:url('img_navsprites.gif') -47px 0;} - Defines the background image 47px to the right (#home width 46px + 1px line divider)
- #next {left:129px;width:43px;} - Positioned 129px to the right (start of #prev is 63px + #prev width 43px + extra space), and the width is 43px.
- #next {background:url('img_navsprites.gif') -91px 0;} - Defines the background image 91px to the right (#home width 46px + 1px line divider + #prev width 43px + 1px line divider)

Image Sprites - Hover Effect

Now we want to add a hover effect to our navigation list.

Tip: The `:hover` selector can be used on all elements, not only on links.

Our new image ("img_navsprites_hover.gif") contains three navigation images and three images to use for hover effects:



Because this is one single image, and not six separate files, there will be **no loading delay** when a user hovers over the image.

We only add three lines of code to add the hover effect:

Example

```
#home a:hover {  
  background: url('img_navsprites_hover.gif') 0 -45px;  
}  
  
#prev a:hover {  
  background: url('img_navsprites_hover.gif') -47px -45px;  
}  
  
#next a:hover {  
  background: url('img_navsprites_hover.gif') -91px -45px;  
}
```

Example explained:

- #home a:hover {background: transparent url('img_navsprites_hover.gif') 0 -45px;} - For all three hover images we specify the same background position, only 45px further down

CSS Attribute Selectors

Style HTML Elements

It is possible to style HTML elements that have specific attributes or attribute values.

CSS [attribute] Selector

The [attribute] selector is used to select elements with a specified attribute.

The following example selects all <a> elements with a target attribute:

Example


```
a[target] {  
  background-color: yellow;  
}
```

CSS [attribute="value"] Selector

The [attribute="value"] selector is used to select elements with a specified attribute and value.

The following example selects all <a> elements with a target="_blank" attribute:

Example

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

CSS [attribute~="value"] Selector

The [attribute~="value"] selector is used to select elements with an attribute value containing a specified word.

The following example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":

Example

```
[title~="flower"] {  
  border: 5px solid yellow;  
}
```

The example above will match elements with title="flower", title="summer flower", and title="flower new", but not title="my-flower" or title="flowers".

CSS [attribute|= "value"] Selector

The [attribute|= "value"] selector is used to select elements with the specified attribute starting with the specified value.

The following example selects all elements with a class attribute value that begins with "top":

Note: The value has to be a whole word, either alone, like class="top", or followed by a hyphen(-), like class="top-text"!

Example

```
[class|= "top"] {  
  background: yellow;  
}
```

CSS [attribute^="value"] Selector

The `[attribute^="value"]` selector is used to select elements whose attribute value begins with a specified value.

The following example selects all elements with a class attribute value that begins with "top":

Note: The value does not have to be a whole word!

Example

```
[class^="top"] {  
  background: yellow;  
}
```

CSS [attribute\$="value"] Selector

The `[attribute$="value"]` selector is used to select elements whose attribute value ends with a specified value.

The following example selects all elements with a class attribute value that ends with "test":

Note: The value does not have to be a whole word!

Example

```
[class$="test"] {  
  background: yellow;  
}
```

CSS [attribute*="value"] Selector

The `[attribute*="value"]` selector is used to select elements whose attribute value contains a specified value.

The following example selects all elements with a class attribute value that contains "te":

Note: The value does not have to be a whole word!

Example

```
[class*="te"] {  
  background: yellow;  
}
```

Styling Forms

The attribute selectors can be useful for styling forms without class or ID:

Example

```
input[type="text"] {  
  width: 150px;  
  display: block;  
  margin-bottom: 10px;  
  background-color: yellow;  
}  
  
input[type="button"] {  
  width: 120px;  
  margin-left: 35px;  
  display: block;  
}
```

The look of an HTML form can be greatly improved with CSS:

First Name Last Name State

Bottom of Form

Styling Input Fields

Use the `width` property to determine the width of the input field:

First Name

Example

```
input {  
  width: 100%;  
}
```

The example above applies to all `<input>` elements. If you only want to style a specific input type, you can use attribute selectors:

- `input[type=text]` - will only select text fields
- `input[type=password]` - will only select password fields
- `input[type=number]` - will only select number fields
- etc..

Padded Inputs

Use the `padding` property to add space inside the text field.

Tip: When you have many inputs after each other, you might also want to add some `margin`, to add more space outside of them:

First Name Last Name

Example

```
input[type=text] {  
  width: 100%;  
  padding: 12px 20px;  
  margin: 8px 0;  
  box-sizing: border-box;  
}
```

Note that we have set the `box-sizing` property to `border-box`. This makes sure that the padding and eventually borders are included in the total width and height of the elements.
Read more about the `box-sizing` property in our [CSS3 Box Sizing](#) chapter.

Bordered Inputs

Use the `border` property to change the border size and color, and use the `border-radius` property to add rounded corners:

First Name

Example

```
input[type=text] {  
  border: 2px solid red;  
  border-radius: 4px;  
}
```

If you only want a bottom border, use the `border-bottom` property:

First Name

Example

```
input[type=text] {  
  border: none;  
  border-bottom: 2px solid red;  
}
```

Colored Inputs

Use the `background-color` property to add a background color to the input, and the `color` property to change the text color:

Example

```
input[type=text] {  
  background-color: #3CBC8D;  
  color: white;  
}
```

Focused Inputs

By default, some browsers will add a blue outline around the input when it gets focus (clicked on). You can remove this behavior by adding `outline: none;` to the input.

Use the `:focus` selector to do something with the input field when it gets focus:



Example

```
input[type=text]:focus {  
  background-color: lightblue;  
}
```



Example

```
input[type=text]:focus {  
  border: 3px solid #555;  
}
```

Input with icon/image

If you want an icon inside the input, use the `background-image` property and position it with the `background-position` property. Also notice that we add a large left padding to reserve the space of the icon:



Example

```
input[type=text] {  
  background-color: white;  
  background-image: url('searchicon.png');  
  background-position: 10px 10px;  
  background-repeat: no-repeat;  
  padding-left: 40px;  
}
```

Animated Search Input

In this example we use the CSS3 `transition` property to animate the width of the search input when it gets focus. You will learn more about the `transition` property later, in our [CSS3 Transitions](#) chapter.



Example

```
input[type=text] {  
  -webkit-transition: width 0.4s ease-in-out;  
  transition: width 0.4s ease-in-out;  
}
```

```
input[type=text]:focus {  
  width: 100%;  
}
```

Styling Textareas

Tip: Use the `resize` property to prevent textareas from being resized (disable the "grabber" in the bottom right corner):



Example

```
textarea {  
  width: 100%;  
  height: 150px;  
  padding: 12px 20px;  
  box-sizing: border-box;  
  border: 2px solid #ccc;  
  border-radius: 4px;  
  background-color: #f8f8f8;  
  resize: none;  
}
```

Styling Select Menus

Example

```
select {  
  width: 100%;  
  padding: 16px 20px;  
  border: none;  
  border-radius: 4px;
```

```
background-color: #f1f1f1;
}
```

Styling Input Buttons

Example

```
input[type=button], input[type=submit], input[type=reset] {
  background-color: #4CAF50;
  border: none;
  color: white;
  padding: 16px 32px;
  text-decoration: none;
  margin: 4px 2px;
  cursor: pointer;
}
```

/ Tip: use **width: 100%** for full-width buttons */*

CSS Counters

CSS counters are "variables" maintained by CSS whose values can be incremented by CSS rules (to track how many times they are used). Counters let you adjust the appearance of content based on its placement in the document.

Automatic Numbering With Counters

CSS counters are like "variables". The variable values can be incremented by CSS rules (which will track how many times they are used).

To work with CSS counters we will use the following properties:

- `counter-reset` - Creates or resets a counter
- `counter-increment` - Increments a counter value
- `content` - Inserts generated content
- `counter()` or `counters()` function - Adds the value of a counter to an element

To use a CSS counter, it must first be created with `counter-reset`.

The following example creates a counter for the page (in the body selector), then increments the counter value for each `<h2>` element and adds "Section *<value of the counter>*:" to the beginning of each `<h2>` element:

Example

```
body {
  counter-reset: section;
}
```

```
h2::before {
```

```

    counter-increment: section;
    content: "Section " counter(section) ": ";
}

```

Nesting Counters

The following example creates one counter for the page (section) and one counter for each <h1> element (subsection). The "section" counter will be counted for each <h1> element with "Section <value of the section counter>.", and the "subsection" counter will be counted for each <h2> element with "<value of the section counter>.<value of the subsection counter>":

Example

```

body {
    counter-reset: section;
}

h1 {
    counter-reset: subsection;
}

h1::before {
    counter-increment: section;
    content: "Section " counter(section) ". ";
}

h2::before {
    counter-increment: subsection;
    content: counter(section) "." counter(subsection) " ";
}

```

A counter can also be useful to make outlined lists because a new instance of a counter is automatically created in child elements. Here we use the `counters()` function to insert a string between different levels of nested counters:

Example

```

ol {
    counter-reset: section;
    list-style-type: none;
}

li::before {
    counter-increment: section;
    content: counters(section, ".") " ";
}

```


CSS Counter Properties

Property	Description
<u>content</u>	Used with the ::before and ::after pseudo-elements, to insert generated content
<u>counter-increment</u>	Increments one or more counter values
<u>counter-reset</u>	Creates or resets one or more counters

Chapter 2 vision three

CSS3 Introduction

CSS3 is the latest standard for CSS.

CSS3 is completely backwards-compatible with earlier versions of CSS.

This section teaches you about the new features in CSS3!

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".

Browser Support

The numbers in the table specify the first browser version that fully supports the property.

Numbers followed by `-webkit-` or `-moz-` specify the first version that worked with a prefix.

Numbers followed by `-webkit-` or `-moz-` specify the first version that worked with a prefix.

Property

	5.0	9.0	4.0	5.0	
<code>border-radius</code>	4.0 <code>-webkit-</code>		3.0 <code>-moz-</code>	3.1 <code>-webkit-</code>	10.5

CSS3 Rounded Corners

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

CSS3 `border-radius` Property

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

Here are three examples:

1. Rounded corners for an element with a specified background color:

Rounded corners!

2. Rounded corners for an element with a border:

Rounded corners!

3. Rounded corners for an element with a background image:

Rounded corners!

Here is the code:

Example

```
#rcorners1 {
  border-radius: 25px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}

#rcorners2 {
  border-radius: 25px;
  border: 2px solid #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}

#rcorners3 {
  border-radius: 25px;
  background: url(paper.gif);
  background-position: left top;
  background-repeat: repeat;
  padding: 20px;
  width: 200px;
  height: 150px;
}
```

Tip: The `border-radius` property is actually a shorthand property for the `border-top-left-radius`, `border-top-right-radius`, `border-bottom-right-radius` and `border-bottom-left-radius` properties.

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;`

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

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

Here is the code:

Example

```
#rcorners4 {  
  border-radius: 15px 50px 30px 5px;  
  background: #73AD21;  
  padding: 20px;  
  width: 200px;  
  height: 150px;  
}
```

```
#rcorners5 {  
  border-radius: 15px 50px 30px;  
  background: #73AD21;  
  padding: 20px;  
  width: 200px;  
  height: 150px;  
}
```

```
#rcorners6 {  
  border-radius: 15px 50px;  
  background: #73AD21;  
  padding: 20px;  
  width: 200px;  
  height: 150px;  
}
```

You could also create elliptical corners:

Example

```
#rcorners7 {  
  border-radius: 50px/15px;  
  background: #73AD21;  
  padding: 20px;  
  width: 200px;  
  height: 150px;  
}
```

```
#rcorners8 {  
  border-radius: 15px/50px;  
  background: #73AD21;  
  padding: 20px;  
  width: 200px;  
}
```

```

    height: 150px;
}

#rcorners9 {
    border-radius: 50%;
    background: #73AD21;
    padding: 20px;
    width: 200px;
    height: 150px;
}

```

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

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:

An image as a border!

Here is the code:

Example

```

#borderimg {
    border: 10px solid transparent;

```

```
padding: 15px;
-webkit-border-image: url(border.png) 30 round; /* Safari 3.1-5 */
-o-border-image: url(border.png) 30 round; /* Opera 11-12.1 */
border-image: url(border.png) 30 round;
}
```

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

An image as a border!

Here is the code:

Example

```
#borderimg {
border: 10px solid transparent;
padding: 15px;
-webkit-border-image: url(border.png) 30 stretch; /* Safari 3.1-5 */
-o-border-image: url(border.png) 30 stretch; /* Opera 11-12.1 */
border-image: url(border.png) 30 stretch;
}
```

Tip: The `border-image` property is actually a shorthand property for the `border-image-source`, `border-image-slice`, `border-image-width`, `border-image-outset` and `border-image-repeat` properties.

CSS3 border-image - Different Slice Values

Different slice values completely changes the look of the border:

Example 1:

```
border-image: url(border.png) 50 round;
```

Example 2:

```
border-image: url(border.png) 20% round;
```

Example 3:

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

Here is the code:

Example

```
#borderimg1 {
border: 10px solid transparent;
padding: 15px;
-webkit-border-image: url(border.png) 50 round; /* Safari 3.1-5 */
```

```

    -o-border-image: url(border.png) 50 round; /* Opera 11-12.1 */
    border-image: url(border.png) 50 round;
}

#borderimg2 {
    border: 10px solid transparent;
    padding: 15px;
    -webkit-border-image: url(border.png) 20% round; /* Safari 3.1-5 */
    -o-border-image: url(border.png) 20% round; /* Opera 11-12.1 */
    border-image: url(border.png) 20% round;
}

#borderimg3 {
    border: 10px solid transparent;
    padding: 15px;
    -webkit-border-image: url(border.png) 30% round; /* Safari 3.1-5 */
    -o-border-image: url(border.png) 30% round; /* Opera 11-12.1 */
    border-image: url(border.png) 30% round;
}

```

Edit

```

<!DOCTYPE html>

<html>

<head>

<style>

#borderimg1 {

    border: 10px solid transparent;

    padding: 15px;

    -webkit-border-image: url(border.png) 50 round; /* Safari 3.1-5 */

    -o-border-image: url(border.png) 50 round; /* Opera 11-12.1 */

    border-image: url(border.png) 50 round;

}

#borderimg2 {

    border: 10px solid transparent;

    padding: 15px;

    -webkit-border-image: url(border.png) 20% round; /* Safari 3.1-5 */

```

```

-o-border-image: url(border.png) 20% round; /* Opera 11-12.1 */

border-image: url(border.png) 20% round;

}

#bordering3 {

border: 10px solid transparent;

padding: 15px;

-webkit-border-image: url(border.png) 30% round; /* Safari 3.1-5 */

-o-border-image: url(border.png) 30% round; /* Opera 11-12.1 */

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

}

</style>

</head>

<body>

<p id="bordering1">border-image: url(border.png) 50 round;</p>

<p id="bordering2">border-image: url(border.png) 20% round;</p>

<p id="bordering3">border-image: url(border.png) 30% round;</p>

<p><strong>Note:</strong> Internet Explorer 10, and earlier versions, do not support the border-image property.</p>

</body>

</html>

```

Demo

border-image: url(border.png) 20% round

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

Note: Internet Explorer 10, and earlier versions, do not support the border-image property.

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`
- `background-origin`
- `background-clip`

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):

Example

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

Multiple background images can be specified using either the individual background properties (as above) or the `background` shorthand property.

The following example uses the `background` shorthand property (same result as example above):

Example

```
#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.

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):

Original background image:

Lorem Ipsum Dolor

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.

Resized background image:

Lorem Ipsum Dolor

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.

Here is the code:

Example

```
#div1 {  
  background: url(img_flower.jpg);  
  background-size: 100px 80px;  
  background-repeat: no-repeat;  
}
```

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.

The following example illustrates the use of `contain` and `cover`:

Example

```
#div1 {  
  background: url(img_flower.jpg);  
  background-size: contain;  
}
```

```

    background-repeat: no-repeat;
}
#div2 {
    background: url(img_flower.jpg);
    background-size: cover;
    background-repeat: no-repeat;
}

```

Define Sizes of Multiple Background Images

The `background-size` property also accepts multiple values for background size (using a comma-separated list), when working with multiple backgrounds.

The following example has three background images specified, with different background-size value for each image:

Example

```

#example1 {
    background: url(img_flwr.gif) left top no-repeat, url(img_flwr.gif) right bottom no-repeat, url(paper.gif)
left top repeat;
    background-size: 50px, 130px, auto;
}

```

Full Size Background Image

Now we want to have a background image on a website that covers the entire browser window at all times.

The requirements are as follows:

- Fill the entire page with the image (no white space)
- Scale image as needed
- Center image on page
- Do not cause scrollbars

The following example shows how to do it; Use the `html` element (the `html` element is always at least the height of the browser window). Then set a fixed and centered background on it. Then adjust its size with the `background-size` property:

Example

```

html {
    background: url(img_flower.jpg) no-repeat center fixed;
    background-size: cover;
}

```

CSS3 background-origin Property

The CSS3 `background-origin` property specifies where the background image is positioned.

The property takes three different values:

- border-box - the background image starts from the upper left corner of the border
- padding-box - (default) the background image starts from the upper left corner of the padding edge
- content-box - the background image starts from the upper left corner of the content

The following example illustrates the `background-origin` property:

Example

```
#example1 {  
  border: 10px solid black;  
  padding: 35px;  
  background: url(img_flwr.gif);  
  background-repeat: no-repeat;  
  background-origin: content-box;  
}
```

CSS3 background-clip Property

The CSS3 `background-clip` property specifies the painting area of the background.

The property takes three different values:

- border-box - (default) the background is painted to the outside edge of the border
- padding-box - the background is painted to the outside edge of the padding
- content-box - the background is painted within the content box

The following example illustrates the `background-clip` property:

Example

```
#example1 {  
  border: 10px dotted black;  
  padding: 35px;  
  background: yellow;  
  background-clip: content-box;  
}
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
#example1 {
```

```
  border: 10px dotted black;
```

```

padding:35px;

background: yellow;

}

#example2 {

border: 10px dotted black;

padding:35px;

background: yellow;

background-clip: padding-box;

}

#example3 {

border: 10px dotted black;

padding:35px;

background: yellow;

background-clip: content-box;

}

</style>

</head>

<body>

<p>No background-clip (border-box is default):</p>

<div id="example1">

<h2>Lorem Ipsum Dolor</h2>

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

</div>

<p>background-clip: padding-box:</p>

<div id="example2">

<h2>Lorem Ipsum Dolor</h2>

```

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

</div>

<p>background-clip: content-box;</p>

<div id="example3">

<h2>Lorem Ipsum Dolor</h2>

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

</div>

</body>

</html>

Demo

No background-clip (border-box is default)

Lorem Ipsum Dolor

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

background-clip: padding-box

Lorem Ipsum Dolor

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

background-clip: content-box:

Lorem Ipsum Dolor

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

CSS3 Colors

CSS supports color names, hexadecimal and RGB colors.

In addition, CSS3 also introduces:

- RGBA colors
- HSL colors
- HSLA colors

Lorem Ipsum Dolor

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

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 (fully opaque).

```
rgba(255, 0, 0, 0.2);
```

```
rgba(255, 0, 0, 0.4);
```

```
rgba(255, 0, 0, 0.6);
```

```
rgba(255, 0, 0, 0.8);
```

The following example defines different RGBA colors:

Example

```
#p1 {background-color: rgba(255, 0, 0, 0.3);} /* red with opacity */  
#p2 {background-color: rgba(0, 255, 0, 0.3);} /* green with opacity */  
#p3 {background-color: rgba(0, 0, 255, 0.3);} /* blue with opacity */
```



HSL Colors

HSL stands for Hue, Saturation and Lightness.

An HSL color value is specified with: `hsl(hue, saturation, lightness)`.

1. Hue is a degree on the color wheel (from 0 to 360):
 - 0 (or 360) is red
 - 120 is green
 - 240 is blue
2. Saturation is a percentage value: 100% is the full color.
3. Lightness is also a percentage; 0% is dark (black) and 100% is white.

`hsl(0, 100%, 30%);`

`hsl(0, 100%, 50%);`

`hsl(0, 100%, 70%);`

`hsl(0, 100%, 90%);`

The following example defines different HSL colors:

Example

```
#p1 {background-color: hsl(120, 100%, 50%);} /* green */
#p2 {background-color: hsl(120, 100%, 75%);} /* light green */
#p3 {background-color: hsl(120, 100%, 25%);} /* dark green */
#p4 {background-color: hsl(120, 60%, 70%);} /* pastel green */
```

HSLA Colors

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with: `hsla(hue, saturation, lightness, alpha)`, where the alpha parameter defines the opacity. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

`hsla(0, 100%, 30%, 0.3);`

`hsla(0, 100%, 50%, 0.3);`

`hsla(0, 100%, 70%, 0.3);`

`hsla(0, 100%, 90%, 0.3);`

The following example defines different HSLA colors:

Example

```
#p1 {background-color: hsla(120, 100%, 50%, 0.3);} /* green with opacity */
#p2 {background-color: hsla(120, 100%, 75%, 0.3);} /* light green with opacity */
#p3 {background-color: hsla(120, 100%, 25%, 0.3);} /* dark green with opacity */
#p4 {background-color: hsla(120, 60%, 70%, 0.3);} /* pastel green with opacity */
```

Opacity

The CSS3 `opacity` property sets the opacity for the whole element (both background color and text will be opaque/transparent).

The `opacity` property value must be a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
rgb(255, 0, 0);opacity:0.2;
rgb(255, 0, 0);opacity:0.4;
rgb(255, 0, 0);opacity:0.6;
rgb(255, 0, 0);opacity:0.8;
```

Notice that the text above will also be transparent/opaque!

The following example shows different elements with opacity:

Example

```
#p1 {background-color:rgb(255,0,0);opacity:0.6;} /* red with opacity */
#p2 {background-color:rgb(0,255,0);opacity:0.6;} /* green with opacity */
#p3 {background-color:rgb(0,0,255);opacity:0.6;} /* blue with opacity */
```

Edit

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
#p1 {background-color:rgb(255,0,0);opacity:0.6;}
```

```
#p2 {background-color:rgb(0,255,0);opacity:0.6;}
```

```
#p3 {background-color:rgb(0,0,255);opacity:0.6;}
```

```
#p4 {background-color:rgb(192,192,192);opacity:0.6;}
```

```
#p5 {background-color:rgb(255,255,0);opacity:0.6;}
```

```
#p6 {background-color:rgb(255,0,255);opacity:0.6;}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<p>Elements with opacity:</p>
```

```
<p id="p1">Red</p>
```

```
<p id="p2">Green</p>
```

```
<p id="p3">Blue</p>
```

```
<p id="p4">Grey</p>
```

```
<p id="p5">Yellow</p>
```

```
<p id="p6">Cerise</p>
```

```
</body>
```

```
</html>
```


Elements with opacity:

Red 

Green 

Blue 

Grey 

Yellow 

Cerise 

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, ...);`

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:

Example

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

Linear Gradient - Left to Right

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

Example

```
#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:

Example

```
#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);
```

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:

Example

```
#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:

Example

```
#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 */
}
```

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

Gradient Background

Example

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

Using Transparency

CSS3 gradients also support transparency, which can be used to create fading effects.

To add transparency, we use the `rgba()` function to define the color stops. The last parameter in the `rgba()` function can be a value from 0 to 1, and it defines the transparency of the color: 0 indicates full transparency, 1 indicates full color (no transparency).

The following example shows a linear gradient that starts from the left. It starts fully transparent, transitioning to full color red:

Example

```
#grad {
  background: red; /* For browsers that do not support gradients */
  background: -webkit-linear-gradient(left,rgba(255,0,0,0),rgba(255,0,0,1)); /*Safari 5.1-6*/
  background: -o-linear-gradient(right,rgba(255,0,0,0),rgba(255,0,0,1)); /*Opera 11.1-12*/
  background: -moz-linear-gradient(right,rgba(255,0,0,0),rgba(255,0,0,1)); /*Fx 3.6-15*/
  background: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1)); /*Standard*/
}
```

Repeating a linear-gradient

The `repeating-linear-gradient()` function is used to repeat linear gradients:

Example

A repeating linear gradient:

```
#grad {  
  background: red; /* For browsers that do not support gradients */  
  /* Safari 5.1 to 6.0 */  
  background: -webkit-repeating-linear-gradient(red, yellow 10%, green 20%);  
  /* Opera 11.1 to 12.0 */  
  background: -o-repeating-linear-gradient(red, yellow 10%, green 20%);  
  /* Firefox 3.6 to 15 */  
  background: -moz-repeating-linear-gradient(red, yellow 10%, green 20%);  
  /* Standard syntax */  
  background: repeating-linear-gradient(red, yellow 10%, green 20%);  
}
```

CSS3 Radial Gradients

A radial gradient is defined by its center.

To create a radial gradient you must also define at least two color stops.

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.

Radial Gradient - Evenly Spaced Color Stops (this is default)

The following example shows a radial gradient with evenly spaced color stops:

Example

```
#grad {  
  background: red; /* For browsers that do not support gradients */  
  background: -webkit-radial-gradient(red, yellow, green); /* Safari 5.1 to 6.0 */  
  background: -o-radial-gradient(red, yellow, green); /* For Opera 11.6 to 12.0 */  
  background: -moz-radial-gradient(red, yellow, green); /* For Firefox 3.6 to 15 */  
  background: radial-gradient(red, yellow, green); /* Standard syntax */  
}
```

Radial Gradient - Differently Spaced Color Stops

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

Example

```
#grad {  
  background: red; /* For browsers that do not support gradients */  
  background: -webkit-radial-gradient(red 5%, yellow 15%, green 60%); /* Safari 5.1-6.0 */  
  background: -o-radial-gradient(red 5%, yellow 15%, green 60%); /* For Opera 11.6-12.0 */  
  background: -moz-radial-gradient(red 5%, yellow 15%, green 60%); /* For Firefox 3.6-15 */  
  background: radial-gradient(red 5%, yellow 15%, green 60%); /* Standard syntax */  
}
```

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:

Example

```
#grad {  
  background: red; /* For browsers that do not support gradients */  
  background: -webkit-radial-gradient(circle, red, yellow, green); /* Safari */  
  background: -o-radial-gradient(circle, red, yellow, green); /* Opera 11.6 to 12.0 */  
  background: -moz-radial-gradient(circle, red, yellow, green); /* Firefox 3.6 to 15 */  
  background: radial-gradient(circle, red, yellow, green); /* Standard syntax */  
}
```

Use of Different Size Keywords

The size parameter defines the size of the gradient. It can take four values:

- **closest-side**
- **farthest-side**
- **closest-corner**
- **farthest-corner**

Example

A radial gradient with different size keywords:

```
#grad1 {  
  background: red; /* For browsers that do not support gradients */  
  /* Safari 5.1 to 6.0 */  
  background: -webkit-radial-gradient(60% 55%, closest-side, red, yellow, black);  
  /* For Opera 11.6 to 12.0 */  
  background: -o-radial-gradient(60% 55%, closest-side, red, yellow, black);  
  /* For Firefox 3.6 to 15 */  
  background: -moz-radial-gradient(60% 55%, closest-side, red, yellow, black);  
  /* Standard syntax */  
  background: radial-gradient(closest-side at 60% 55%, red, yellow, black);  
}
```

```
#grad2 {
  /* Safari 5.1 to 6.0 */
  background: -webkit-radial-gradient(60% 55%, farthest-side, red, yellow, black);
  /* Opera 11.6 to 12.0 */
  background: -o-radial-gradient(60% 55%, farthest-side, red, yellow, black);
  /* For Firefox 3.6 to 15 */
  background: -moz-radial-gradient(60% 55%, farthest-side, red, yellow, black);
  /* Standard syntax */
  background: radial-gradient(farthest-side at 60% 55%, red, yellow, black);
}
```

Repeating a radial-gradient

The repeating-radial-gradient() function is used to repeat radial gradients:

Example

A repeating radial gradient:

```
#grad {
  background: red; /* For browsers that do not support gradients */
  /* For Safari 5.1 to 6.0 */
  background: -webkit-repeating-radial-gradient(red, yellow 10%, green 15%);
  /* For Opera 11.6 to 12.0 */
  background: -o-repeating-radial-gradient(red, yellow 10%, green 15%);
  /* For Firefox 3.6 to 15 */
  background: -moz-repeating-radial-gradient(red, yellow 10%, green 15%);
  /* Standard syntax */
  background: repeating-radial-gradient(red, yellow 10%, green 15%);
}
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
#grad1 {
```

```
  height: 150px;
```

```
  width: 200px;
```

```
  background: red; /* For browsers that do not support gradients */
```



```
background: -webkit-radial-gradient(red, yellow, green); /* Safari 5.1 to 6.0 */

background: -o-radial-gradient(red, yellow, green); /* For Opera 11.6 to 12.0 */

background: -moz-radial-gradient(red, yellow, green); /* For Firefox 3.6 to 15 */

background: radial-gradient(red, yellow, green); /* Standard syntax (must be last) */

}

</style>

</head>

<body>

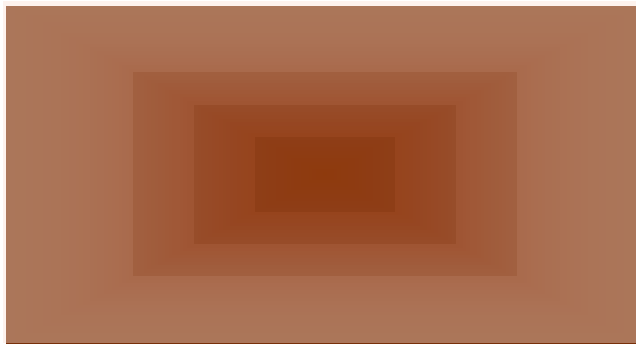
<h3>Radial Gradient - Evenly Spaced Color Stops</h3>

<div id="grad1"></div>

<p><strong>Note:</strong> Internet Explorer 9 and earlier versions do not support gradients.</p>

</body>

</html>
```



CSS3 Shadow Effects



Box Shadow

With CSS3 you can create shadow effects!

Hover over me!

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 Shadow Effects



Box Shadow

With CSS3 you can create shadow effects!

Hover over me!

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):

Text shadow effect!

Example

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

Next, add a color to the shadow:

Text shadow effect!

Example

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

Then, add a blur effect to the shadow:

Text shadow effect!

Example

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

The following example shows a white text with black shadow:

Text shadow effect!

Example

```
h1 {
  color: white;
  text-shadow: 2px 2px 4px #000000;
}
```

The following example shows a red neon glow shadow:

Text shadow effect!

Example

```
h1 {
  text-shadow: 0 0 3px #FF0000;
}
```

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 red and blue neon glow shadow:

Text shadow effect!

Example

```
h1 {
  text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;
}
```

The following example shows a white text with black, blue, and darkblue shadow:

Text shadow effect!

Example

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

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:

This is a yellow `<div>` element with a black box-shadow

Example

```
div {  
  box-shadow: 10px 10px;  
}
```

Next, add a color to the shadow:

This is a yellow <div> element with a grey box-shadow

Example

```
div {  
  box-shadow: 10px 10px grey;  
}
```

Next, add a blur effect to the shadow:

This is a yellow <div> element with a blurred, grey box-shadow

Example

```
div {  
  box-shadow: 10px 10px 5px grey;  
}
```

You can also add shadows to the ::before and ::after pseudo-classes, to create an interesting effect:

Example

```
#boxshadow {  
  position: relative;  
  box-shadow: 1px 2px 4px rgba(0, 0, 0, .5);  
  padding: 10px;  
  background: white;  
}  
#boxshadow img {  
  width: 100%;  
  border: 1px solid #8a4419;  
  border-style: inset;  
}  
  
#boxshadow::after {  
  content: "";  
  position: absolute;  
  z-index: -1; /* hide shadow behind image */  
  box-shadow: 0 15px 20px rgba(0, 0, 0, 0.3);  
  width: 70%;  
  left: 15%; /* one half of the remaining 30% */  
  height: 100px;  
  bottom: 0;  
}
```

Cards

An example of using the `box-shadow` property to create paper-like cards:



Hardanger, Norway

Example

```
div.card {  
  width: 250px;  
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);  
  text-align: center;  
}
```

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.

It can be clipped:

This is some long text that will not fit in the box

or it can be rendered as an ellipsis (...):

This is some long text that will not fit in the box

The CSS code is as follows:

Example

```
p.test1 {  
  white-space: nowrap;  
  width: 200px;  
  border: 1px solid #000000;  
  overflow: hidden;  
  text-overflow: clip;  
}
```

```
p.test2 {  
  white-space: nowrap;  
  width: 200px;  
  border: 1px solid #000000;  
  overflow: hidden;  
  text-overflow: ellipsis;  
}
```

The following example shows how you can display the overflowed content when hovering over the element:

Example

```
div.test:hover {  
  text-overflow: inherit;  
  overflow: visible;  
}
```

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:

This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

The word-wrap property allows you to force the text to wrap - even if it means splitting it in the middle of a word:

This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

The CSS code is as follows:

Example

Allow long words to be able to be broken and wrap onto the next line:

```
p {  
  word-wrap: break-word;  
}
```

CSS3 Word Breaking

The CSS3 `word-break` property specifies line breaking rules.

This paragraph contains some text. This line will-break-at-hyphens.

This paragraph contains some text. The lines will break at any character.

The CSS code is as follows:

Example

```
p.test1 {  
  word-break: keep-all;  
}
```

```
p.test2 {  
  word-break: break-all;  
}
```

CSS3 Web Fonts

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 defines 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.

*IE: The font format only works when set to be "installable".

*Firefox: Not supported by default, but can be enabled (need to set a flag to "true" to use WOFF2).

The Font You Want

In the CSS3 `@font-face` rule you must first define a name for the font (e.g. `myFirstFont`), and then point to the font file.

Tip: Always use lowercase letters for the font URL. Uppercase letters can give unexpected results in IE.

To use the font for an HTML element, refer to the name of the font (`myFirstFont`) through the `font-family` property:

Example

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

```
div {  
  font-family: myFirstFont;  
}
```

Using Bold Text

You must add another `@font-face` rule containing descriptors for bold text:

Example

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

The file "sansation_bold.woff" is another font file, that contains the bold characters for the Sansation font.

Browsers will use this whenever a piece of text with the font-family "myFirstFont" should render as bold.

This way you can have many `@font-face` rules for the same font.

CSS3 2D 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.

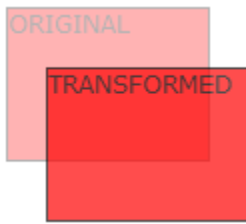
Mouse over the elements below to see the difference between a 2D and a 3D transformation:

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

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

Tip: You will learn about 3D transformations in the next chapter.

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:

Example

```
div {  
  -ms-transform: translate(50px, 100px); /* IE 9 */  
  -webkit-transform: translate(50px, 100px); /* Safari */  
  transform: translate(50px, 100px);  
}
```

The rotate() Method



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:

Example

```
div {  
  -ms-transform: rotate(20deg); /* IE 9 */  
  -webkit-transform: rotate(20deg); /* Safari */  
  transform: rotate(20deg);  
}
```

Using negative values will rotate the element counter-clockwise.

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

Example

```
div {
  -ms-transform: rotate(-20deg); /* IE 9 */
  -webkit-transform: rotate(-20deg); /* Safari */
  transform: rotate(-20deg);
}
```

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:

Example

```
div {
  -ms-transform: scale(2, 3); /* IE 9 */
  -webkit-transform: scale(2, 3); /* Safari */
  transform: scale(2, 3);
}
```

The following example decreases the `<div>` element to be half of its original width and height:

Example

```
div {
  -ms-transform: scale(0.5, 0.5); /* IE 9 */
  -webkit-transform: scale(0.5, 0.5); /* Safari */
  transform: scale(0.5, 0.5);
}
```

The skewX() Method

The `skewX()` method skews an element along the X-axis by the given angle.

The following example skews the `<div>` element 20 degrees along the X-axis:

Example

```
div {
  -ms-transform: skewX(20deg); /* IE 9 */
  -webkit-transform: skewX(20deg); /* Safari */
}
```

```
transform: skewX(20deg);  
}
```

The skewY() Method

The `skewY()` method skews an element along the Y-axis by the given angle.

The following example skews the `<div>` element 20 degrees along the Y-axis:

Example

```
div {  
  -ms-transform: skewY(20deg); /* IE 9 */  
  -webkit-transform: skewY(20deg); /* Safari */  
  transform: skewY(20deg);  
}
```

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:

Example

```
div {  
  -ms-transform: skew(20deg, 10deg); /* IE 9 */  
  -webkit-transform: skew(20deg, 10deg); /* Safari */  
  transform: skew(20deg, 10deg);  
}
```

If the second parameter is not specified, it has a zero value. So, the following example skews the `<div>` element 20 degrees along the X-axis:

Example

```
div {  
  -ms-transform: skew(20deg); /* IE 9 */  
  -webkit-transform: skew(20deg); /* Safari */  
  transform: skew(20deg);  
}
```

The matrix() Method



The `matrix()` method combines all the 2D transform methods into one.

The `matrix()` method takes six parameters, containing mathematical functions, which allows you to rotate, scale, move (translate), and skew elements.

The parameters are as follows: `matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())`:

Example

```
div {  
  -ms-transform: matrix(1, -0.3, 0, 1, 0, 0); /* IE 9 */  
  -webkit-transform: matrix(1, -0.3, 0, 1, 0, 0); /* Safari */  
  transform: matrix(1, -0.3, 0, 1, 0, 0);  
} <!DOCTYPE html>  
  
<html>  
<head>  
<style>  
div {  
  width: 300px;  
  height: 100px;  
  background-color: yellow;  
  border: 1px solid black;  
}  
div#myDiv1 {  
  -ms-transform: matrix(1, -0.3, 0, 1, 0, 0); /* IE 9 */  
  -webkit-transform: matrix(1, -0.3, 0, 1, 0, 0); /* Safari */  
  transform: matrix(1, -0.3, 0, 1, 0, 0); /* Standard syntax */  
}
```

```

div#myDiv2 {
    -ms-transform: matrix(1, 0, 0.5, 1, 150, 0); /* IE 9 */
    -webkit-transform: matrix(1, 0, 0.5, 1, 150, 0); /* Safari */
    transform: matrix(1, 0, 0.5, 1, 150, 0); /* Standard syntax */
}

</style>
</head>
<body>

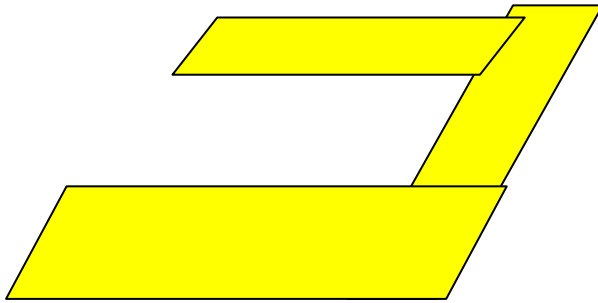
<p>The matrix() method combines all the 2D transform methods into one.</p>

<div>
This a normal div element.
</div>
<div id="myDiv1">
Using the matrix() method.
</div>
<div id="myDiv2">
Another use of the matrix() method.
</div>
</body>
</html>

```

he matrix() method combines all the 2D transform methods into one.

.



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:

2D rotate

3D rotate

Browser Support for 3D Transforms

The numbers in the table specify the first browser version that fully supports the property.

Numbers followed by -webkit-, -moz-, or -o- specify the first version that worked with a prefix.

CSS3 3D Transforms

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

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

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:

Example

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

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
div {
```

```
  width: 300px;
```

```
  height: 100px;
```

```
  background-color: yellow;
```

```
  border: 1px solid black;
```

```
}
```

```

div#myDiv {

    -webkit-transform: rotateZ(90deg); /* Safari */

    transform: rotateZ(90deg); /* Standard syntax */

}

</style>

</head>

<body>

<div>

This a normal div element.

</div>

<div id="myDiv">

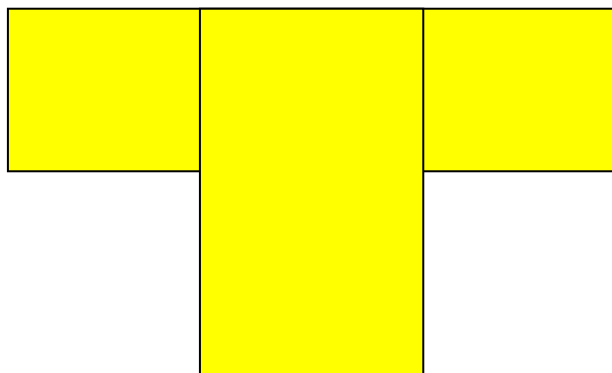
The rotateZ() method rotates an element around its Z-axis at a given degree. This div element is rotated 90 degrees.

</div>

<p><b>Note:</b> Internet Explorer 9 (and earlier versions) does not support the rotateZ() method.</p></body>

</html>

```



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:

How to Use CSS3 Transitions?

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

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

The following example shows a 100px * 100px red <div> element. The <div> element has also specified a transition effect for the width property, with a duration of 2 seconds:

Example

```
div {  
  width: 100px;  
  height: 100px;  
  background: red;  
  -webkit-transition: width 2s; /* Safari */  
  transition: width 2s;  
}
```

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

Now, let us specify a new value for the width property when a user mouses over the <div> element:

Example

```
div:hover {  
  width: 300px;  
}
```

Notice that when the cursor mouses out of the element, it will gradually change back to its original style.

Change Several Property Values

The following example adds a transition effect for both the width and height property, with a duration of 2 seconds for the width and 4 seconds for the height:

Example

```
div {  
  -webkit-transition: width 2s, height 4s; /* Safari */  
  transition: width 2s, height 4s;  
}
```

Specify the Speed Curve of the Transition

The `transition-timing-function` property specifies the speed curve of the transition effect.

The `transition-timing-function` property 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

The following example shows the some of the different speed curves that can be used:

Example

```
#div1 {transition-timing-function: linear;}
#div2 {transition-timing-function: ease;}
#div3 {transition-timing-function: ease-in;}
#div4 {transition-timing-function: ease-out;}
#div5 {transition-timing-function: ease-in-out;}
```

Delay the Transition Effect

The `transition-delay` property specifies a delay (in seconds) for the transition effect.

The following example has a 1 second delay before starting:

Example

```
div {
  -webkit-transition-delay: 1s; /* Safari */
  transition-delay: 1s;
}
```

Transition + Transformation

The following example also adds a transformation to the transition effect:

Example

```
div {
  -webkit-transition: width 2s, height 2s, -webkit-transform 2s; /* Safari */
  transition: width 2s, height 2s, transform 2s;
}
```

More Transition Examples

The CSS3 transition properties can be specified one by one, like this:

Example

```
div {  
  transition-property: width;  
  transition-duration: 2s;  
  transition-timing-function: linear;  
  transition-delay: 1s;  
}
```

or by using the shorthand property `transition`:

Example

```
div {  
  transition: width 2s linear 1s;  
}
```

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":

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;
}

```

Note: If the `animation-duration` property is not specified, the animation will have no effect, because the default value is 0.

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

It is also possible to use percent. By using percent, you can add as many style changes as you like.

The following example will change the background-color of the `<div>` element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

Example

```

/* 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;
}

```

The following example will change both the background-color and the position of the `<div>` element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

Examples

```

/* The animation code */
@keyframes example {
  0% {background-color: red; left:0px; top:0px;}
}

```

```

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

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

```

Delay an Animation

The `animation-delay` property specifies a delay for the start of an animation.

The following example has a 2 seconds delay before starting the animation:

Example

```

div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
  animation-delay: 2s;
}

```

Set How Many Times an Animation Should Run

The `animation-iteration-count` property specifies the number of times an animation should run.

The following example will run the animation 3 times before it stops:

Example

```

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

```

```
    animation-duration: 4s;
    animation-iteration-count: 3;
}
```

The following example uses the value "infinite" to make the animation continue for ever:

Example

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

Run Animation in Reverse Direction or Alternate Cycles

The `animation-direction` property is used to let an animation run in reverse direction or alternate cycles.

The following example will run the animation in reverse direction:

Example

```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
    animation-iteration-count: 3;
    animation-direction: reverse;
}
```

The following example uses the value "alternate" to make the animation first run forward, then backward, then forward:

Example

```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
    animation-iteration-count: 3;
```



```
    animation-direction: alternate;
}
```

Specify the Speed Curve of the Animation

The `animation-timing-function` property specifies the speed curve of the animation.

The `animation-timing-function` property can have the following values:

- `ease` - specifies an animation with a slow start, then fast, then end slowly (this is default)
- `linear` - specifies an animation with the same speed from start to end
- `ease-in` - specifies an animation with a slow start
- `ease-out` - specifies an animation with a slow end
- `ease-in-out` - specifies an animation with a slow start and end
- `cubic-bezier(n,n,n,n)` - lets you define your own values in a cubic-bezier function

The following example shows the some of the different speed curves that can be used:

Example

```
#div1 {animation-timing-function: linear;}
#div2 {animation-timing-function: ease;}
#div3 {animation-timing-function: ease-in;}
#div4 {animation-timing-function: ease-out;}
#div5 {animation-timing-function: ease-in-out;}
```

Animation Shorthand Property

The example below uses six of the animation properties:

Example

```
div {
    animation-name: example;
    animation-duration: 5s;
    animation-timing-function: linear;
    animation-delay: 2s;
    animation-iteration-count: infinite;
    animation-direction: alternate;
}
```

The same animation effect as above can be achieved by using the shorthand `animation` property:

Example

```
div {
    animation: example 5s linear 2s infinite alternate;
}
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
div {
```

```
    width: 100px;
```

```
    height: 100px;
```

```
    background-color: red;
```

```
    position: relative;
```

```
    /* Safari 4.0 - 8.0 */
```

```
    -webkit-animation-name: example;
```

```
    -webkit-animation-duration: 5s;
```

```
    -webkit-animation-timing-function: linear;
```

```
    -webkit-animation-delay: 2s;
```

```
    -webkit-animation-iteration-count: infinite;
```

```
    -webkit-animation-direction: alternate;
```

```
    /* Standard syntax */
```

```
    animation-name: example;
```

```
    animation-duration: 5s;
```

```
    animation-timing-function: linear;
```

```
    animation-delay: 2s;
```

```
    animation-iteration-count: infinite;
```

```
    animation-direction: alternate;
```

```

}

/* Safari 4.0 - 8.0 */

@-webkit-keyframes example {

    0% {background-color:red; left:0px; top:0px;}

    25% {background-color:yellow; left:200px; top:0px;}

    50% {background-color:blue; left:200px; top:200px;}

    75% {background-color:green; left:0px; top:200px;}

    100% {background-color:red; left:0px; top:0px;}

}

/* Standard syntax */

@keyframes example {

    0% {background-color:red; left:0px; top:0px;}

    25% {background-color:yellow; left:200px; top:0px;}

    50% {background-color:blue; left:200px; top:200px;}

    75% {background-color:green; left:0px; top:200px;}

    100% {background-color:red; left:0px; top:0px;}

}

</style>

</head>

<body>

<p><b>Note:</b> This example does not work in Internet Explorer 9 and earlier versions.</p>

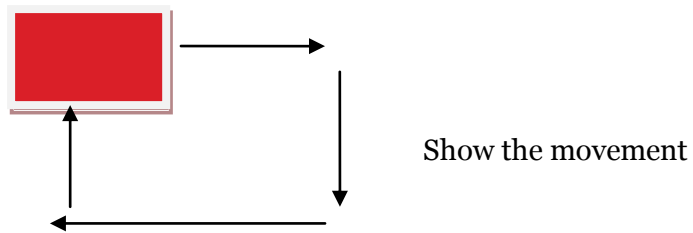
```

```
<div></div>
```

```
</body>
```

```
</html>
```

Demo



CSS Images

Learn how to style images using CSS.

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%;  
}
```

Thumbnail Images

Use the `border` property to create thumbnail images.

Thumbnail Image:



Example

```
img {  
  border: 1px solid #ddd;  
  border-radius: 4px;  
  padding: 5px;  
}
```

```

```

Thumbnail Image as Link:



Example

```
a {  
  display: inline-block;  
  border: 1px solid #ddd;  
  border-radius: 4px;  
  padding: 5px;  
  transition: 0.3s;  
}  
  
a:hover {  
  box-shadow: 0 0 2px 1px rgba  
    (0, 140, 186, 0.5);  
}  
  
<a href="paris.jpg">  
    
</a>
```

Responsive Images

Responsive images will automatically adjust to fit the size of the screen.

Resize the browser window to see the effect:



If you want an image to scale down if it has to, but never scale up to be larger than its original size, add the following:

Example

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

Center an Image

To center an image within the page, use `margin: auto;` and make it into a **block** element:



Example

```
img {  
  display: block;  
  margin: auto;  
  width: 50%;  
}
```

Image Text

How to position text in an image:

Example



Bottom Left

Top Left

Top Right

Bottom Right

Centered

CSS Buttons

Learn how to style buttons using CSS.

Basic Button Styling

Example

```
.button {  
  background-color: #4CAF50; /* Green */  
  border: none;  
  color: white;  
  padding: 15px 32px;  
  text-align: center;  
  text-decoration: none;  
  display: inline-block;  
  font-size: 16px;  
}
```

Button Colors

Use the `background-color` property to change the background color of a button:

Example


```
.button1 {background-color: #4CAF50;} /* Green */  
.button2 {background-color: #008CBA;} /* Blue */  
.button3 {background-color: #f44336;} /* Red */  
.button4 {background-color: #e7e7e7; color: black;} /* Gray */  
.button5 {background-color: #555555;} /* Black */
```

Button Sizes

Use the `font-size` property to change the font size of a button:

Example

```
.button1 {font-size: 10px;}  
.button2 {font-size: 12px;}  
.button3 {font-size: 16px;}  
.button4 {font-size: 20px;}  
.button5 {font-size: 24px;}
```

Use the `padding` property to change the padding of a button:

Example

```
.button1 {padding: 10px 24px;}  
.button2 {padding: 12px 28px;}  
.button3 {padding: 14px 40px;}  
.button4 {padding: 32px 16px;}  
.button5 {padding: 16px;}
```

Rounded Buttons

Use the `border-radius` property to add rounded corners to a button:

Example

```
.button1 {border-radius: 2px;}  
.button2 {border-radius: 4px;}  
.button3 {border-radius: 8px;}  
.button4 {border-radius: 12px;}  
.button5 {border-radius: 50%;}
```

Colored Button Borders

Use the `border` property to add a colored border to a button:

Example

```
.button1 {  
  background-color: white;  
  color: black;
```

```
border: 2px solid #4CAF50; /* Green */
}
```

...

Hoverable Buttons

Use the `:hover` selector to change the style of a button when you move the mouse over it.

Tip: Use the `transition-duration` property to determine the speed of the "hover" effect:

Example

```
.button {
  -webkit-transition-duration: 0.4s; /* Safari */
  transition-duration: 0.4s;
}

.button:hover {
  background-color: #4CAF50; /* Green */
  color: white;
}
```

...

Shadow Buttons

Use the `box-shadow` property to add shadows to a button:

Example

```
.button1 {
  box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2), 0 6px 20px 0 rgba(0,0,0,0.19);
}

.button2:hover {
  box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);
}
```

Disabled Buttons

Use the `opacity` property to add transparency to a button (creates a "disabled" look).

Tip: You can also add the `cursor` property with a value of "not-allowed", which will display a "no parking sign" when you mouse over the button:

Example

```
.disabled {
  opacity: 0.6;
  cursor: not-allowed;
}
```

Button Width

By default, the size of the button is determined by its text content (as wide as its content). Use the `width` property to change the width of a button:

Example

```
.button1 {width: 250px;}  
.button2 {width: 50%;}  
.button3 {width: 100%;}
```

Button Groups

Remove margins and add `float:left` to each button to create a button group:

Example

```
.button {  
  float: left;  
}
```

Bordered Button Groups

Use the `border` property to create a bordered button group:

Example

```
.button {  
  float: left;  
  border: 1px solid green  
}
```

CSS3 Multi-column Layout

In this chapter you will learn about the following multi-column properties:

- `column-count`
- `column-gap`
- `column-rule-style`
- `column-rule-width`
- `column-rule-color`
- `column-rule`
- `column-span`
- `column-width`

CSS3 Create Multiple Columns

The `column-count` property specifies the number of columns an element should be divided into.

The following example will divide the text in the `<div>` element into 3 columns:

Example

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

CSS3 Specify the Gap Between Columns

The `column-gap` property specifies the gap between the columns.

The following example specifies a 40 pixels gap between the columns:

Example

```
div {
  -webkit-column-gap: 40px; /* Chrome, Safari, Opera */
  -moz-column-gap: 40px; /* Firefox */
  column-gap: 40px;
}
```

CSS3 Column Rules

The `column-rule-style` property specifies the style of the rule between columns:

Example

```
div {
  -webkit-column-rule-style: solid; /* Chrome, Safari, Opera */
  -moz-column-rule-style: solid; /* Firefox */
  column-rule-style: solid;
}
```

The `column-rule-width` property specifies the width of the rule between columns:

Example

```
div {
  -webkit-column-rule-width: 1px; /* Chrome, Safari, Opera */
  -moz-column-rule-width: 1px; /* Firefox */
  column-rule-width: 1px;
}
```

The `column-rule-color` property specifies the color of the rule between columns:

Example

```
div {  
  -webkit-column-rule-color: lightblue; /* Chrome, Safari, Opera */  
  -moz-column-rule-color: lightblue; /* Firefox */  
  column-rule-color: lightblue;  
}
```

The `column-rule` property is a shorthand property for setting all the `column-rule-*` properties above.

The following example sets the width, style, and color of the rule between columns:

Example

```
div {  
  -webkit-column-rule: 1px solid lightblue; /* Chrome, Safari, Opera */  
  -moz-column-rule: 1px solid lightblue; /* Firefox */  
  column-rule: 1px solid lightblue;  
}
```

Specify How Many Columns an Element Should Span

The `column-span` property specifies how many columns an element should span across.

The following example specifies that the `<h2>` element should span across all columns:

Example

```
h2 {  
  -webkit-column-span: all; /* Chrome, Safari, Opera */  
  column-span: all;  
}
```

Specify The Column Width

The `column-width` property specifies a suggested, optimal width for the columns.

The following example specifies that the suggested, optimal width for the columns should be 100px:

Example

```
div {  
  -webkit-column-width: 100px; /* Chrome, Safari, Opera */  
  -moz-column-width: 100px; /* Firefox */  
  column-width: 100px;  
}
```

CSS3 User Interface

CSS3 has new user interface features such as resizing elements, outlines, and box sizing.

In this chapter you will learn about the following user interface properties:

- `resize`
- `outline-offset`

CSS3 Resizing

The `resize` property specifies whether or not an element should be resizable by the user.

This div element is resizable by the user (works in Chrome, Firefox, Safari and Opera).

The following example lets the user resize only the width of a `<div>` element:

Example

```
div {  
  resize: horizontal;  
  overflow: auto;  
}
```

The following example lets the user resize only the height of a `<div>` element:

Example

```
div {  
  resize: vertical;  
  overflow: auto;  
}
```

The following example lets the user resize both the height and the width of a `<div>` element:

Example

```
div {  
  resize: both;  
  overflow: auto;  
}
```

CSS3 Outline Offset

The `outline-offset` property adds space between an outline and the edge or border of an element.

Outlines differ from borders in three ways:

- An outline is a line drawn around elements, outside the border edge
- An outline does not take up space
- An outline may be non-rectangular

This div has an outline 15px outside the border edge.

The following example uses the outline-offset property to add a 15px space between the border and the outline:

Example

```
div {
  border: 1px solid black;
  outline: 1px solid red;
  outline-offset: 15px;
}
```

CSS3 User Interface Properties

The following table lists all the user interface properties:

Property	Description
box-sizing	Allows you to include the padding and border in an element's total width and height
nav-down	Specifies where to navigate when using the arrow-down navigation key
nav-index	Specifies the tabbing order for an element
nav-left	Specifies where to navigate when using the arrow-left navigation key
nav-right	Specifies where to navigate when using the arrow-right navigation key
nav-up	Specifies where to navigate when using the arrow-up navigation key
outline-offset	Adds space between an outline and the edge or border of an element
resize	Specifies whether or not an element is resizable by the user

```
<!DOCTYPE html>
```

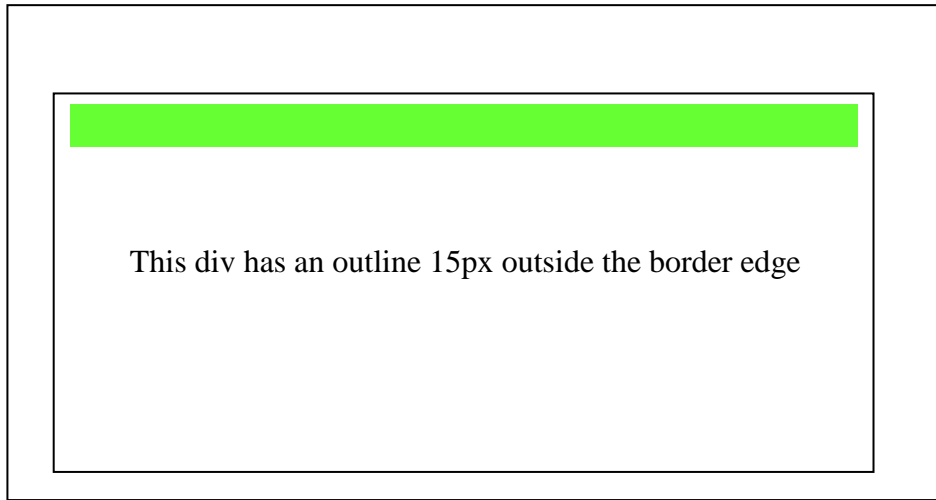
```
<html>
```

```
<head>
```

```
<style>
```

```
div {  
  
    margin: 20px;  
  
    padding: 10px;  
  
    width: 300px;  
  
    height: 100px;  
  
    border: 1px solid black;  
  
    outline: 1px solid red;  
  
    /* Move the outline 15px away from the border */  
  
    outline-offset: 15px;  
  
}  
  
</style>  
  
</head>  
  
<body>  
  
<p><b>Note:</b> Internet Explorer does not support the outline-offset property.</p>  
  
<div>This div has an outline 15px outside the border edge.</div>  
  
</body>  
  
</html>
```

Note: Internet Explorer does not support the outline-offset property.



CSS3 Box Sizing

The CSS3 `box-sizing` property allows us to include the padding and border in an element's total width and height.

Without the CSS3 box-sizing Property

By default, the width and height of an element is calculated like this:

width + padding + border = actual width of an element
height + padding + border = actual height of an element

This means: When you set the width/height of an element, the element often appear bigger than you have set (because the element's border and padding are added to the element's specified width/height).

The following illustration shows two `<div>` elements with the same specified width and height:

This div is smaller (width is 300px and height is 100px).

This div is bigger (width is also 300px and height is 100px).

The two `<div>` elements above end up with different sizes in the result (because div2 has a padding specified):

Example

```
.div1 {  
  width: 300px;  
  height: 100px;  
  border: 1px solid blue;  
}  
  
.div2 {  
  width: 300px;  
  height: 100px;  
  padding: 50px;  
  border: 1px solid red;  
}
```

So, for a long time web developers have specified a smaller width value than they wanted, because they had to subtract out the padding and borders.

With CSS3, the `box-sizing` property solves this problem.

With the CSS3 `box-sizing` Property

The CSS3 `box-sizing` property allows us to include the padding and border in an element's total width and height.

If you set `box-sizing: border-box;` on an element padding and border are included in the width and height:

Both divs are the same size now!

Hooray!

Here is the same example as above, with `box-sizing: border-box;` added to both `<div>` elements:

Example

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

.div2 {
  width: 300px;
  height: 100px;
  padding: 50px;
  border: 1px solid red;
  box-sizing: border-box;
}
```

Since the result of using the `box-sizing: border-box;` is so much better, many developers want all elements on their pages to work this way.

The code below ensures that all elements are sized in this more intuitive way. Many browsers already use `box-sizing: border-box;` for many form elements (but not all - which is why inputs and text areas look different at width: 100%;).

Applying this to all elements is safe and wise:

Example

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

CSS3 Flexbox

Flexible boxes, or flexbox, is a new layout mode in CSS3.

Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.

For many applications, the flexible box model provides an improvement over the block model in that it does not use floats, nor do the flex container's margins collapse with the margins of its contents.

CSS3 Flex box Concepts

Flexbox consists of flex containers and flex items.

A flex container is declared by setting the `display` property of an element to either `flex` (rendered as a block) or `inline-flex` (rendered as inline).

Inside a flex container there is one or more flex items.

Note: Everything outside a flex container and inside a flex item is rendered as usual. Flexbox defines how flex items are laid out inside a flex container.

Flex items are positioned inside a flex container along a flex line. By default there is only one flex line per flex container.

The following example shows three flex items. They are positioned by default: along the horizontal flex line, from left to right:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: -webkit-flex;
  display: flex;
  width: 400px;
  height: 250px;
  background-color: lightgrey;
}

.flex-item {
  background-color: cornflowerblue;
  width: 100px;
  height: 100px;
  margin: 10px;
}
</style>
```

```

</head>
<body>

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

</body>
</html>

```

It is also possible to change the direction of the flex line.

If we set the `direction` property to `rtl` (right-to-left), the text is drawn right to left, and also the flex line changes direction, which will change the page layout:

Example

```

body {
  direction: rtl;
}

.flex-container {
  display: -webkit-flex;
  display: flex;
  width: 400px;
  height: 250px;
  background-color: lightgrey;
}

.flex-item {
  background-color: cornflowerblue;
  width: 100px;
  height: 100px;
  margin: 10px;
}

```

Flex Direction

The `flex-direction` property specifies the direction of the flexible items inside the flex container. The default value of `flex-direction` is `row` (left-to-right, top-to-bottom).

The other values are as follows:

- `row-reverse` - If the writing-mode (direction) is left to right, the flex items will be laid out right to left
- `column` - If the writing system is horizontal, the flex items will be laid out vertically
- `column-reverse` - Same as `column`, but reversed

The following example shows the result of using the `row-reverse` value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-flex-direction: row-reverse;  
  flex-direction: row-reverse;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

The following example shows the result of using the `column` value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-flex-direction: column;  
  flex-direction: column;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

The following example shows the result of using the `column-reverse` value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-flex-direction: column-reverse;  
  flex-direction: column-reverse;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

The justify-content Property

The `justify-content` property horizontally aligns the flexible container's items when the items do not use all available space on the main-axis.

The possible values are as follows:

- `flex-start` - Default value. Items are positioned at the beginning of the container

- `flex-end` - Items are positioned at the end of the container
- `center` - Items are positioned at the center of the container
- `space-between` - Items are positioned with space between the lines
- `space-around` - Items are positioned with space before, between, and after the lines

The following example shows the result of using the `flex-end` value:

Example

```
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-justify-content: flex-end;
  justify-content: flex-end;
  width: 400px;
  height: 250px;
  background-color: lightgrey;
}
```

The following example shows the result of using the `center` value:

Example

```
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-justify-content: center;
  justify-content: center;
  width: 400px;
  height: 250px;
  background-color: lightgrey;
}
```

The following example shows the result of using the `space-between` value:

Example

```
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-justify-content: space-between;
  justify-content: space-between;
  width: 400px;
  height: 250px;
  background-color: lightgrey;
}
```

The following example shows the result of using the `space-around` value:

Example

```
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-justify-content: space-around;
  justify-content: space-around;
  width: 400px;
  height: 250px;
  background-color: lightgrey;
}
```

The align-items Property

The `align-items` property vertically aligns the flexible container's items when the items do not use all available space on the cross-axis.

The possible values are as follows:

- `stretch` - Default value. Items are stretched to fit the container
- `flex-start` - Items are positioned at the top of the container
- `flex-end` - Items are positioned at the bottom of the container
- `center` - Items are positioned at the center of the container (vertically)
- `baseline` - Items are positioned at the baseline of the container

The following example shows the result of using the `stretch` value (this is the default value):

Example

```
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-align-items: stretch;
  align-items: stretch;
  width: 400px;
  height: 250px;
  background-color: lightgrey;
}
```

The following example shows the result of using the `flex-start` value:

Example

```
.flex-container {
  display: -webkit-flex;
  display: flex;
  -webkit-align-items: flex-start;
  align-items: flex-start;
  width: 400px;
  height: 250px;
  background-color: lightgrey;
}
```

The following example shows the result of using the `flex-end` value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-align-items: flex-end;  
  align-items: flex-end;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

The following example shows the result of using the `center` value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-align-items: center;  
  align-items: center;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

The following example shows the result of using the `baseline` value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-align-items: baseline;  
  align-items: baseline;  
  width: 400px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

The flex-wrap Property

The `flex-wrap` property specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line.

The possible values are as follows:

- `nowrap` - Default value. The flexible items will not wrap

- `wrap` - The flexible items will wrap if necessary
- `wrap-reverse` - The flexible items will wrap, if necessary, in reverse order

The following example shows the result of using the `nowrap` value (this is the default value):

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-flex-wrap: nowrap;  
  flex-wrap: nowrap;  
  width: 300px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

The following example shows the result of using the `wrap` value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-flex-wrap: wrap;  
  flex-wrap: wrap;  
  width: 300px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

The following example shows the result of using the `wrap-reverse` value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-flex-wrap: wrap-reverse;  
  flex-wrap: wrap-reverse;  
  width: 300px;  
  height: 250px;  
  background-color: lightgrey;  
}
```

The align-content Property

The `align-content` property modifies the behavior of the `flex-wrap` property. It is similar to `align-items`, but instead of aligning flex items, it aligns flex lines.

The possible values are as follows:

- `stretch` - Default value. Lines stretch to take up the remaining space
- `flex-start` - Lines are packed toward the start of the flex container
- `flex-end` - Lines are packed toward the end of the flex container
- `center` - Lines are packed toward the center of the flex container
- `space-between` - Lines are evenly distributed in the flex container
- `space-around` - Lines are evenly distributed in the flex container, with half-size spaces on either end

The following example shows the result of using the `center` value:

Example

```
.flex-container {  
  display: -webkit-flex;  
  display: flex;  
  -webkit-flex-wrap: wrap;  
  flex-wrap: wrap;  
  -webkit-align-content: center;  
  align-content: center;  
  width: 300px;  
  height: 300px;  
  background-color: lightgrey;  
}
```

Flex Item Properties

Ordering

The `order` property specifies the order of a flexible item relative to the rest of the flexible items inside the same container:

Example

```
.flex-item {  
  background-color: cornflowerblue;  
  width: 100px;  
  height: 100px;  
  margin: 10px;  
}  
  
.first {  
  -webkit-order: -1;  
  order: -1;  
}
```

Margin

Setting `margin: auto;` will absorb extra space. It can be used to push flex items into different positions.

In the following example we set `margin-right: auto;` on the first flex item. This will cause all the extra space to be absorbed to the right of that element:

Example

```
.flex-item {
  background-color: cornflowerblue;
  width: 75px;
  height: 75px;
  margin: 10px;
}

.flex-item:first-child {
  margin-right: auto;
}
```

Perfect Centering

In the following example we will solve an almost daily problem: perfect centering.

It is very easy with flexbox. Setting `margin: auto;` will make the item perfectly centered in both axis:

Example

```
.flex-item {
  background-color: cornflowerblue;
  width: 75px;
  height: 75px;
  margin: auto;
}
```

align-self

The `align-self` property of flex items overrides the flex container's `align-items` property for that item. It has the same possible values as the `align-items` property.

The following example sets different `align-self` values to each flex item:

Example

```
.flex-item { background-color: cornflowerblue;
  width: 60px;
  min-height: 100px;
  margin: 10px;
}

.item1 {
  -webkit-align-self: flex-start;
  align-self: flex-start;
}
```

```
.item2 {  
  -webkit-align-self: flex-end;  
  align-self: flex-end;  
}
```

```
.item3 {  
  -webkit-align-self: center;  
  align-self: center;  
}
```

```
.item4 {  
  -webkit-align-self: baseline;  
  align-self: baseline;  
}
```

```
.item5 {  
  -webkit-align-self: stretch;  
  align-self: stretch;  
}
```

flex

The `flex` property specifies the length of the flex item, relative to the rest of the flex items inside the same container.

In the following example, the first flex item will consume 2/4 of the free space, and the other two flex items will consume 1/4 of the free space each:

Example

```
.flex-item {  
  background-color: cornflowerblue;  
  margin: 10px;  
}
```

```
.item1 {  
  -webkit-flex: 2;  
  flex: 2;  
}
```

```
.item2 {  
  -webkit-flex: 1;  
  flex: 1;  
}
```

```
.item3 {  
  -webkit-flex: 1;  
  flex: 1;  
}
```

Edit

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
.flex-container {
```

```
    display: -webkit-flex;
```

```
    display: flex;
```

```
    width: 400px;
```

```
    height: 250px;
```

```
    background-color: lightgrey;
```

```
}
```

```
.flex-item {
```

```
    background-color: cornflowerblue;
```

```
    margin: 10px;
```

```
}
```

```
.item1 {
```

```
    -webkit-flex: 2;
```

```
    flex: 2;
```

```
}
```

```
.item2 {
```

```
    -webkit-flex: 1;
```

```
    flex: 1;

}

.item3 {

    -webkit-flex: 1;

    flex: 1;

}

</style>

</head>

<body>

<div class="flex-container">

    <div class="flex-item item1">flex item 1</div>

    <div class="flex-item item2">flex item 2</div>

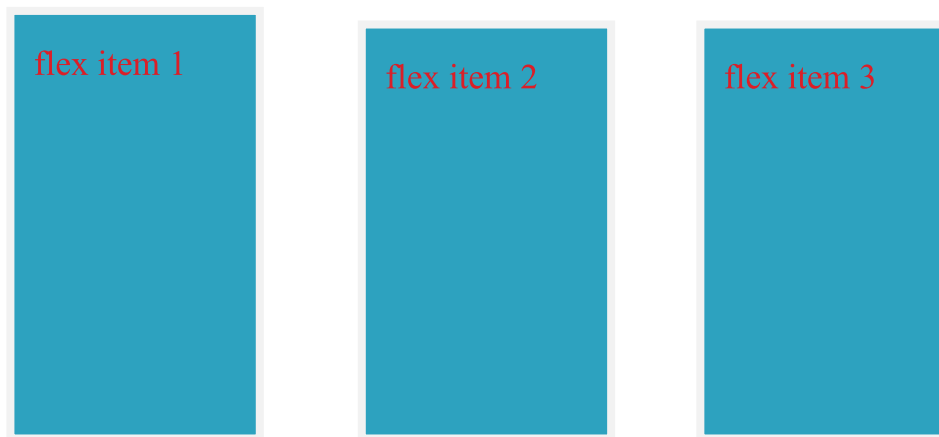
    <div class="flex-item item3">flex item 3</div>

</div>

</body>

</html>
```

Demo



CSS3 Media Queries

CSS2 Introduced Media Types

The `@media` rule, introduced in CSS2, made it possible to define different style rules for different media types.

Examples: You could have one set of style rules for computer screens, one for printers, one for handheld devices, one for television-type devices, and so on.

Unfortunately these media types never got a lot of support by devices, other than the print media type.

CSS3 Introduces Media Queries

Media queries in CSS3 extend 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 tablets, iPhone, and Androids.

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.

You can also have different stylesheets for different media:

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

CSS3 Media Types

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

Media Queries Simple Examples

One way to use media queries is to have an alternate CSS section right inside your style sheet.

The following example changes the background-color to lightgreen if the viewport is 480 pixels wide or wider (if the viewport is less than 480 pixels, the background-color will be pink):

Example

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

The following example shows a menu that will float to the left of the page if the viewport is 480 pixels wide or wider (if the viewport is less than 480 pixels, the menu will be on top of the content):

Example

```
@media screen and (min-width: 480px) {  
    #leftsidebar {width: 200px; float: left;}  
}
```



```
#main {margin-left:216px;}  
}
```

Edit

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

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

```
<style>
```

```
.wrapper {overflow: auto;}
```

```
#main {margin-left: 4px;}
```

```
#leftsidebar {
```

```
    float: none;
```

```
    width: auto;
```

```
}
```

```
#menulist {
```

```
    margin: 0;
```

```
    padding: 0;
```

```
}
```

```
.menuitem {
```

```
    background: #CDF0F6;
```

```
    border: 1px solid #d4d4d4;
```

```
    border-radius: 4px;
```

```
    list-style-type: none;
```

```
    margin: 4px;
```

```
    padding: 2px;
```

```

}

@media screen and (min-width: 480px) {

    #leftsidebar {width: 200px; float: left;}

    #main {margin-left: 216px;}

}

</style>

</head>

<body>

<div class="wrapper">

    <div id="leftsidebar">

        <ul id="menulist">

            <li class="menuitem">Menu-item 1</li>

            <li class="menuitem">Menu-item 2</li>

            <li class="menuitem">Menu-item 3</li>

            <li class="menuitem">Menu-item 4</li>

            <li class="menuitem">Menu-item 5</li>

        </ul>

    </div>

    <div id="main">

        <h1>Resize the browser window to see the effect!</h1>

        <p>This example shows a menu that will float to the left of the page if the viewport is 480 pixels wide or wider. If the viewport is less than 480 pixels, the menu
will be on top of the content.</p>

    </div>

</div>

```

</body>

</html>

CSS3 Media Queries - Examples

Let us look at some more examples of using media queries.

We will start with a list of names which function as email links. The HTML is:

Example 1

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  list-style-type: none;
}

ul li a {
  color: green;
  text-decoration: none;
  padding: 3px;
  display: block;
}
</style>
</head>
<body>

<ul>
  <li><a data-email="johndoe@example.com" href="mailto:johndoe@example.com">John Doe</a></li>
  <li><a data-email="marymoe@example.com" href="mailto:marymoe@example.com">Mary
Moe</a></li>
  <li><a data-email="amandapanda@example.com" href="mailto:amandapanda@example.com">Amanda
Panda</a></li>
</ul>

</body>
</html>
```

Notice the `data-email` attribute. In HTML5, we can use attributes prefixed with `data-` to store information. We will use the `data-` attribute later.

Width from 520 to 699px - Apply an email icon to each link

When the browser's width is between 520 and 699px, we will apply an email icon to each email link:

Example 2

```
@media screen and (max-width: 699px) and (min-width: 520px) {  
  ul li a {  
    padding-left: 30px;  
    background: url(email-icon.png) left center no-repeat;  
  }  
}
```

Width from 700 to 1000px - Preface the links with a text

When the browser's width is between from 700 to 1000px, we will preface each email link with the text "Email: ":

Example 3

```
@media screen and (max-width: 1000px) and (min-width: 700px) {  
  ul li a:before {  
    content: "Email: ";  
    font-style: italic;  
    color: #666666;  
  }  
}
```

Width above 1001px - Apply email address to links

When the browser's width is above 1001px, we will append the email address to the links.

We will use the value of the `data-` attribute to add the email address after the person's name:

Example 4

```
@media screen and (min-width: 1001px) {  
  ul li a:after {  
    content: " (" attr(data-email) )";  
    font-size: 12px;  
    font-style: italic;  
    color: #666666;  
  }  
}
```

Width above 1151px - Add icon as we used before

For browser widths above 1151px, we will again add the icon as we used before.

Here, we do not have to write an additional media query block, we can just append an additional media query to our already existing one using a comma (this will behave like an OR operator):

Example 5

```
@media screen and (max-width: 699px) and (min-width: 520px), (min-width: 1151px) {  
  ul li a {  
    padding-left: 30px;  
    background: url(email-icon.png) left center no-repeat;  
  }  
}
```

Chapter 3

Responsive Web Design - Introduction

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:

Desktop



Tablet



Phone



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.

Responsive Web Design - 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">
```

A `<meta>` viewport element 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:

Tip: If you are browsing this page with a phone or a tablet, you can click on the two links to see the difference.



Without the viewport meta tag



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

What is a Grid-View?

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

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:

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

Read more about the `box-sizing` property in our [CSS3 Box Sizing](#) chapter.

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

Example

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

The example above is fine if the web page only contains two columns.

However, we want to use a responsive grid-view with 12 columns, to have more control over the web page.

First we must calculate the percentage for one column: $100\% / 12 \text{ columns} = 8.33\%$.

Then we make one class for each of the 12 columns, `class="col-"` and a number defining how many columns the section should span:

CSS:

```
.col-1 {width: 8.33%;}  
.col-2 {width: 16.66%;}  
.col-3 {width: 25%;}  
.col-4 {width: 33.33%;}  
.col-5 {width: 41.66%;}  
.col-6 {width: 50%;}  
.col-7 {width: 58.33%;}  
.col-8 {width: 66.66%;}  
.col-9 {width: 75%;}  
.col-10 {width: 83.33%;}  
.col-11 {width: 91.66%;}  
.col-12 {width: 100%;}
```

All these columns should be floating to the left, and have a padding of 15px:

CSS:

```
[class*="col-"] {  
  float: left;  
  padding: 15px;  
  border: 1px solid red;  
}
```

Each row should be wrapped in a `<div>`. The number of columns inside a row should always add up to 12:

HTML:

```
<div class="row">
  <div class="col-3">...</div>
  <div class="col-9">...</div>
</div>
```

The columns inside a row are all floating to the left, and are therefore taken out of the flow of the page, and other elements will be placed as if the columns does not exist. To prevent this, we will add a style that clears the flow:

CSS:

```
.row::after {
  content: "";
  clear: both;
  display: block;
}
```

We also want to add some styles and colors to make it look better:

Example

```
html {
  font-family: "Lucida Sans", sans-serif;
}
.header {
  background-color: #9933cc;
  color: #ffffff;
  padding: 15px;
}
.menu ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
.menu li {
  padding: 8px;
  margin-bottom: 7px;
  background-color: #33b5e5;
  color: #ffffff;
  box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}
.menu li:hover {
  background-color: #0099cc;
}
```

Responsive Web Design - Media Queries

Media query is a CSS technique introduced in CSS3.

It uses the `@media` rule to include a block of CSS properties only if a certain condition is true.

Example

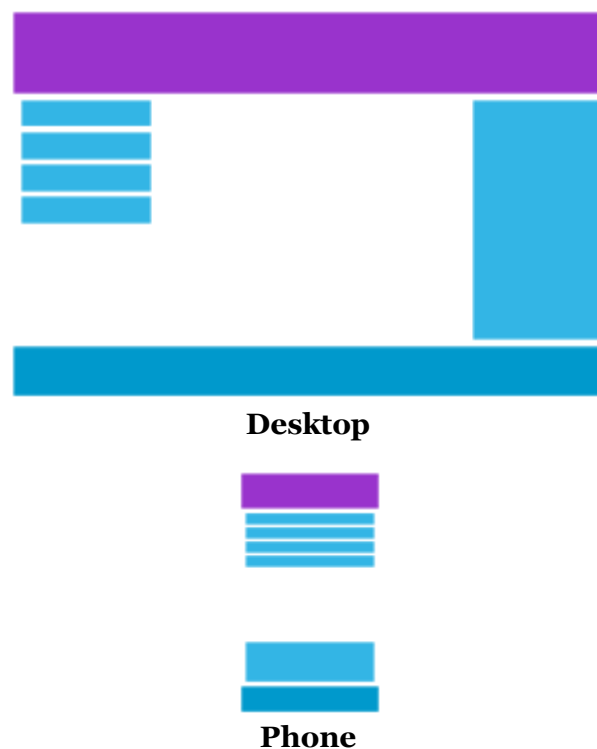
If the browser window is smaller than 500px, the background color will change to lightblue:

```
@media only screen and (max-width: 500px) {  
  body {  
    background-color: lightblue;  
  }  
}
```

Add a Breakpoint

Earlier in this tutorial we made a web page with rows and columns, and it was responsive, but it did not look good on a small screen.

Media queries can help with that. We can add a breakpoint where certain parts of the design will behave differently on each side of the breakpoint.



Use a media query to add a breakpoint at 768px:

Example

When the screen (browser window) gets smaller than 768px, each column should have a width of 100%:

```
/* For desktop: */
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}

@media only screen and (max-width: 768px) {
  /* For mobile phones: */
  [class*="col-"] {
    width: 100%;
  }
}
```

Always Design for Mobile First

Mobile First means designing for mobile before designing for desktop or any other device (This will make the page display faster on smaller devices).

This means that we must make some changes in our CSS.

Instead of changing styles when the width gets *smaller* than 768px, we should change the design when the width gets *larger* than 768px. This will make our design Mobile First:

Example

```
/* For mobile phones: */
[class*="col-"] {
  width: 100%;
}

@media only screen and (min-width: 768px) {
  /* For desktop: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
```

```

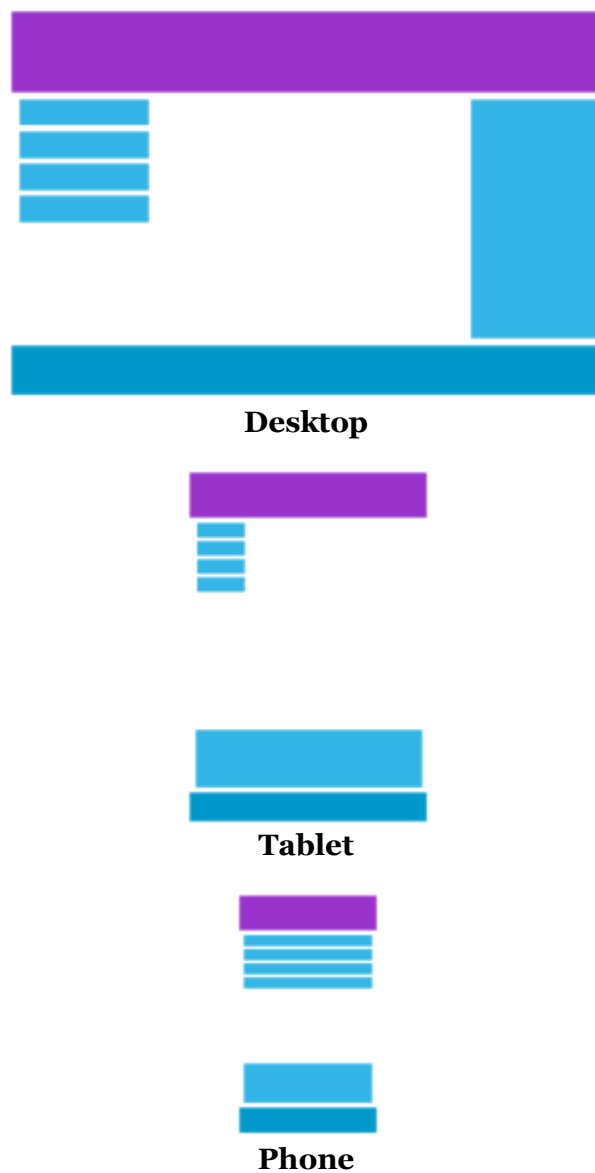
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
}

```

Another Breakpoint

You can add as many breakpoints as you like.

We will also insert a breakpoint between tablets and mobile phones.



We do this by adding one more media query (at 600px), and a set of new classes for devices larger than 600px (but smaller than 768px):

Example

Note that the two sets of classes are almost identical, the only difference is the name (col- and col-m-):

```
/* For mobile phones: */
[class*="col-"] {
  width: 100%;
}
@media only screen and (min-width: 600px) {
  /* For tablets: */
  .col-m-1 {width: 8.33%;}
  .col-m-2 {width: 16.66%;}
  .col-m-3 {width: 25%;}
  .col-m-4 {width: 33.33%;}
  .col-m-5 {width: 41.66%;}
  .col-m-6 {width: 50%;}
  .col-m-7 {width: 58.33%;}
  .col-m-8 {width: 66.66%;}
  .col-m-9 {width: 75%;}
  .col-m-10 {width: 83.33%;}
  .col-m-11 {width: 91.66%;}
  .col-m-12 {width: 100%;}
}
@media only screen and (min-width: 768px) {
  /* For desktop: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
}

<!DOCTYPE html>

<html>

<head>

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

<style>

* {

  box-sizing: border-box;
```

```

}

.row::after {
    content: "";
    clear: both;
    display: block;
}

[class*="col-"] {
    float: left;
    padding: 15px;
}

html {
    font-family: "Lucida Sans", sans-serif;
}

.header {
    background-color: #9933cc;
    color: #ffffff;
    padding: 15px;
}

.menu ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

.menu li {
    padding: 8px;
    margin-bottom: 7px;
    background-color: #33b5e5;
    color: #ffffff;
    box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.menu li:hover {
    background-color: #0099cc;

```



```

}

.aside {
    background-color: #33b5e5;
    padding: 15px;
    color: #ffffff;
    text-align: center;
    font-size: 14px;
    box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.footer {
    background-color: #0099cc;
    color: #ffffff;
    text-align: center;
    font-size: 12px;
    padding: 15px;
}

/* For mobile phones: */
[class*="col-"] {
    width: 100%;
}

@media only screen and (min-width: 600px) {
    /* For tablets: */

    .col-m-1 {width: 8.33%;}
    .col-m-2 {width: 16.66%;}
    .col-m-3 {width: 25%;}
    .col-m-4 {width: 33.33%;}
    .col-m-5 {width: 41.66%;}
    .col-m-6 {width: 50%;}
    .col-m-7 {width: 58.33%;}
    .col-m-8 {width: 66.66%;}
    .col-m-9 {width: 75%;}
    .col-m-10 {width: 83.33%;}
}

```

```

.col-m-11 {width: 91.66%;}
.col-m-12 {width: 100%;}
}

@media only screen and (min-width: 768px) {
  /* For desktop: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
}

</style>
</head>
<body>
<div class="header">
<h1>Chania</h1>
</div>
<div class="row">
<div class="col-3 col-m-3 menu">
<ul>
<li>The Flight</li>
<li>The City</li>
<li>The Island</li>
<li>The Food</li>
</ul>

```

</div>

<div class="col-6 col-m-9">

<h1>The City</h1>

<p>Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old town and the modern city.</p>

</div>

<div class="col-3 col-m-12">

<div class="aside">

<h2>What?</h2>

<p>Chania is a city on the island of Crete.</p>

<h2>Where?</h2>

<p>Crete is a Greek island in the Mediterranean Sea.</p>

<h2>How?</h2>

<p>You can reach Chania airport from all over Europe.</p>

</div>

</div>

</div>

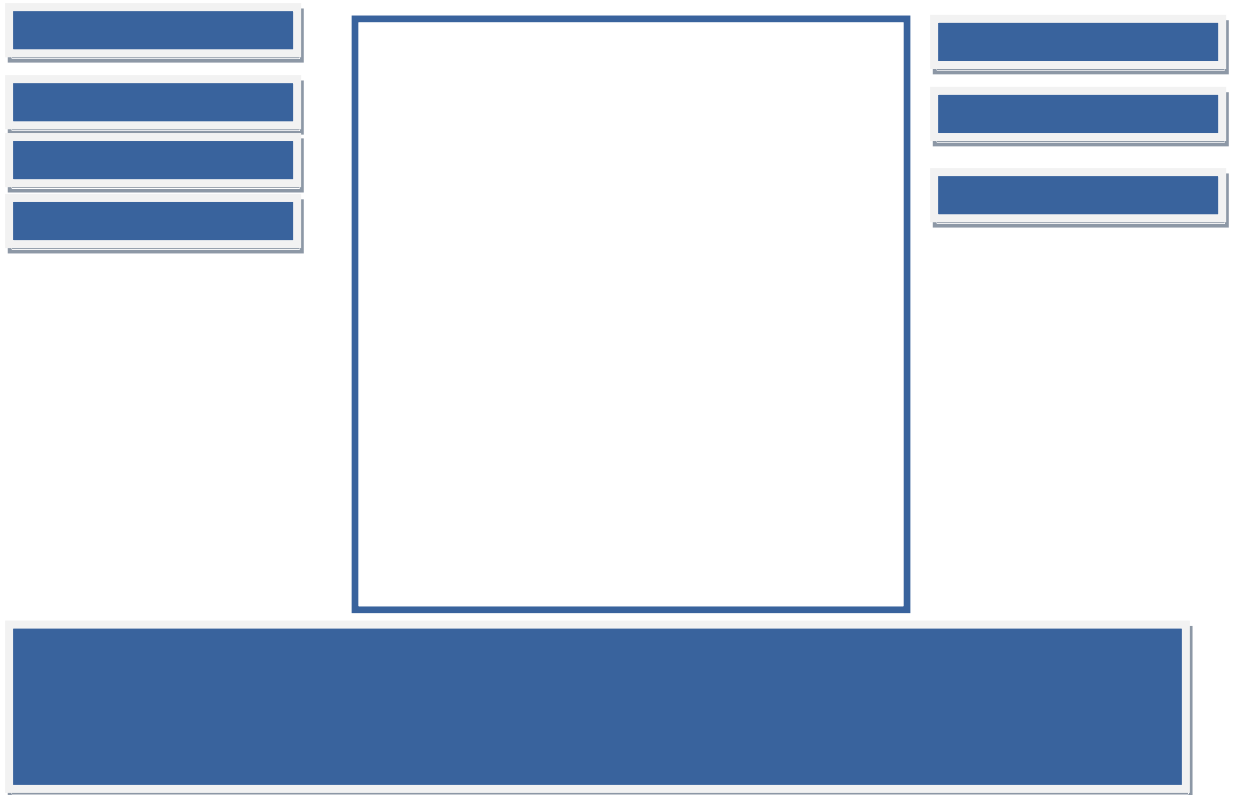
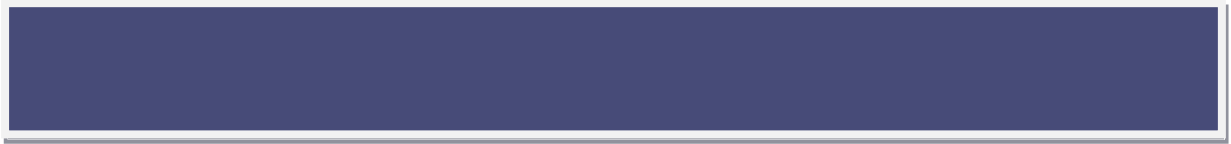
<div class="footer">

<p>Resize the browser window to see how the content respond to the resizing.</p>

</div>

</body>

</html>



Responsive Web Design - Images

Using The width Property

If the `width` property is set to 100%, the image will be responsive and scale up and down:

Example

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

Notice that in the example above, the image can be scaled up to be larger than its original size. A better solution, in many cases, will be to use the `max-width` property instead.

Using The max-width Property

If the `max-width` property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:

Example

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

Add an Image to The Example Web Page

Example

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

Background Images

Background images can also respond to resizing and scaling.

Here we will show three different methods:

1. If the `background-size` property is set to "contain", the background image will scale, and try to fit the content area. However, the image will keep its aspect ratio (the proportional relationship between the image's width and height):

Here is the CSS code:

Example

```
div {  
  width: 100%;  
  height: 400px;  
  background-image: url('img_flowers.jpg');  
  background-repeat: no-repeat;  
  background-size: contain;  
  border: 1px solid red;  
}
```

2. If the `background-size` property is set to "100% 100%", the background image will stretch to cover the entire content area:

Here is the CSS code:

Example

```
div {  
  width: 100%;  
  height: 400px;  
  background-image: url('img_flowers.jpg');  
  background-size: 100% 100%;  
  border: 1px solid red;  
}
```

3. If the `background-size` property is set to "cover", the background image will scale to cover the entire content area. Notice that the "cover" value keeps the aspect ratio, and some part of the background image may be clipped:

Here is the CSS code:

Example

```
div {  
  width: 100%;  
  height: 400px;  
  background-image: url('img_flowers.jpg');  
  background-size: cover;  
  border: 1px solid red;  
}
```

Different Images for Different Devices

A large image can be perfect on a big computer screen, but useless on a small device. Why load a large image when you have to scale it down anyway? To reduce the load, or for any other reasons, you can use media queries to display different images on different devices.

Here is one large image and one smaller image that will be displayed on different devices:



Example

```
/* For width smaller than 400px: */  
body {  
  background-image: url('img_smallflower.jpg');  
}  
  
/* For width 400px and larger: */  
@media only screen and (min-width: 400px) {  
  body {  
    background-image: url('img_flowers.jpg');  
  }  
}
```

You can use the media query `min-device-width`, instead of `min-width`, which checks the device width, instead of the browser width. Then the image will not change when you resize the browser window:

Example

```

/* For devices smaller than 400px: */
body {
    background-image: url('img_smallflower.jpg');
}

/* For devices 400px and larger: */
@media only screen and (min-device-width: 400px) {
    body {
        background-image: url('img_flowers.jpg');
    }
}

```

HTML5 <picture> Element

HTML5 introduced the <picture> element, which lets you define more than one image.

The <picture> element works similar to the <video> and <audio> elements. You set up different sources, and the first source that fits the preferences is the one being used:

Example

```

<picture>
  <source srcset="img_smallflower.jpg" media="(max-width: 400px)">
  <source srcset="img_flowers.jpg">
  
</picture>

```

The `srcset` attribute is required, and defines the source of the image.

The `media` attribute is optional, and accepts the media queries you find in [CSS @media rule](#).

You should also define an element for browsers that do not support the <picture> element.

Responsive Web Design - Videos

If the `width` property is set to 100%, the video player will be responsive and scale up and down:

Example

```

video {
    width: 100%;
    height: auto;
}

```

Notice that in the example above, the video player can be scaled up to be larger than its original size. A better solution, in many cases, will be to use the `max-width` property instead.

Using The max-width Property

If the `max-width` property is set to 100%, the video player will scale down if it has to, but never scale up to be larger than its original size:

Example

```
video {  
  max-width: 100%;  
  height: auto;  
}
```

Add a Video to the Example Web Page

We want to add a video in our example web page. The video will be resized to always take up all the available space:

Example

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

Edit

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

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

```
<style>
```

```
* {
```

```
  box-sizing: border-box;
```

```
}
```

```
video {
```

```
  width: 100%;
```

```
  height: auto;
```

```
}
```

```
.row:after {
```

```

    content: "";
    clear: both;
    display: block;
}

[class*="col-"] {
    float: left;
    padding: 15px;
    width: 100%;
}

@media only screen and (min-width: 600px) {
    .col-s-1 {width: 8.33%;}
    .col-s-2 {width: 16.66%;}
    .col-s-3 {width: 25%;}
    .col-s-4 {width: 33.33%;}
    .col-s-5 {width: 41.66%;}
    .col-s-6 {width: 50%;}
    .col-s-7 {width: 58.33%;}
    .col-s-8 {width: 66.66%;}
    .col-s-9 {width: 75%;}
    .col-s-10 {width: 83.33%;}
    .col-s-11 {width: 91.66%;}
    .col-s-12 {width: 100%;}
}

@media only screen and (min-width: 768px) {
    .col-1 {width: 8.33%;}
    .col-2 {width: 16.66%;}
    .col-3 {width: 25%;}

```

```
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
}

html {
    font-family: "Lucida Sans", sans-serif;
}

.header {
    background-color: #9933cc;
    color: #ffffff;
    padding: 15px;
}

.menu ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}

.menu li {
    padding: 8px;
    margin-bottom: 7px;
    background-color :#33b5e5;
```

```
    color: #ffffff;

    box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.menu li:hover {
    background-color: #0099cc;
}

.aside {
    background-color: #33b5e5;
    padding: 15px;
    color: #ffffff;
    text-align: center;
    font-size: 14px;
    box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.footer {
    background-color: #0099cc;
    color: #ffffff;
    text-align: center;
    font-size: 12px;
    padding: 15px;
}

</style>
</head>
<body>

<div class="header">
<h1>Chania</h1>
```

```

</div>

<div class="row">
  <div class="col-3 col-s-3 menu">
    <ul>
      <li>The Flight</li>
      <li>The City</li>
      <li>The Island</li>
      <li>The Food</li>
    </ul>
  </div>

  <div class="col-6 col-s-9">
    <h1>The City</h1>

    <p>Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old town and the modern city.</p>

    <video width="400" controls>
      <source src="mov_bbb.mp4" type="video/mp4">
      <source src="mov_bbb.ogv" type="video/ogg">
      Your browser does not support HTML5 video.
    </video>
  </div>

  <div class="col-3 col-s-12">
    <div class="aside">
      <h2>What?</h2>

      <p>Chania is a city on the island of Crete.</p>

      <h2>Where?</h2>

      <p>Crete is a Greek island in the Mediterranean Sea.</p>

      <h2>How?</h2>

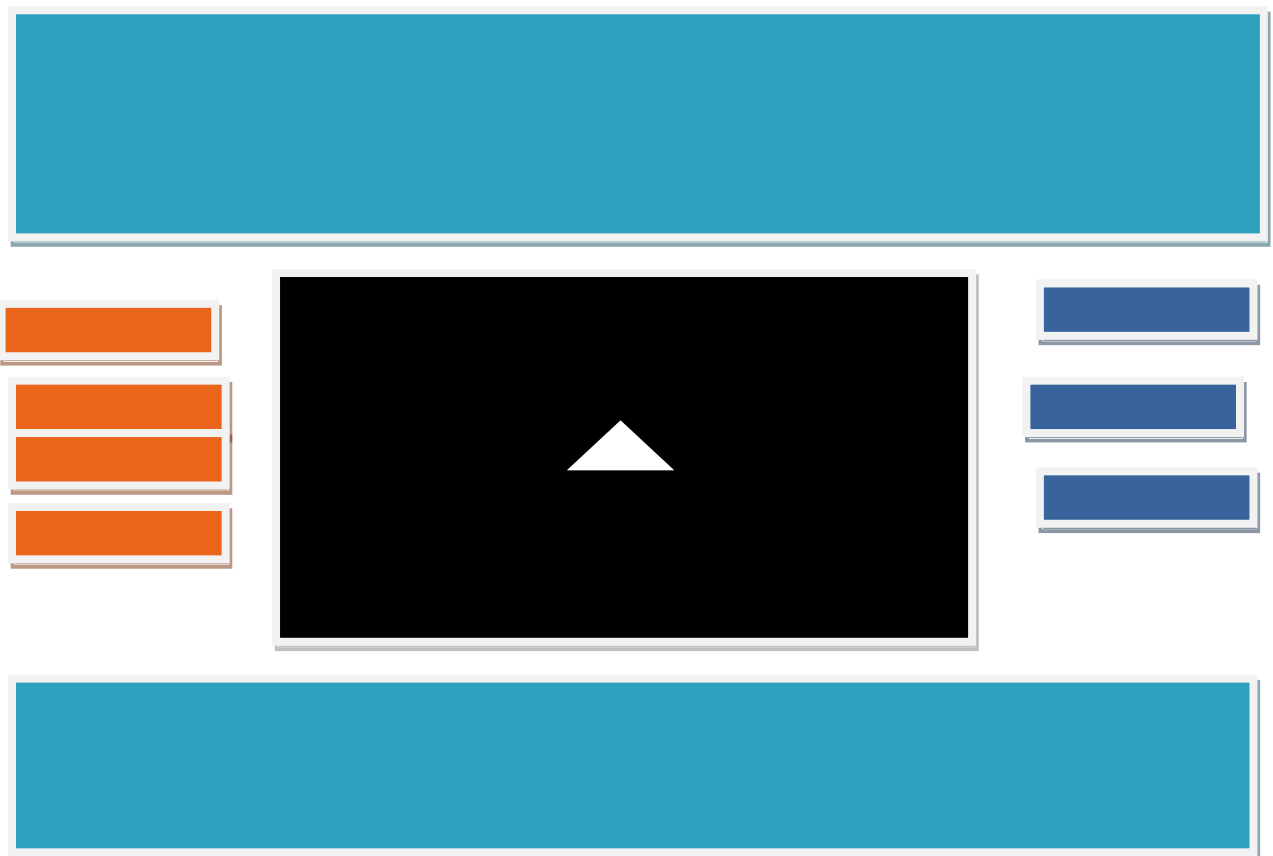
      <p>You can reach Chania airport from all over Europe.</p>

```

```

</div>
</div>
</div>
<div class="footer">
<p>Resize the browser window to see how the content respond to the resizing.</p>
</div>
</body>
</html>

```



There are many existing CSS Frameworks that offer Responsive Design.

They are free, and easy to use.

Using W3.CSS

A great way to create a responsive design, is to use a responsive style sheet, like [W3.CSS](https://www.w3schools.com/w3css/)

W3.CSS makes it easy to develop sites that look nice at any size; desktop, laptop, tablet, or phone:

Resize the page to see the responsiveness!

Resize the page to see the responsiveness!

London	Paris	Tokyo
London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.	Paris is the capital of France. The Paris area is one of the largest population centers in Europe, with more than 12 million inhabitants	Tokyo is the capital of Japan. It is the center of the Greater Tokyo Area, and the most populous metropolitan area in the world.

Example

```
<!DOCTYPE html>
<html>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/lib/w3.css">
<body>

<div class="w3-container w3-green">
  <h1>W3Schools Demo</h1>
  <p>Resize this responsive page!</p>
</div>

<div class="w3-row-padding">
  <div class="w3-third">
    <h2>London</h2>
    <p>London is the capital city of England.</p>
    <p>It is the most populous city in the United Kingdom,
    with a metropolitan area of over 13 million inhabitants.</p>
  </div>

  <div class="w3-third">
    <h2>Paris</h2>
    <p>Paris is the capital of France.</p>
    <p>The Paris area is one of the largest population centers in Europe,
    with more than 12 million inhabitants.</p>
  </div>
```

```

<div class="w3-third">
  <h2>Tokyo</h2>
  <p>Tokyo is the capital of Japan.</p>
  <p>It is the center of the Greater Tokyo Area,
    and the most populous metropolitan area in the world.</p>
</div>
</div>

</body>
</html>

```

Bootstrap

Another popular framework is Bootstrap, it uses HTML, CSS and jQuery to make responsive web pages.

Example

```

<!DOCTYPE html>
<html lang="en">
<head>
<title>Bootstrap Example</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
  <div class="jumbotron">
    <h1>My First Bootstrap Page</h1>
  </div>
  <div class="row">
    <div class="col-sm-4">
      ...
    </div>
    <div class="col-sm-4">
      ...
    </div>
    <div class="col-sm-4">
      ...
    </div>
  </div>
</div>

</body>
</html>

```


CHAPTER 4

CSS References

1. HTML and CSS: Design and Build Websites, by Jon Duckett 2000
2. JavaScript and JQuery: Interactive Front-End Web Development, by Jon Duckett
3. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics, by Jennifer Niederst Robbins
4. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, by Steve Krug 2000
5. Communicating Design: Developing Web Site Documentation for Design and Planning, by Dan M. Brown
6. Universal Principles of Design, Revised and Updated, by William Lidwell, Kritina Holden & Jill Butler< ISBN-13: 978-1592535873
7. Design for Hackers: Reverse Engineering Beauty, by David Kadavy
8. Design for Hackers: Reverse Engineering Beauty 1st Edition by David Kadavy
9. Neuro Web Design: What Makes Them Click? 1st Edition by Susan Weinschenk 2016
10. Mobile Design Book Kindle Edition by Paula Borowska **Publication Date:** August 12, 2014
11. Designing Web Interfaces: Principles and Patterns for Rich Interactions 1st Edition by Bill Scott
12. Designing Web Navigation: Optimizing the User Experience 1st Edition by James Kalbach
13. Mapping Experiences: A Complete Guide to Creating Value through Journeys, Blueprints, and Diagrams 1st Edition by James Kalbach
14. This is Service Design Thinking: Basics, Tools, Cases Paperback – January 11, 2012 by MarcStickdorn , Jakob Schneider2012

About the author



James kyaligonza was born in 1990 by the two peasant farmers kyaligonza Joseph and kyaligonza Edinus in 1994 joined Kibanjwa nursery school, in 1995 joined Bukerenge primary school where he finished his primary level 2004, in 2006 he joined secondary school in central school Hoima where he finished his O. level 2009, in 2010 he joined premier s.s.s Hoima where he earned his certificate of advanced level in 2012 and in 2013 he joined Makerere university Kampala(Mak) where he pursued his degree in library and information science in IT in 2015, currently he is working with express logistics group limited as the records manager and same time self employed practicing book writing, web site design