

Understanding CSS Essentials: Layouts, Managing Text Flow, Managing the Graphical Interface

Vyacheslav Koldovskyy
Last update: 01/07/2015

Agenda

- UI design
- Traditional CSS Box model
- Block-level and inline element
- Parent/child relationships
- Vendor prefixes
- CSS Flexbox Box model
- CSS Grid Layout model

Vendor Prefixes

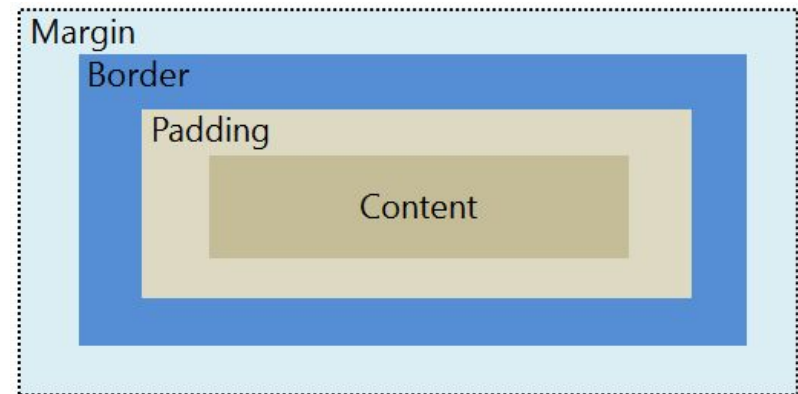
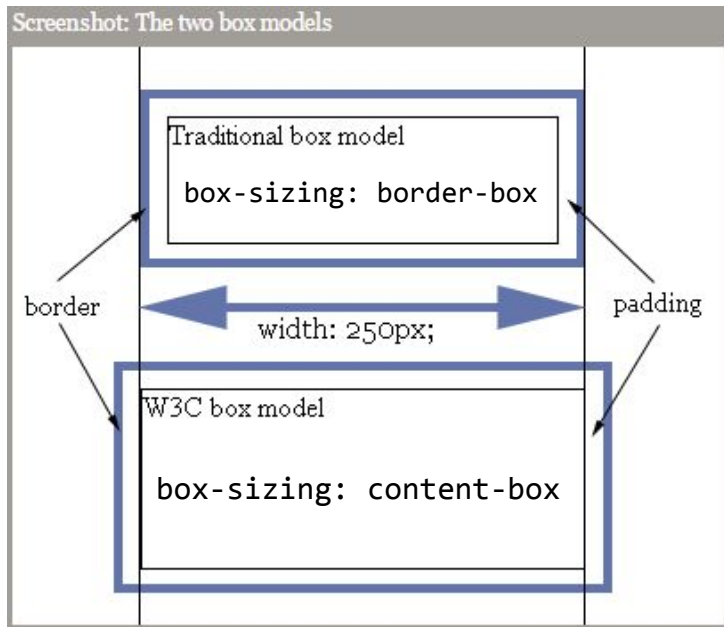
- CSS3 specification is still in draft format and undergoing modifications
- Need to use vendor prefixes with several CSS3 constructs
 - Internet Explorer uses the `-ms-` prefix.
 - Firefox supports the `-moz-` prefix.
 - Chrome and Safari support the `-webkit-` prefix.
 - ~~Opera supports the `-o-` prefix.~~

Two CSS box models

Box models

1. In the W3C box model, the width of an element gives the width of the *content* of the box, excluding padding and border.
2. In the traditional box model, the width of an element gives the width between the *borders* of the box, including padding and border.

By default, all browsers use the W3C box model, with the exception of IE in "Quirks Mode" (IE5.5 Mode), which uses the traditional one.

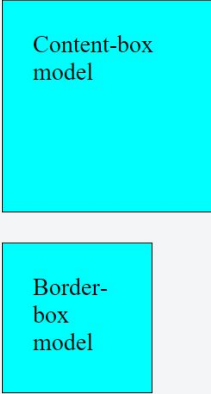


Box model sample

```
1 <div class="use-content-box"> Content-box model </div> HTML ⚙️
2 <div class="use-border-box"> Border-box model </div>

1
JAVASCRIPT ⚙️
```

```
1 div {
2   background: aqua;
3   width: 100px;
4   height: 100px;
5   border: 1px solid black;
6   padding: 20px;
7   margin: 20px;
8 }
9 .use-content-box {
10  box-sizing: content-box;
11 }
12 .use-border-box {
13  box-sizing: border-box;
14 }
```



The image shows two cyan boxes illustrating the box model. The top box is labeled "Content-box model" and the bottom box is labeled "Border-box model".

<http://jsfiddle.net/koldovsky/e1984en9/1/>

Inherited Properties

- A parent box can contain one or more child boxes.
- A child can inherit CSS styles from a parent.
- Sample inherited property:

```
p { color: green }
```

```
<p>This paragraph has <em>emphasized text</em> in it.</p>
```

- Sample non-inherited property:

```
p { border: medium solid }
```

```
<p>This paragraph has <em>emphasized text</em> in it.</p>
```

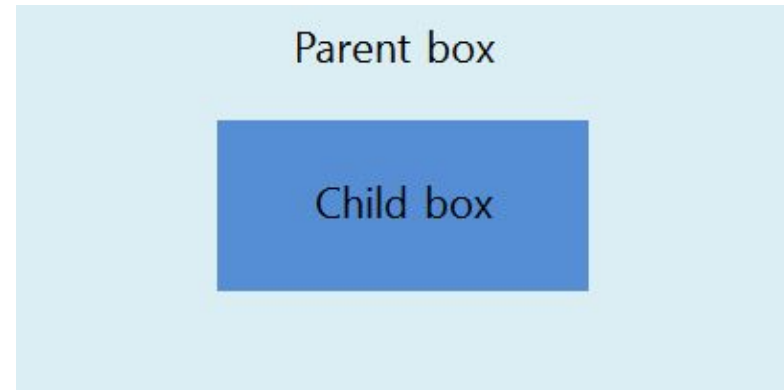
- Using **inherit** property:

```
/* make second-level headers green */
```

```
h2 { color: green; }
```

```
/* ...but leave those in the sidebar alone so they use their  
parent's color */
```

```
#sidebar h2 { color: inherit; }
```



Browser Default Styles

- Web browsers have default CSS styles for HTML elements, consider sample:

<http://plnkr.co/edit/Qgapgl8yuc328XV888Q8?p=preview>

- Also these styles are different for different browsers, so same markup may look different
- To ensure same markup looks the same it is recommended to use "reset" or "normalize" stylesheets (second is preferred):
 - Reset CSS: <http://meyerweb.com/eric/tools/css/reset/>
 - Normalize CSS: <http://necolas.github.io/normalize.css/>

Media Queries

- A **media query** consists of a media type and at least one expression that limits the style sheets' scope by using media features, such as width, height, and color.
- Media queries allow to create responsive websites
- Details: https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries

```
<!-- CSS media query on a link element -->  
<link rel="stylesheet" media="(max-width: 800px)" href="example.css" />
```

```
<!-- CSS media query within a stylesheet -->  
<style>  
  @media (max-width: 600px) {  
    .facet_sidebar {  
      display: none;  
    }  
  }  
</style>
```

- Sample: <http://plnkr.co/edit/xYTDjomz5JNAvpQjt6LZ?p=preview>
- Some issues with media queries: it's not so simple to reorder blocks

UI Challenges

- Developers have used `float` property for relative positioning of UI elements for years
- CSS3 Provides two new options:
 - **CSS3 Flexbox Box** model ideal for items that should resize or reposition themselves
 - **CSS3 Grid Layout** model good for complex layouts

CSS Flexbox Box Model

- Good for controls, toolbars, menus, and forms that resize and reposition automatically when the user changes the size of the browser window
- Browser takes the available space into account and calculates the dimensions for the user
- Enables relative sizes and positioning
- Good tutorial:
<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

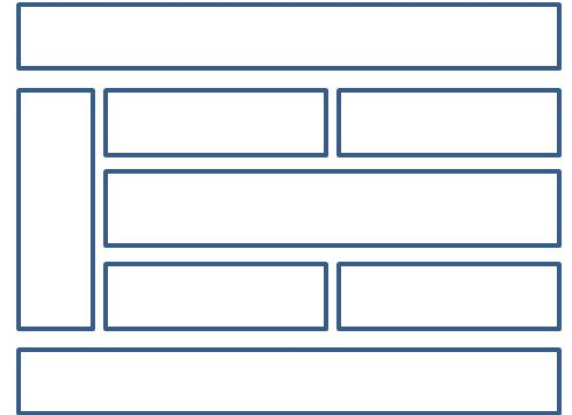
CSS Flexbox Model Reordering Sample

- <http://jsfiddle.net/koldovsky/jb5h57jw/>

```
1 <div id="flex">
2   <div id="a">A</div>
3   <div id="b">B</div>
4   <div id="c">C</div>
5 </div>
1
JAVASCRIPT
1 #flex { display: flex; }
2 #flex > #a { order: 2; }
3 #flex > #b { order: 1; }
4 #flex > #c { order: 3; }
5
BAC
```

CSS3 Grid Layout Model

- Gives developers greater control over complex layouts than the flexbox model
- Lets you control the design of sections or entire HTML-based documents using CSS3
- Grid layouts use columns, rows, and cells, but you can move blocks of content from one part of page or application to another by moving code lines in CSS



Multi-column Layout

- Create columns by dividing text across multiple columns
- Specify the amount of space that appears between columns (the gap)
- Make vertical lines (rules) appear between columns
- Define where columns break

Multi-column Layout

- Main CSS properties for creating multiple columns in an HTML document:
 - `column-count`: Sets the number of columns
 - Alternative: Use `columns` property with `column-count` and `column-width` properties
 - `column-gap`: Specifies the **gap** between the columns, known as the gutter or alley
 - `column-rule`: Creates a vertical line in the gap between columns and sets the width, style (single or double line, solid, dashed, 3D, etc.) and color of the rule

Multi-column Layout Example

<https://jsfiddle.net/koldovsky/4k1h7bg0/1/>

The image shows a screenshot of a web development tool interface. It is divided into three main sections: HTML, CSS, and a preview. The HTML section contains a long paragraph of Lorem Ipsum text. The CSS section contains a single rule for a `p` element, setting `column-count: 3;` along with vendor prefixes for `-ms-column-count`, `-webkit-column-count`, and `-moz-column-count`. The preview section shows the rendered result: the text from the HTML section is displayed in three columns. The first column contains the first two paragraphs, the second column contains the third and fourth paragraphs, and the third column contains the fifth and sixth paragraphs. The text is wrapped to fit the columns.

```
1 <p> HTML
2 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris gravida congue sem id venenatis. Donec blandit interdum sem. Duis eu risus ut erat commodo semper. Quisque nec blandit odio, quis tincidunt dui. Maecenas nec elit finibus magna sodales luctus eget nec massa. Suspendisse at dolor in ligula ullamcorper vehicula. Nullam pretium fermentum lacinia. Aliquam sodales nisl justo, nec finibus enim accumsan eget. Maecenas vulputate urna vitae tellus tincidunt vestibulum. Integer mi metus, consectetur eget consectetur sit amet, sodales id lectus. Aliquam commodo nisi id volutpat venenatis. Nulla quis mi laoreet, aliquet velit sed, egestas massa.
3
4 Praesent ac enim neque. Ut lorem ex, egestas in mattis vitae, sagittis aliquam ante. Morbi nec sem ultricies, suscipit diam vitae, fermentum est. In id lectus placerat, faucibus enim sed, gravida enim. In commodo eros at ligula malesuada, id vulputate orci feugiat. Proin id euismod nunc, et vestibulum purus. Suspendisse maximus sem sit amet libero consequat. et
1 JAVASCRIPT
```

```
1 p {column-count: 3;
2   -ms-column-count: 3;
3   -webkit-column-count: 3;
4   -moz-column-count: 3;}
```

1 CSS

1 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris gravida congue sem id venenatis. Donec blandit interdum sem. Duis eu risus ut erat commodo semper. Quisque nec blandit odio, quis tincidunt dui. Maecenas nec elit finibus magna sodales luctus eget nec massa. Suspendisse at dolor in ligula ullamcorper vehicula. Nullam pretium fermentum lacinia. Aliquam sodales nisl justo, nec finibus enim accumsan eget. Maecenas vulputate urna vitae tellus tincidunt vestibulum. Integer mi metus, consectetur eget

2 eu in nibh. In mollis eros et sollicitudin iaculis. Aliquam erat volutpat. Nulla vel risus sit amet nisi commodo mollis eu eget lacus. Sed volutpat mauris velit, sit amet dictum felis mollis a. Praesent vitae ante id purus scelerisque malesuada. Ut commodo volutpat iaculis. Maecenas sodales maximus neque non dignissim. Fusce vel ipsum at nibh facilisis condimentum eu in nunc. Phasellus efficitur, arcu ac convallis dignissim, nisl mi consequat elit, eu mattis mi nunc sed elit. Duis lacinia justo at nisi consequat semper. Ut aliquet

3 dolor vulputate, in dapibus risus faucibus. Aliquam erat volutpat. Donec iaculis mauris enim, non feugiat neque sodales in. Vivamus porta ex quis rhoncus sollicitudin. Etiam ut lacus non sem tristique aliquet sit amet vel sem. Suspendisse sed neque luctus, ultricies lacus nec, pharetra mauris. Praesent nec ligula at ex condimentum tempus vel non est. In gravida leo non urna suscipit finibus. Integer quis nibh eu urna auctor ullamcorper in vitae ipsum. Pellentesque mattis eleifend erat vitae laoreet. Etiam venenatis. urna in

Practice Task:

Explore <http://learnlayout.com/>

Use different approaches to create layouts



Learn CSS Layout

This site teaches the CSS fundamentals that are used in any website's layout.

I assume you already know what selectors, properties, and values are. And you probably know a thing or two about layout, though it may still be a rage-provoking activity for you. If you want to learn HTML and CSS from the beginning, you should check out [this tutorial](#). Otherwise, let's see if we can save you some fury on your next project.

Get Started

english español français deutsch dutch italiano
português (brasil) português (português) русский
فارسی 中文 正體中文 한국어 日本語

Hyphenation

- The process of connecting two words with a hyphen mark (-) or breaking words between syllables at the end of a line.
- CSS3 `hyphens` property controls hyphenation
- Values:
 - `auto`: Enables automatic hyphenation of words based on line-break opportunities within words or by a “language-appropriate hyphenation resource”
 - `manual`: Enables hyphenation of words based only on line-break opportunities within words
 - `none`: Prevents hyphenation

Language Declaration

- W3C requires a language declaration for correct automatic hyphenation to occur:

```
<!doctype html>
```

```
<html lang="en-us">
```

or

```
<html
```

```
xmlns="http://www.w3.org/1999/xhtml"
```

```
xml:lang="en" lang="en">
```

border-radius Property

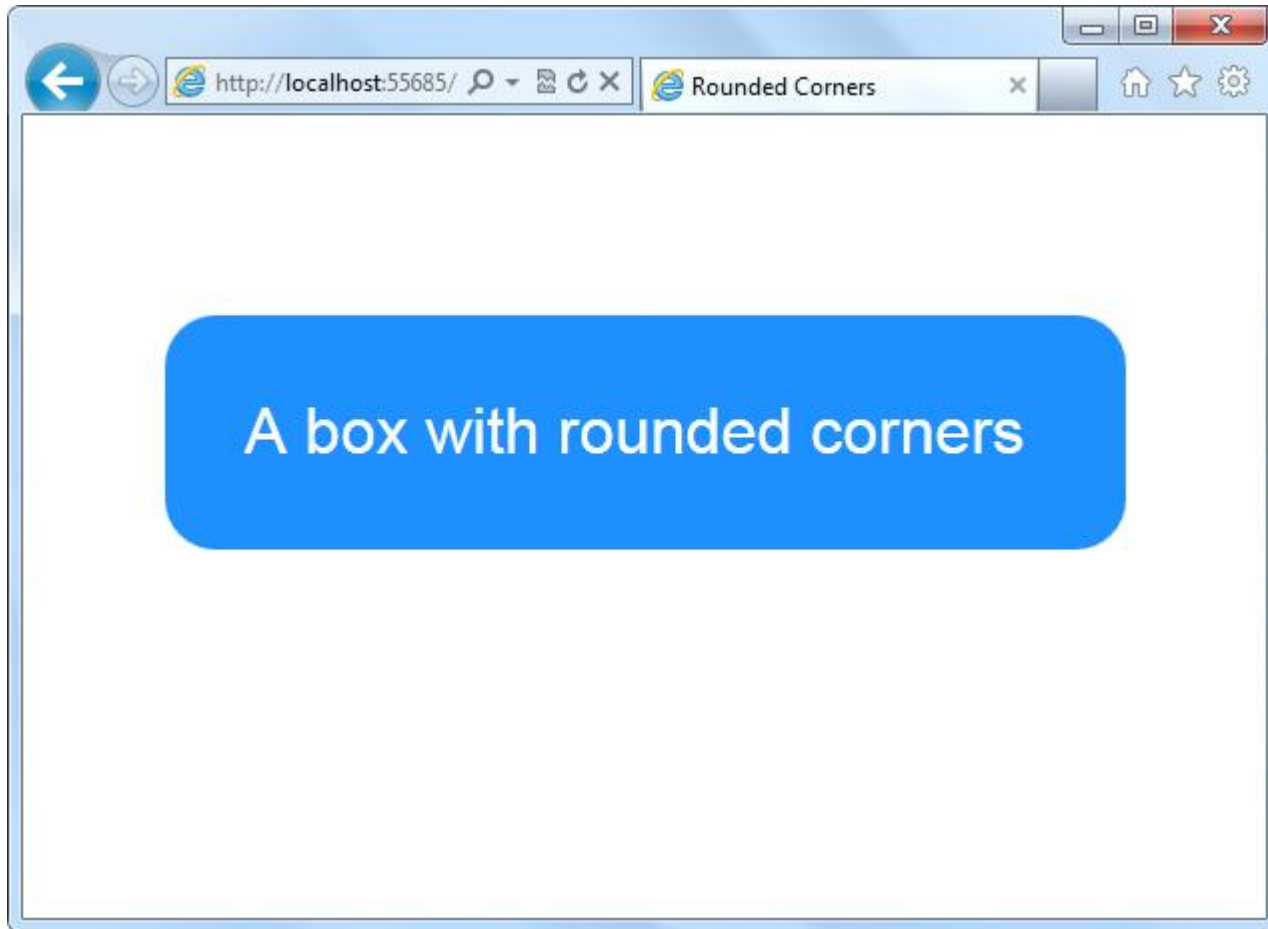
- Creates rounded corners around layout elements, like headers, footers, sidebars, graphics boxes, and outlines around images
- `border-radius` is a length, which is usually expressed in pixels or ems but can be a percentage

border-radius Example

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Rounded Corners</title>
  <style type="text/css">
div {
  padding: 40px 40px;
  background: dodgerblue;
  width: 400px;
  color: #fff;
  font-family: sans-serif;
  font-size: xx-large;
  border-radius: 25px;
  margin-left: auto;
  margin-right: auto;
  margin-top: 100px;
}
</style>
</head>

<body>
  <div>A box with rounded corners</div>
</body>
</html>
```

border-radius Example



box-shadow Property

- Creates drop shadows around layout elements
- CSS syntax for creating a shadow:
`box-shadow: h-shadow v-shadow blur spread color inset;`
- Required: *h-shadow* and *v-shadow* attributes set the horizontal and vertical position of the shadow in relation to the box
- Optional: *blur*, *spread*, *color*, and *inset*

box-shadow Example

```
<!doctype html>
<html>
<head>
<title>Rounded Corners</title>
<style type="text/css">
div
{
padding:40px 40px;
background:lightgreen;
width:400px;
color: #000;
font-family: sans-serif;
font-size: xx-large;
border-radius:25px;
-ms-border-radius:25px;
-moz-border-radius:25px;
-o-border-radius:25px;
-webkit-border-radius:25px;
box-shadow: 10px 10px 5px #808080;
margin-left:auto;
margin-right:auto;
margin-top: 100px;
}
</style>
</head>

<body>
<div>A box with a drop shadow</div>
</body>
</html>
```



A box with a drop shadow

Opacity and Transparency

- An **opaque** item does not let light pass through, whereas you can see through a **transparent** item.
- Syntax for applying a transparency to an image or other element:
`opacity: value`
- Value is a floating-point value between 0.0 (100% transparent) and 1.0 (100% opaque)

Transparency Example



Original



With transparency

Photo: © AVTG/iStockphoto

CSS Gradients

- ***Gradient*** is a smooth change of colors, within the same hue or starting with one color and ending with a different color
- Used for subtle shading within backgrounds, button embellishments, and more
- Created as methods to the CSS `background` property

Gradient Examples

Linear gradient: background:
linear-gradient(black, white);



A gradient from black to white

Radial gradient: radial-gradient(50% 50%, 70%
70%, #99CCFF, #3D5266);



A radial gradient

2D and 3D Transformations

- A **transform** is an effect that lets you change the size, shape, and position of an element.
- Transformations use the `transform` property.
 - **Methods:** `matrix`, `perspective`, `rotate`, `scale`, `skew`, `translate`
- To see the “action” of a transformation requires JavaScript; using only CSS shows the before and after effects of properties and their values.

Transformations Sample

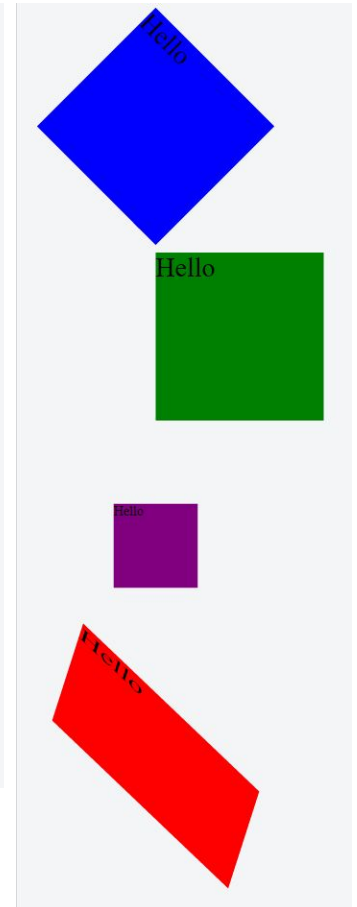
```
1 <div class="blue"><p>Hello</p></div>
2 <div class="green"><p>Hello</p></div>
3 <div class="purple"><p>Hello</p></div>
4 <div class="red"><p>Hello</p></div>
```

HTML ⚙️

```
1
```

JAVASCRIPT ⚙️

```
1 div {
2   width:100px;
3   height:100px;
4   margin:25px;
5 }
6 .blue {
7   background-color:blue;
8   transform:rotate(45deg);
9 }
10 .green {
11   background-color:green;
12   transform:translate(50px,0px );
13 }
14 .purple {
15   background-color:purple;
16   transform:scale(0.5);
17 }
18 .red {
19   background-color:red;
20   transform:skew(20deg,30deg) rotate(30deg);
21 }
```



<http://jsfiddle.net/koldovsky/2m2Zb/36/>

CSS Transition

- A **transition** is a change from one thing to another; in CSS, a transition is the change in an element from one style to another.
- In CSS3, the action of a transition renders onscreen—no JavaScript is needed!
- The `transition` property requires the CSS property to be acted upon during the transition.

Transition Sample

```
1 <div>
2   Transition is cool (hover over me)
3 </div>
```

HTML ⚙️

```
1
```

JAVASCRIPT ⚙️

```
1 div {
2   background:white;
3   border:1px solid #aaa;
4   margin:10px;
5   padding:10px;
6   width:330px;
7   transition:transform 0.7s ease-in-out;
8 }
9
10 div:hover {
11   cursor:pointer;
12   transform:scale(2) rotate(180deg);
13 }
```

Transition is cool (hover over me)

<http://jsfiddle.net/koldovsky/PkJaD/357/>

CSS Animation

CSS animations animates transitions between CSS styles to another.

Consist of two components: a style describing the CSS animation and a set of keyframes that indicate the start and end states of the animation's style, as well as possible waypoints.

There are three key advantages to CSS animations over traditional script-driven animation:

- Easy to use for simple animations.
- The animations run well, even under moderate system load. The rendering engine can use frame-skipping and other techniques to keep the performance as smooth as possible.
- Letting the browser control the animation sequence lets the browser optimize performance and efficiency by, for example, reducing the update frequency of animations running in tabs that aren't currently visible.

Some cool samples:

<http://webdesign.tutsplus.com/articles/15-inspiring-examples-of-css-animation-on-codepen-cms-23937>

Details: <https://css-tricks.com/almanac/properties/a/animation/>

Simple Animation Example

<pre>1 <div id="spin"> 2 <p>(^-^)</p> 3 </div> 4</pre>	<pre>HTML ⚙️ 1 @keyframes spinnerRotate 2 { 3 from{transform:rotate(0deg);} 4 to{transform:rotate(360deg);} 5 } 6 #spin{ 7 position: fixed; 8 top: 10px; 9 left: 10px; 10 animation-name: spinnerRotate; 11 animation-duration: 2s; 12 animation-iteration-count: infinite; 13 animation-timing-function: linear; 14 }</pre>
<pre>1</pre>	<pre>JAVASCRIPT ⚙️ (^-^)</pre>

<http://jsfiddle.net/koldovsky/e2tt2mao/56/>

Timing Functions

```
1 <div id="div1">linear</div>
2 <div id="div2">ease</div>
3 <div id="div3">ease-in</div>
4 <div id="div4">ease-out</div>
5 <div id="div5">ease-in-out</div>
```

HTML ⚙

```
12 padding-left: 20px;
13 margin: 10px 0;
14 color:white;
15 font: 14px Georgia;
16 vertical-align: middle;
17 background:#05f;
18 animation: width 5s infinite;
19 }
20
21 #div1 {animation-timing-function: linear;}
22 #div2 {animation-timing-function: ease;}
23 #div3 {animation-timing-function: ease-in;}
```

JAVASCRIPT ⚙

- linear
- ease
- ease-in
- ease-out
- ease-in-out

<https://jsfiddle.net/koldovsky/HDsw2/11/>

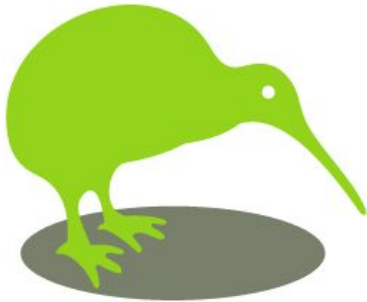
Details:

<https://www.smashingmagazine.com/2014/04/understanding-css-timing-functions/>

SVG Filters Support

- Small file sizes that compress well
- Scales to any size without losing clarity (except very tiny)
- Looks great on high-res displays
- An **SVG filter** is a set of operations that use CSS to style or otherwise modify an SVG graphic
- The enhanced graphic is displayed in a browser while the original graphic is left alone.

Sample: <http://codepen.io/chriscoyier/pen/evcBu>



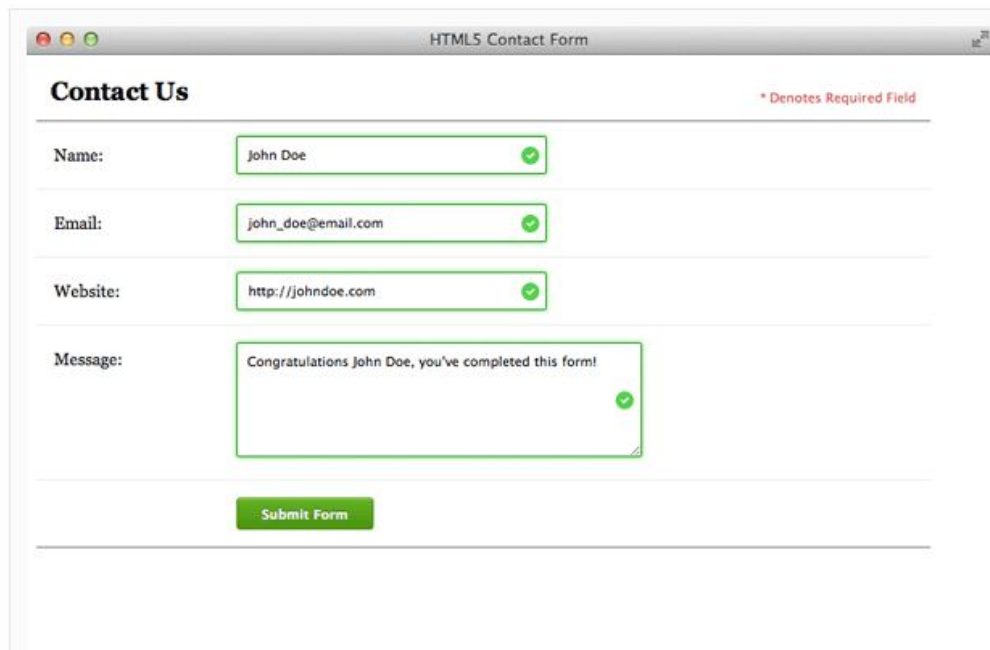
Step-by-step guide: <https://css-tricks.com/using-svg/>

Styling forms

<http://webdesign.tutsplus.com/tutorials/bring-your-forms-up-to-date-with-css3-and-html5-validation--webdesign-4738>

Step 16: Sit Back and Admire Your Beautiful HTML5 Form

Go ahead and take a look at your final product!



The screenshot shows a web browser window titled "HTML5 Contact Form". The form is titled "Contact Us" and includes a legend: "* Denotes Required Field". The form contains four input fields, each with a green border and a green checkmark icon on the right, indicating successful validation:

- Name: John Doe
- Email: john_doe@email.com
- Website: http://johndoe.com
- Message: Congratulations John Doe, you've completed this form!

At the bottom of the form is a green "Submit Form" button.

Styling Tables

http://www.w3schools.com/css/css_table.asp

CSS Tables

« Previous

Next Chapter »

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

Practice Task

Advanced Topics

CSS Regions

- Feature allows developers to dynamically flow content across multiple boxes, or containers, in HTML documents with fluid layouts
- Content adjusts and displays properly whether viewed on large or small

Content Flow with CSS Regions

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris id dui nisl, vitae congue est. Nulla molestie sollicitudin ligula, eu suscipit lorem. Sed molestie posuere suscipit. Nullam ornare est sed nisl. Lacinia a lobortis enim mollis.

1

Ut tempus enim sit amet lorem feugiat sit amet eleifend velit pretium. Nulla facilisi. Phasellus ultricies lobortis gravida. Praesent pharetra ligula vitae nunc lacinia quis accumsan eros tincidunt. Sed tellus massa euismod ullamcorper vitae velit.

2

Morbi viverra scelerisque suscipit. Quisque purus metus, ullamcorper eu scelerisque eu, mollis a ipsum. Sed ut augue accumsan leo pulvinar ultrices at et ipsum. Curabitur sit amet mi eu justo congue condimentum. Sed nec nisi.

3

Vestibulum semper nisl quis ligula scelerisque facilisis. Aliquam pretium, justo ut egestas commodo, est ligula egestas tortor, id ullamcorper nibh odio ultricies urna. Donec dictum mi eget eros sagittis in tincidunt eros vestibulum. Sed at mauris non mauris semper scelerisque. Sed congue semper elit in viverra. In eget sapien neque, a tempor augue. In enim quam, sagittis at interdum nec, mollis eget est.

CSS Exclusions

- Formerly referred to as positioned floats
- Enables positioning of images, text, and boxes anywhere in an HTML document and wrapping of text completely around these elements
- Can control the position of a float precisely, at a specified distance from the top, bottom, left, or right sides of a container

CSS Exclusions Example 1

This is an element formatted into 2 columns using CSS3 Multicolumn

On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look. You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly. To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template. On the Insert tab, the galleries include items that

This is a Positioned Float.

Click to drag it around the page.

are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look. You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly. To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template. On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building

Screen shot from Internet Explorer 10 Test Drive Web page

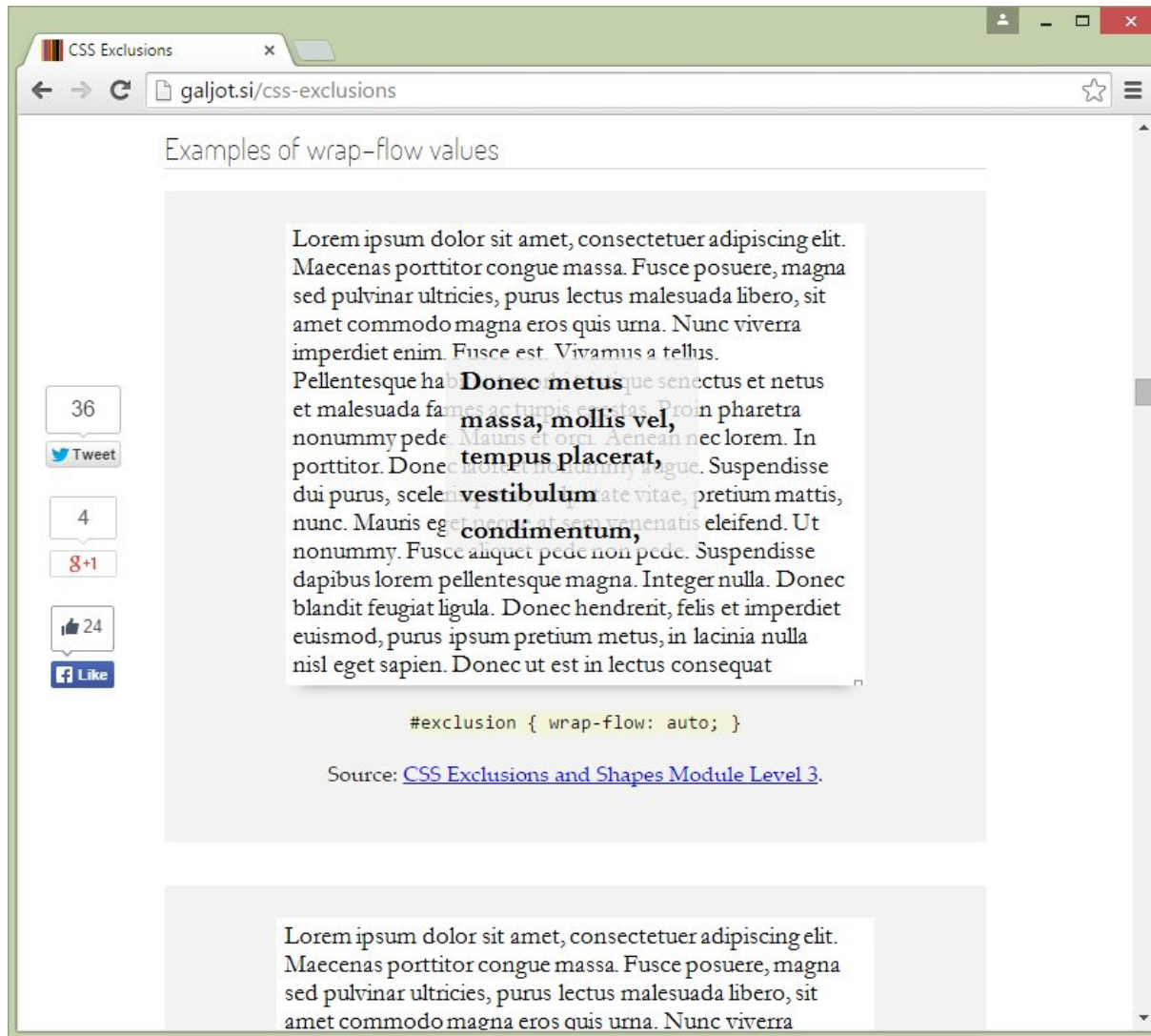
CSS Exclusions Properties

- `wrap-flow: both` displays content on all sides of the exclusion
- `wrap-flow: clear` displays content above and below the exclusion but leaves the sides blank
- `shape-inside` and `shape-outside` define the content and the general shape of an exclusion, respectively
- `-ms-` vendor prefix required for Internet Explorer 10; Exclusions not supported in Internet Explorer 9

CSS Exclusions Example 2

Vestibulum semper nisl quis ligula scelerisque facilisis. Aliquam pretium, justo ut egestas commodo, est ligula egestas tortor, id ullamcorper nibh odio ultricies urna. Donec dictum mi eget eros sagittis in tincidunt eros vestibulum. Sed at mauris non mauris semper scelerisque. Sed congue semper elit in viverra. In eget sapien neque, a tempor augue. In enim quam, sagittis at interdum nec, mollis eget est. Duis ac nulla libero. Sed cursus laoreet tortor nec varius. Aenean aliquet neque quis velit ornare at commodo a tortor tempus. Nunc blandit turpis nisi. Aliquam metus mi, gravida sit amet gravida eget, dapibus vitae neque. Praesent tempus pulvinars ipsum, vel hendrerit enim placerat non. Donec scelerisque tincidunt diam et condimentum. Donec feugiat purus sed diam tempus elementum in eu elit. Cras euismod, erat at interdum dignissim, velit sapien euismod turpis, at pretium urna metus vitae metus. Cras blandit, sem ut bibendum aliquam, lacus lacus aliquet ante, non fringilla dui sapien ut nibh. Donec ac magna tortor. Etiam diam nulla, rhoncus ac bibendum sit amet, porttitor a quam. Fusce at leo in diam pellentesque suscipit. Quisque mattis facilisis massa, in commodo felis pretium ac.

CSS Exclusions Step-by-step



The screenshot shows a web browser window with the address bar containing "galjot.si/css-exclusions". The page title is "CSS Exclusions". The main content area is titled "Examples of wrap-flow values". It features a large text block with several words highlighted in bold: "Donec metus", "massa, mollis vel,", "tempus placerat,", "vestibulum", and "condimentum,". To the left of this text block are social media sharing buttons: a counter for 36, a "Tweet" button, a counter for 4, a "g+1" button, a counter for 24, and a "Like" button. Below the text block is a code snippet:

```
#exclusion { wrap-flow: auto; }
```

 and a source link: [Source: CSS Exclusions and Shapes Module Level 3.](#)

border-radius Property, Single Corners

- Rounding a single corner of a box:
 - `border-top-left-radius`
 - `border-top-right-radius`
 - `border-bottom-right-radius`
 - `border-bottom-left-radius`

Single rounded corner, top left

Single rounded corner, top right

Single rounded corner, bottom left

Single rounded corner, bottom right

CSS Gradient Methods

- CSS3 gradient methods:
 - `linear-gradient`: Creates a gradient from top to bottom or vice versa, or from corner to corner
 - `radial-gradient`: Creates a gradient that radiates out from a central point
 - `repeating-linear-gradient`: Creates a repeating linear gradient, which results in straight bands of gradient color
 - `repeating-radial-gradient`: Creates a repeating radial gradient, which results in circular bands of gradient color

Gradient Color Interpolation and Color Stops

- CSS gradients support color interpolation in the alpha color space
 - Part of the red blue green alpha (RGBA) color model
- Can specify multiple color stops, with an RGBA color and position for each one
- Example of the use of rgba colors:
`linear-gradient(to right,
rgba(255, 255, 255, 0)`

2D Translation

- To **translate** an element means to move it without rotating, skewing, or otherwise turning the image.
- Use the `translate()` method in CSS and provide x- and y-axis values to position the element relative to its original or default position.
 - x-axis value specifies the left position of the element
 - y-axis value specifies the top position.

2D Translation Example

```
<!doctype html>
<html>
<head>
<style type="text/css">
div
{
padding:20px 20px;
background:limegreen;
width:150px;
height:75px;
color:#fff;
font-family:sans-serif;
font-size:xx-large;
}
div#div2
{
transform:translate(100px,50px);
-ms-transform:translate(100px,50px); /* IE 9 */
-moz-transform:translate(100px,50px); /* Firefox */
-webkit-transform:translate(100px,50px); /* Safari and Chrome */
-o-transform:translate(100px,50px); /* Opera */
}
</style>
</head>

<body>
<div>Original position</div>
<div id="div2">Translated position</div>
</body>
</html>
```

2D Translation Example



Original
position



Translated
position

2D Scaling

- To **scale** an element is to increase or decrease its size.
- Use the `scale()` method in CSS and provide x-axis (width) and y-axis (height) values.
- The example on the following two slides increases the width of the element two times its original size, and increases the height four times its original size:

```
transform: scale(2, 4);
```

2D Scaling Example

```
<!doctype html>
<html>
<head>
<style type="text/css">
div
{
padding:10px 10px;
background:limegreen;
width:100px;
height:50px;
color:#fff;
font-family:sans-serif;
font-size: x-large;
}
div#div2
{
margin:200px;
transform:scale(2,4);
-ms-transform:scale(2,4); /* IE 9 */
-moz-transform:scale(2,4); /* Firefox */
-webkit-transform:scale(2,4); /* Safari and Chrome */
-o-transform:scale(2,4); /* Opera */
}
</style>
</head>

<body>
<div>Original element</div>
<div id="div2">Scaled element</div>
</body>
</html>
```

2D Scaling Example



Original
element



Scaled
element

2D Rotation

- To **rotate** an element turns it clockwise by a specified number of degrees.
- Use the `rotate()` method in CSS and specify the degrees of rotation.
- The example on the following two slides rotates an element by 30 degrees in the 2D plane:

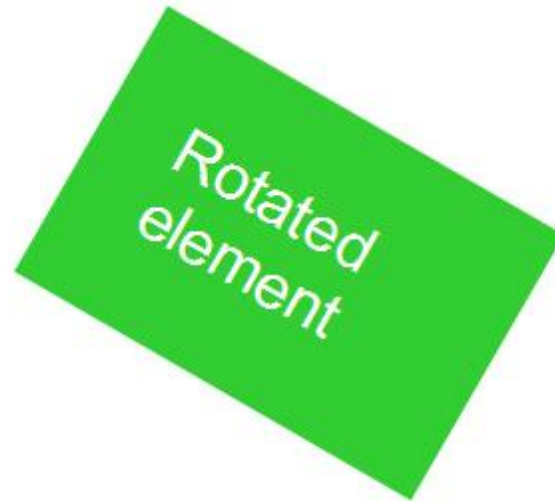
```
transform: rotate(30deg);
```


2D Rotation Example

```
<!doctype html>
<html>
<head>
<style type="text/css">
div
{
padding:40px 40px;
background:limegreen;
width:150px;
color:#fff;
font-family:sans-serif;
font-size: xx-large;
}
div#div2
{
margin:100px;
transform:rotate(30deg);
-ms-transform:rotate(30deg); /* IE 9 */
-moz-transform:rotate(30deg); /* Firefox */
-webkit-transform:rotate(30deg); /* Safari and Chrome */
-o-transform:rotate(30deg); /* Opera */
}
</style>
</head>

<body>
<div>Original element</div>
<div id="div2">Rotated element</div>
</body>
</html>
```

2D Example



3D Rotation

- 3D rotation uses the `rotateX()` and `rotateY()` methods.
 - `rotateX()`: Element rotates around its x-axis
 - `rotateY()`: Element rotates around its y-axis

2D Skewing

- To **skew** an element is to stretch it in one or more directions.
- Use the `skew()` method and provide x-axis and y-axis values, in degrees, to create an angular shape.
- The example on the following two slides turns an element 20 degrees around the x-axis and 30 degrees around the y-axis:

```
transform: skew(20deg, 30deg);
```

2D Skewing Example

```
<!doctype html>
<html>
<head>
<style type="text/css">
div
{
padding:40px 40px;
background:limegreen;
width:150px;
color:#fff;
font-family:sans-serif;
font-size:xx-large;
}
div#div2
{
margin:100px;
transform:skew(20deg,30deg);
-ms-transform:skew(20deg,30deg); /* IE 9 */
-moz-transform:skew(20deg,30deg); /* Firefox */
-webkit-transform:skew(20deg,30deg); /* Safari and Chrome */
-o-transform:skew(20deg,30deg); /* Opera */
}
</style>
</head>

<body>
<div>Original element</div>
<div id="div2">Skewed element</div>
</body>
</html>
```

2D Skewing Example



3D Skewing

- 3D skewing uses the `skewX()` and `skewY()` methods to skew an element around its x-axis and y-axis, respectively.
- As an example, the following code skews an element 45 degrees:

```
transform: skewX(45deg);
```

3D Perspective

- The CSS3 **3D perspective** property defines how a browser renders the depth of a 3D transformed element.
- The property takes on a number value: lower values (in the range of 1 to 1000) create a more pronounced effect than higher values.

Transition Example

```
<!doctype html>
<html>
<head>
<style type="text/css">

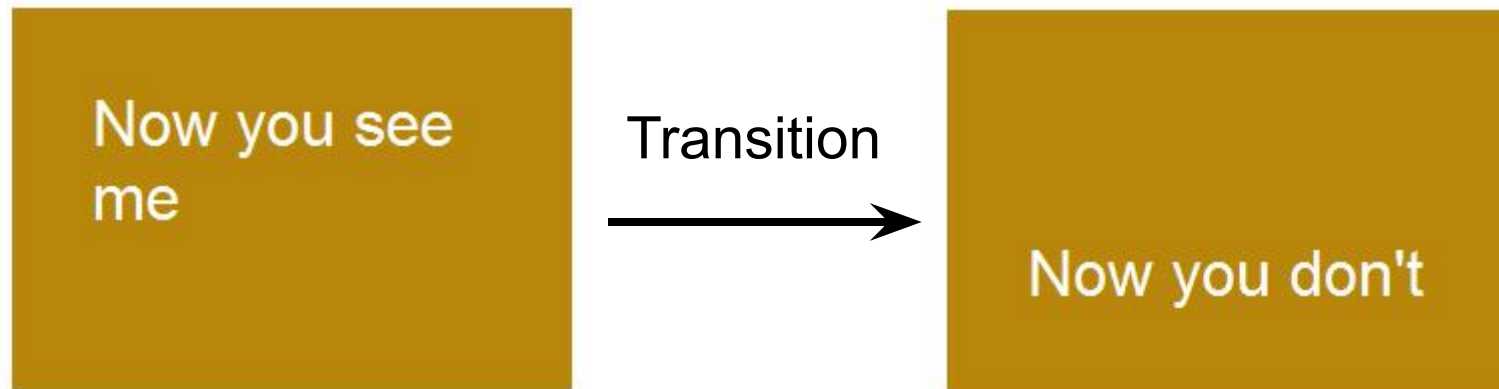
#wrapper {
  padding: 40px 40px;
  background: #88860B;
  width: 200px;
  color: #fff;
  font-family: sans-serif;
  font-size: xx-large;
  margin-left: auto;
  margin-right: auto;
  margin-top: 100px;
  -webkit-transition-property:opacity;
  -webkit-transition-duration:3s;
  -webkit-transition-delay:1s;
  -webkit-transition-timing-function:linear;
}
```

```
#wrapper #before, #wrapper:hover #after { -webkit-opacity: 1; }
#wrapper:hover #before, #wrapper #after { -webkit-opacity: 0; }

</style>
</head>

<body>
<div id="wrapper">
  <div id="before">Now you see me</div>
  <div id="after">Now you don't</div>
</div>
</body>
</html>
```

Transition Example



Animation (Continued)

- Specify a CSS style within the @keyframes rule
- An example of a rule for a fadeout:

```
@keyframes fadeout {  
  from { opacity: 1; }  
  to   { opacity: 0; }  
}
```

Animation (Continued)

- Code snippet that configures animation properties for a fadeout:

```
div { animation-duration: 3s;  
animation-delay: 0s;  
animation-timing-function: ease; }  
div:hover { animation-name: fadeout; }
```

SVG Filters

- An **SVG filter** is a set of operations that use CSS to style or otherwise modify an SVG graphic.
- The enhanced graphic is displayed in a browser while the original graphic is left alone.

SVG Filters

- feBlend
- feColorMatrix
- feComponentTransfer
- feComposite
- feConvolveMatrix
- feDiffuseLighting
- feDisplacementMap
- feFlood
- feGaussianBlur
- feImage
- feMerge
- feMorphology

SVG Filters Gaussian Blur Example

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <title>SVG Gaussian Blur Example</title>
  <style type="text/css">
</style>
</head>
<body>
<svg>
  <defs>
    <filter id="a1" x="0" y="0">
      <feGaussianBlur in="SourceGraphic" stdDeviation="20" />
    </filter>
  </defs>
  <rect width="150" height="150" stroke="plum"
    stroke-width="3" fill="plum" filter="url(#a1)" />
</svg>
</body>
</html>
```



SVG Filters Offset Example

```
<!doctype html>
<html>
<body>
<svg>
  <defs>
    <filter id="i1" x="0" y="0">
      <feOffset dx="5" dy="5" />
    </filter>
  </defs>
  <rect width="150" height="150" fill="grey"
    filter="url(#i1)" />
  <rect width="150" height="150" fill="plum" />
</svg>
</body>
</html>
```



Canvas

- Use canvas to draw pixel-based shapes.
- The `canvas` element accepts only two attributes—`height` and `width`.
- You can use most CSS properties to style the `canvas` element, adding color, gradients, pattern fills, transformation, animation, and much more.

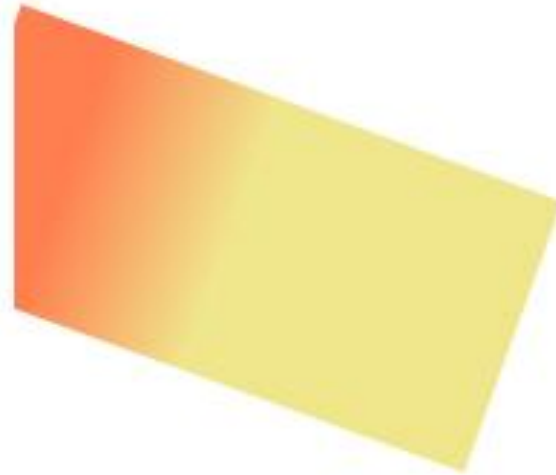
Canvas Example 1

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Canvas Example</title>
  </head>
  <script>
    function f1() {
      var canvas =
      document.getElementById("smlRectangle");
      context = canvas.getContext("2d");
      var grd = context.createLinearGradient(0, 0, 150, 0);
      grd.addColorStop(0.3, "coral");
      grd.addColorStop(0.7, "khaki");
      context.fillStyle = grd;
      context.fillRect(10, 20, 200, 100);
    }
  </script>
</head>
<body onload = "f1();">
<canvas id="smlRectangle" height='300' width='500'>
</canvas>
</body>
</html>
```



Canvas Example 2

- `context.rotate(20*Math.PI/180);`



Canvas Example 3

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Canvas Text</title>
</head>
<body>
  <canvas id="myText" width="400" height="250" style="border:3px solid #0000FF;">
  </canvas>
  <script type="text/javascript">
    var canvas = document.getElementById("myText");
    context = canvas.getContext("2d");
    context.font = "30px Arial";
    context.strokeText("Canvas-generated text", 40, 120);
  </script>
</body>
</html>
```



Contacts

Europe Headquarters

52 V. Velykoho Str.
Lviv 79053, Ukraine

Tel: +380-32-240-9090

Fax: +380-32-240-9080

E-mail: info@softserveinc.com

Website: www.softserveinc.com

US Headquarters

12800 University Drive, Suite 250
Fort Myers, FL 33907, USA

Tel: 239-690-3111

Fax: 239-690-3116

Thank You!