Styling HTML with CSS

Creating selector using property and value

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- The CSS element selector
- The CSS id selector
- The CSS class selector
- The CSS Universal Selector
- The CSS grouping selector

The CSS id selector

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

The id of an element is 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.

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

The CSS class selector

The class selector selects HTML elements with a specific class attribute.

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

```
.container_main{
    width: 960px;
    margin: 0 auto;
}
```

You can also specify that only specific HTML elements should be affected by a class.

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

HTML elements can also refer to more than one class.

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

The CSS Universal selector

The universal selector (*) selects all HTML elements on the page.

```
* {
    text-align: center;
    color: blue;
}
```

The CSS grouping selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

```
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.

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

Creative a Responsive Design with CSS3 media query

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 Introduced Media Queries

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

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

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

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

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.

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.

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):

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

Add a Breakpoint

We can add a breakpoint where certain parts of the design will behave differently on each side of the breakpoint.

```
.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:

```
/* 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):

Note that the two sets of classes are almost identical, the only difference is the name (col- and col-s-):

```
/* For mobile phones: */
[class*="col-"] {
    width: 100%;
}
@media only screen and (min-width: 600px) {
    /* For tablets: */
    .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) {
    /* 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%;}
}
```

Typical Device Breakpoints

There are tons of screens and devices with different heights and widths, so it is hard to create an exact breakpoint for each device. To keep things simple you could target five groups:

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}

/* Large devices (laptops/desktops, 992px and up) */
```

```
@media only screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops, 1200px and up) */
@media only screen and (min-width: 1200px) {...}
```

Apply border to box element

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

CSS 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 border).

```
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;}
```

CSS 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:

```
p.one {
    border-style: solid;
    border-width: 5px;
}
p.two {
    border-style: solid;
    border-width: medium;
}
p.three {
    border-style: dotted;
    border-width: 2px;
}
  p.four {
    border-style: dotted;
    border-width: thick;
}
```

```
p.one {
    border-style: solid;
    border-width: 5px 20px; /* 5px top and bottom, 20px on the sides */
}

p.two {
    border-style: solid;
    border-width: 20px 5px; /* 20px top and bottom, 5px on the sides */
}
```

```
p.three {
    border-style: solid;
    border-
width: 25px 10px 4px 35px; /* 25px top, 10px right, 4px bottom and 35px le
ft */
}
```

CSS 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)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- transparent

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

```
p.one {
    border-style: solid;
    border-color: red;
}
p.two {
    border-style: solid;
    border-color: green;
}
p.three {
    border-style: dotted;
    border-color: blue;
}
```

```
p.one {
    border-style: solid;
    border-
color: red green blue yellow; /* red top, green right, blue bottom and yel
low left */
```

```
}
```

```
p.one {
    border-style: solid;
    border-color: #ff0000; /* red */
}
```

```
p.one {
    border-style: solid;
    border-color: rgb(255, 0, 0); /* red */
}
```

```
p.one {
    border-style: solid;
    border-color: rgba(255, 0, 0,0.5); /* red */
}
```

CSS Border - Shorthand Property

Like you saw in the previous page, 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

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

```
p {
    border-bottom: 6px solid red;
```

```
    background-color: lightgrey;
}
```

CSS Rounded Borders

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

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

CSS Padding

The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are 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

Note: Negative values are not allowed.

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

Padding - Shorthand Property

padding: 25px 50px 75px 100px;

- top padding is 25px
- right padding is 50px
- bottom padding is 75px
- left padding is 100px

padding: 25px 50px 75px;

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

padding: 25px 50px;

- top and bottom paddings are 25px
- right and left paddings are 50px

padding: 25px;

- all four paddings are 25px

Padding and Element Width

The CSS width property specifies the width of the element's content area. The content area is the portion inside the padding, border, and margin of an element (the box model).

So, if an element has a specified width, the padding added to that element will be added to the total width of the element. This is often an undesirable result.

Example

Here, the element is given a width of 300px. However, the actual width of the element will be 350px (300px + 25px of left padding + 25px of right padding):

```
div {
    width: 300px;
    padding: 25px;
}
```

To keep the width at 300px, no matter the amount of padding, you can use the box-sizing property. This causes the element to maintain its width; if you increase the padding, the available content space will decrease.

```
div {
    width: 300px;
    padding: 25px;
    box-sizing: border-box;
}
```

CSS Margins

The CSS margin properties are used to create space around elements, outside of any defined borders.

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

Margin - Individual Sides

```
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: 25px 50px 75px 100px;

- top margin is 25px
- right margin is 50px
- bottom margin is 75px
- left margin is 100px

margin: 25px 50px 75px;

- top margin is 25px
- right and left margins are 50px
- bottom margin is 75px

margin: 25px 50px;

- top and bottom margins are 25px
- right and left margins are 50px

margin: 25px;

- all four margins are 25px

Apply Position to box

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

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

There are five different position values:

- 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: 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.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
    div.relative {
      position: relative;
      left: 30px;
      border: 3px solid #73AD21;
    }
    </style>
  </head>
  <body>
    <h2>position: relative;</h2>
    <p>An element with position: relative; is positioned relative to its n
ormal position:</p>
    <div class="relative">
      This div element has position: relative;
    </div>
  </body>
</html>
```

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:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
    div.fixed {
```

```
      position: fixed;
      bottom: 0;
      right: 0;
      width: 300px;
      border: 3px solid #73AD21;
    }
    </style>
  </head>
  <body>
    <h2>position: fixed;</h2>
    <p>An element with position: fixed; is positioned relative to the view
port, which means it always stays in the same place even if the page is sc
rolled:</p>
    <div class="fixed">
    This div element has position: fixed;
    </div>
  </body>
</html>
```

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:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      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;
      }
    </style>
  </head>
  <body>
    <h2>position: absolute;</h2>
    <p>An element with position: absolute; is positioned relative to the n
earest positioned ancestor (instead of positioned relative to the viewport
, like fixed):</p>
    <div class="relative">This div element has position: relative;
      <div class="absolute">This div element has position: absolute;</div>
    </div>
  </body>
</html>
```

CSS 2D Transforms

CSS transforms allow you to move, rotate, scale, and skew elements.

CSS 2D Transforms Methods

With the CSS transform property you can use the following 2D transformation methods:

- translate()
- rotate()
- scaleX()
- scaleY()
- scale()

- skewX()
- skewY()
- skew()
- matrix()

The translate() Method

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

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

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 300px;
        height: 100px;
        background-color: yellow;
        border: 1px solid black;
        -ms-transform: translate(50px,100px); /* IE 9 */
        transform: translate(50px,100px); /* Standard syntax */
      }
    </style>
  </head>
  <body>
    <h1>The translate() Method</h1>
    <p>The translate() method moves an element from its current position:</p>
    <div>
    This div element is moved 50 pixels to the right, and 100 pixels down from its current position.
    </div>
  </body>
</html>
```

The rotate() Method

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

The following example rotates the element clockwise with 20 degrees:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 300px;
        height: 100px;
        background-color: yellow;
        border: 1px solid black;
      }
      div#myDiv {
        -ms-transform: rotate(20deg); /* IE 9 */
        transform: rotate(20deg); /* Standard syntax */
      }
    </style>
  </head>
  <body>
    <h1>The rotate() Method</h1>
    <p>The rotate() method rotates an element clockwise or counter-
clockwise.</p>
    <div>
    This a normal div element.
    </div>
    <div id="myDiv">
      This div element is rotated clockwise 20 degrees.
    </div>
  </body>
</html>
```

Using negative values will rotate the element counter-clockwise.

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 element to be two times of its original width, and three times of its original height:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        margin: 150px;
        width: 200px;
        height: 100px;
        background-color: yellow;
        border: 1px solid black;
        -ms-transform: scale(2,3); /* IE 9 */
        transform: scale(2,3); /* Standard syntax */
      }
    </style>
  </head>
  <body>
    <h1>The scale() Method</h1>
    <p>The scale() method increases or decreases the size of an element.</
p>
    <div>
    This div element is two times of its original width, and three times o
f its original height.
    </div>
  </body>
</html>
```

The scaleX() Method

The scaleX() method increases or decreases the width of an element.

The following example increases the element to be two times of its original width:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        margin: 150px;
        width: 200px;
        height: 100px;
        background-color: yellow;
        border: 1px solid black;
        -ms-transform: scaleX(2); /* IE 9 */
        transform: scaleX(2); /* Standard syntax */
      }
    </style>
  </head>
  <body>
    <h1>The scaleX() Method</h1>
    <p>The scaleX() method increases or decreases the width of an element.
</p>
    <div>
    This div element is two times of its original width.
    </div>
  </body>
</html>
```

The scaleY() Method

The scaleY() method increases or decreases the height of an element.

The following example increases the element to be three times of its original height:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        margin: 150px;
        width: 200px;
        height: 100px;
```

```
        background-color: yellow;
        border: 1px solid black;
        -ms-transform: scaleY(3); /* IE 9 */
        transform: scaleY(3); /* Standard syntax */
      }
    </style>
  </head>
  <body>
    <h1>The scaleY() Method</h1>
    <p>The scaleY() method increases or decreases the height of an element
.</p>
    <div>
    This div element is three times of its original height.
    </div>
  </body>
</html>
```

The skewX() Method

The skewX() method skews an element along the X-axis by the given angle.

The following example skews the element 20 degrees along the X-axis:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 300px;
        height: 100px;
        background-color: yellow;
        border: 1px solid black;
      }

      div#myDiv {
        -ms-transform: skewX(20deg); /* IE 9 */
        transform: skewX(20deg); /* Standard syntax */
      }
    </style>
```

```
  </head>
  <body>
    <h1>The skewX() Method</h1>
    <p>The skewX() method skews an element along the X-
axis by the given angle.</p>
    <div>
      This a normal div element.
    </div>
    <div id="myDiv">
      This div element is skewed 20 degrees along the X-axis.
    </div>
  </body>
</html>
```

The skewY() Method

The skewY() method skews an element along the Y-axis by the given angle.

The following example skews the element 20 degrees along the Y-axis:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 300px;
        height: 100px;
        background-color: yellow;
        border: 1px solid black;
      }

      div#myDiv {
        -ms-transform: skewY(20deg); /* IE 9 */
        transform: skewY(20deg); /* Standard syntax */
      }
    </style>
  </head>
  <body>
```

```
    <h1>The skewY() Method</h1>
    <p>The skewY() method skews an element along the Y-
axis by the given angle.</p>
    <div>
      This a normal div element.
    </div>
    <div id="myDiv">
      This div element is skewed 20 degrees along the Y-axis.
    </div>
  </body>
</html>
```

The skew() Method

The skew() method skews an element along the X and Y-axis by the given angles.

The following example skews the element 20 degrees along the X-axis, and 10 degrees along the Y-axis:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 300px;
        height: 100px;
        background-color: yellow;
        border: 1px solid black;
      }

      div#myDiv {
        -ms-transform: skew(20deg,10deg); /* IE 9 */
        transform: skew(20deg,10deg); /* Standard syntax */
      }
    </style>
  </head>
  <body>
    <h1>The skew() Method</h1>
```

```
    <p>The skew() method skews an element into a given angle.</p>
    <div>
      This a normal div element.
    </div>
    <div id="myDiv">
      This div element is skewed 20 degrees along the X-
axis, and 10 degrees along the Y-axis.
    </div>
  </body>
</html>
```

If the second parameter is not specified, it has a zero value. So, the following example skews the element 20 degrees along the X-axis:

The matrix() Method

The matrix() method combines all the 2D transform methods into one.

The matrix() method take six parameters, containing mathematic functions, which allows you to rotate, scale, move (translate), and skew elements.

The parameters are as follow:
matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

```
<!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 */
        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 */
```

```
        transform: matrix(1, 0, 0.5, 1, 150, 0); /* Standard syntax */
      }
    </style>
  </head>
  <body>
    <h1>The matrix() Method</h1>
    <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>
```

CSS Transitions

CSS Transitions

CSS transitions allows you to change property values smoothly, over a given duration.

you will learn about the following properties:

- transition
- transition-delay
- transition-duration

How to Use CSS 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 element. The element has also specified a transition effect for the width property, with a duration of 2 seconds:

```
div {
        width: 100px;
        height: 100px;
        background: red;
        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 element:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 100px;
        height: 100px;
        background: red;
        transition: width 2s;
      }
      div:hover {
        width: 300px;
      }
    </style>
  </head>
  <body>
    <h1>The transition Property</h1>
    <p>Hover over the div element below, to see the transition effect:</p>
    <div></div>
    <p><b>Note:</b> This example does not work in Internet Explorer 9 and
earlier versions.</p>
```

```
  </body>
</html>
```

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:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 100px;
        height: 100px;
        background: red;
        transition: width 2s, height 4s;
      }
      div:hover {
        width: 300px;
        height: 300px;
      }
    </style>
  </head>
  <body>
    <h1>The transition Property</h1>
    <p>Hover over the div element below, to see the transition effect:</p>
    <div></div>
    <p><b>Note:</b> This example does not work in Internet Explorer 9 and
earlier versions.</p>
  </body>
</html>
```

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:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 100px;
        height: 100px;
        background: red;
        transition: width 3s;
        transition-delay: 1s;
      }
      div:hover {
        width: 300px;
      }
    </style>
  </head>
  <body>
    <h1>The transition-delay Property</h1>
    <p>Hover over the div element below, to see the transition effect:</p>
    <div></div>
    <p><b>Note:</b> The transition effect has a 1 second delay before star
ting.</p>
    <p><b>Note:</b> This example does not work in Internet Explorer 9 and
earlier versions.</p>
  </body>
</html>
```

Transition + Transformation

The following example adds a transition effect to the transformation:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 100px;
        height: 100px;
        background: red;
```

```
        transition: width 2s, height 2s, transform 2s;
      }
      div:hover {
        width: 300px;
        height: 300px;
        transform: rotate(180deg);
      }
    </style>
  </head>
  <body>
    <h1>Transition + Transform</h1>
    <p>Hover over the div element below:</p>
    <div></div>
    <p><b>Note:</b> This example does not work in Internet Explorer 9 and
earlier versions.</p>
  </body>
</html>
```

creating animation with CSS

CSS allows animation of HTML elements without using JavaScript or Flash!

In this chapter you will learn about the following properties:

- @keyframes
- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode
- animation

What are CSS Animations?

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

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

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

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

The @keyframes Rule

When 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 element. The animation will last for 4 seconds, and it will gradually change the background-color of the element from "red" to "yellow":

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 100px;
        height: 100px;
        background-color: red;
        animation-name: example;
        animation-duration: 4s;
      }
      @keyframes example {
        from {background-color: red;}
        to {background-color: yellow;}
      }
    </style>
  </head>
  <body>
    <p><b>Note:</b> This example does not work in Internet Explorer 9 and
earlier versions.</p>
    <div></div>
    <p><b>Note:</b> When an animation is finished, it changes back to its
original style.</p>
  </body>
```

```
</html>
```

Note: The animation-duration property defines how long time an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).

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 element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 100px;
        height: 100px;
        background-color: red;
        animation-name: example;
        animation-duration: 4s;
      }
      @keyframes example {
        0%   {background-color: red;}
        25%  {background-color: yellow;}
        50%  {background-color: blue;}
        100% {background-color: green;}
      }
    </style>
  </head>
  <body>
    <p><b>Note:</b> This example does not work in Internet Explorer 9 and earlier versions.</p>
    <div></div>
  </body>
</html>
```

The following example will change both the background-color and the position of the element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 100px;
        height: 100px;
        background-color: red;
        position: relative;
        animation-name: example;
        animation-duration: 4s;
      }
      @keyframes example {
        0%   {background-color:red; left:0px; top:0px;}
        25%  {background-color:yellow; left:200px; top:0px;}
        50%  {background-color:blue; left:200px; top:200px;}
        75%  {background-color:green; left:0px; top:200px;}
        100% {background-color:red; left:0px; top:0px;}
      }
    </style>
  </head>
  <body>
    <p><b>Note:</b> This example does not work in Internet Explorer 9 and
earlier versions.</p>
    <div></div>
  </body>
</html>
```

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:

```
<!DOCTYPE html>
<html>
```

```
<head>
  <style>
    div {
      width: 100px;
      height: 100px;
      background-color: red;
      position: relative;
      animation-name: example;
      animation-duration: 4s;
      animation-delay: 2s;
    }
    @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>
```

Negative values are also allowed. If using negative values, the animation will start as if it had already been playing for *N* seconds.

In the following example, the animation will start as if it had already been playing for 2 seconds:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div {
      width: 100px;
      height: 100px;
      background-color: red;
```

```
        position: relative;
        animation-name: example;
        animation-duration: 4s;
        animation-delay: -2s;
      }
      @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>Using negative values: Here, the animation will start as if it had
already been playing for 2 seconds:</p>
    <div></div>
    <p><b>Note:</b> This example does not work in Internet Explorer 9 and
earlier versions.</p>
  </body>
</html>
```

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:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 100px;
        height: 100px;
        background-color: red;
        position: relative;
        animation-name: example;
```

```
        animation-duration: 4s;
        animation-iteration-count: 3;
      }
      @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>
```

The following example uses the value "infinite" to make the animation continue forever:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div {
        width: 100px;
        height: 100px;
        background-color: red;
        position: relative;
        animation-name: example;
        animation-duration: 4s;
        animation-iteration-count: infinite;
      }
      @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>
```

HTML5 & CSS Web Forms

Styling Input Fields

Use the **width** property to determine the width of the input field:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      input {
        width: 100%;
      }
    </style>
  </head>
  <body>
    <p>A full-width input field:</p>
    <form>
      <label for="fname">First Name</label>
      <input type="text" id="fname" name="fname">
    </form>
  </body>
</html>
```

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

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:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      input[type=text] {
        width: 100%;
        padding: 12px 20px;
        margin: 8px 0;
        box-sizing: border-box;
      }
    </style>
  </head>
  <body>
    <p>Padded text fields:</p>
    <form>
      <label for="fname">First Name</label>
      <input type="text" id="fname" name="fname">
      <label for="lname">Last Name</label>
      <input type="text" id="lname" name="lname">
    </form>
  </body>
</html>
```

Bordered Inputs

Use the **border** property to change the border size and color, and use the **border-radius** property to add rounded corners:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      input[type=text] {
        width: 100%;
        padding: 12px 20px;
```

```
        margin: 8px 0;
        box-sizing: border-box;
        border: 2px solid red;
        border-radius: 4px;
      }
    </style>
  </head>
  <body>
    <p>Text fields with borders:</p>
    <form>
      <label for="fname">First Name</label>
      <input type="text" id="fname" name="fname">
      <label for="lname">Last Name</label>
      <input type="text" id="lname" name="lname">
    </form>
  </body>
</html>
```

If you only want a bottom border, use the **border-bottom** property:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      input[type=text] {
        width: 100%;
        padding: 12px 20px;
        margin: 8px 0;
        box-sizing: border-box;
        border: none;
        border-bottom: 2px solid red;
      }
    </style>
  </head>
  <body>
    <p>Text fields with only a bottom border:</p>
    <form>
      <label for="fname">First Name</label>
      <input type="text" id="fname" name="fname">
      <label for="lname">Last Name</label>
      <input type="text" id="lname" name="lname">
```

```
    </form>
  </body>
</html>
```

Colored Inputs

Use the background-color property to add a background color to the input, and the color property to change the text color:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      input[type=text] {
        width: 100%;
        padding: 12px 20px;
        margin: 8px 0;
        box-sizing: border-box;
        border: none;
        background-color: #3CBC8D;
        color: white;
      }
    </style>
  </head>
  <body>
    <p>Colored text fields:</p>
    <form>
      <label for="fname">First Name</label>
      <input type="text" id="fname" name="fname" value="John">
      <label for="lname">Last Name</label>
      <input type="text" id="lname" name="lname" value="Doe">
    </form>
  </body>
</html>
```

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:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      input[type=text] {
        width: 100%;
        padding: 12px 20px;
        margin: 8px 0;
        box-sizing: border-box;
        border: 1px solid #555;
        outline: none;
      }
      input[type=text]:focus {
        background-color: lightblue;
      }
    </style>
  </head>
  <body>
  <p>In this example, we use the :focus selector to add a background color
 to the text field when it gets focused (clicked on):</p>
  <form>
    <label for="fname">First Name</label>
    <input type="text" id="fname" name="fname" value="John">
    <label for="lname">Last Name</label>
    <input type="text" id="lname" name="lname" value="Doe">
  </form>
  </body>
</html>
<!DOCTYPE html>
<html>
  <head>
    <style>
      input[type=text] {
        width: 100%;
        padding: 12px 20px;
        margin: 8px 0;
        box-sizing: border-box;
        border: 3px solid #ccc;
        -webkit-transition: 0.5s;
```

```
        transition: 0.5s;
        outline: none;
      }
      input[type=text]:focus {
        border: 3px solid #555;
      }
    </style>
  </head>
  <body>
    <p>In this example, we use the :focus selector to add a black border c
olor to the text field when it gets focused (clicked on):</p>
    <p>Note that we have added the CSS transition property to animate the
border color (takes 0.5 seconds to change the color on focus).</p>
    <form>
      <label for="fname">First Name</label>
      <input type="text" id="fname" name="fname" value="John">
      <label for="lname">Last Name</label>
      <input type="text" id="lname" name="lname" value="Doe">
    </form>
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      input[type=text] {
        width: 100%;
        padding: 12px 20px;
        margin: 8px 0;
        box-sizing: border-box;
        border: 3px solid #ccc;
        -webkit-transition: 0.5s;
        transition: 0.5s;
        outline: none;
      }
      input[type=text]:focus {
        border: 3px solid #555;
```

```
      }
    </style>
  </head>
  <body>
    <p>In this example, we use the :focus selector to add a black border c
olor to the text field when it gets focused (clicked on):</p>
    <p>Note that we have added the CSS transition property to animate the
border color (takes 0.5 seconds to change the color on focus).</p>
    <form>
      <label for="fname">First Name</label>
      <input type="text" id="fname" name="fname" value="John">
      <label for="lname">Last Name</label>
      <input type="text" id="lname" name="lname" value="Doe">
    </form>
  </body>
</html>
```

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:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      input[type=text] {
        width: 100%;
        box-sizing: border-box;
        border: 2px solid #ccc;
        border-radius: 4px;
        font-size: 16px;
        background-color: white;
        background-image: url('searchicon.png');
        background-position: 10px 10px;
        background-repeat: no-repeat;
        padding: 12px 20px 12px 40px;
      }
    </style>
```

```
  </head>
  <body>
    <p>Input with icon:</p>
    <form>
      <input type="text" name="search" placeholder="Search..">
    </form>
  </body>
</html>
```

Animated Search Input

In this example we use the CSS **transition** property to animate the width of the search input when it gets focus.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      input[type=text] {
        width: 130px;
        box-sizing: border-box;
        border: 2px solid #ccc;
        border-radius: 4px;
        font-size: 16px;
        background-color: white;
        background-image: url('searchicon.png');
        background-position: 10px 10px;
        background-repeat: no-repeat;
        padding: 12px 20px 12px 40px;
        transition: width 0.4s ease-in-out;
      }
      input[type=text]:focus {
        width: 100%;
      }
    </style>
  </head>
  <body>
    <p>Animated search input:</p>
    <form>
      <input type="text" name="search" placeholder="Search..">
    </form>
```

```
  </body>
</html>
```

Styling Textareas

Tip: Use the **resize** property to prevent textareas from being resized (disable the "grabber" in the bottom right corner):

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      textarea {
        width: 100%;
        height: 150px;
        padding: 12px 20px;
        box-sizing: border-box;
        border: 2px solid #ccc;
        border-radius: 4px;
        background-color: #f8f8f8;
        font-size: 16px;
        resize: none;
      }
    </style>
  </head>
  <body>
    <p><strong>Tip:</strong> Use the resize property to prevent textareas from being resized (disable the "grabber" in the bottom right corner):</p>
    <form>
      <textarea>Some text...</textarea>
    </form>
  </body>
</html>
```

Styling Select Menus

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      select {
        width: 100%;
```

```
        padding: 16px 20px;
        border: none;
        border-radius: 4px;
        background-color: #f1f1f1;
      }
    </style>
  </head>
  <body>
    <p>A styled select menu.</p>
    <form>
      <select id="country" name="country">
      <option value="au">Australia</option>
      <option value="ca">Canada</option>
      <option value="usa">USA</option>
      </select>
    </form>
  </body>
</html>
```

Styling Input Buttons

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      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;
      }
    </style>
  </head>
  <body>
    <p>Styled input buttons.</p>
    <input type="button" value="Button">
    <input type="reset" value="Reset">
```

```
    <input type="submit" value="Submit">
  </body>
</html>
```

Responsive Form

Resize the browser window to see the effect. When the screen is less than 600px wide, make the two columns stack on top of each other instead of next to each other.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      * {
        box-sizing: border-box;
      }
      input[type=text], select, textarea {
        width: 100%;
        padding: 12px;
        border: 1px solid #ccc;
        border-radius: 4px;
        resize: vertical;
      }
      label {
        padding: 12px 12px 12px 0;
        display: inline-block;
      }
      input[type=submit] {
        background-color: #4CAF50;
        color: white;
        padding: 12px 20px;
        border: none;
        border-radius: 4px;
        cursor: pointer;
        float: right;
      }
      input[type=submit]:hover {
        background-color: #45a049;
      }
      .container {
        border-radius: 5px;
```

```
        background-color: #f2f2f2;
        padding: 20px;
      }
      .col-25 {
        float: left;
        width: 25%;
        margin-top: 6px;
      }
      .col-75 {
        float: left;
        width: 75%;
        margin-top: 6px;
      }
      /* Clear floats after the columns */
      .row:after {
        content: "";
        display: table;
        clear: both;
      }
      /* Responsive layout -
 when the screen is less than 600px wide, make the two columns stack on to
p of each other instead of next to each other */
      @media screen and (max-width: 600px) {
        .col-25, .col-75, input[type=submit] {
          width: 100%;
          margin-top: 0;
        }
      }
    </style>
  </head>
  <body>

    <h2>Responsive Form</h2>
    <p>Resize the browser window to see the effect. When the screen is les
s than 600px wide, make the two columns stack on top of each other instead
 of next to each other.</p>
    <div class="container">
      <form action="/action_page.php">
      <div class="row">
```

```
        <div class="col-25">
          <label for="fname">First Name</label>
        </div>
        <div class="col-75">
          <input type="text" id="fname" name="firstname" placeholder="Your
 name..">
        </div>
      </div>
      <div class="row">
        <div class="col-25">
          <label for="lname">Last Name</label>
        </div>
        <div class="col-75">
          <input type="text" id="lname" name="lastname" placeholder="Your
last name..">
        </div>
      </div>
      <div class="row">
        <div class="col-25">
          <label for="country">Country</label>
        </div>
        <div class="col-75">
          <select id="country" name="country">
            <option value="australia">Australia</option>
            <option value="canada">Canada</option>
            <option value="usa">USA</option>
          </select>
        </div>
      </div>
      <div class="row">
        <div class="col-25">
          <label for="subject">Subject</label>
        </div>
        <div class="col-75">
          <textarea id="subject" name="subject" placeholder="Write somethi
ng.." style="height:200px"></textarea>
        </div>
      </div>
      <div class="row">
```

```
        <input type="submit" value="Submit">
      </div>
      </form>
    </div>
  </body>
</html>
```