CSS PROGRAMMING COOKBOOK

HOT RECIPES FOR CSS DEVELOPMENT

FABIO CIMO

WEB CODE GEEKS
WEB DEVELOPERS RESOURCE CENTER

CSS Programming Cookbook

Contents

Preface

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. Although most often used to change the style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content, such as semantically insignificant tables that were widely used to format pages before consistent CSS rendering was available in all major browsers. CSS makes it possible to separate presentation instructions from the HTML content in a separate file or style section of the HTML file. (Source: https://en.wikipedia.org/wiki/Cascading_Style_Sheets)

In this ebook, we provide a compilation of CSS based examples that will help you kick-start your own web projects. We cover a wide range of topics, from text styling and table design, to class inheritance and hover effects. With our straightforward tutorials, you will be able to get your own projects up and running in minimum time.

About the Author

Fabio is a passionate student in web tehnologies including front-end (HTML/CSS) and web design. He likes exploring as much as possible about the world wide web and how it can be more productive for us all. Currently he studies Computer Engineering, at the same time he works as a freelancer on both web programming and graphic design.

Chapter 1

CSS Inheritance

In this example, we'll focus on a css property value that I guess many of you ignore (do not use) while styling elements on websites.

It is `inherit`. Before we go straight into that, know the following:

Each element in HTML is part of a *tree*, and every element (except the initial `html` element) has a parent element that encloses it.

Whatever styles applied to the parent element can be applied to the elements inside it if the properties are inherited.

Below, we're going to have a look at several cases of inheritance property.

1.1 Prerequisites

Create a `html` document with the basic syntax inside like this:

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS Inheritance Property Value</title>
</head>
<body>

        <!-- STYLE SECTION -->
        <style type="text/css">

        </style>

        <!-- HTML SECTION -->

</body>
</html>
```

In the html section, we'll add some elements to show their relation to other elements and understand when they automatically inherit property values from their parents and when not.

Lets first see a basic application of the `inherit` property value.

1.2 Basic Application of Inherit

The inherit property value is at some cases automatically applied to child elements, and at other cases needs to be applied manually.

1.2.1 Automatic Inheritance

I always tend to give the `body` element a custom font-family so that all elements have the same font applied.

In other words, all elements `inherit` the font-family property from the body parent.

Look at the code below:

```
<!-- HTML SECTION -->

 This is a <span>parent</span> element.

<div class="child">This is a <span>child</span> element.

This is a <a href="#">link</a> inside a paragraph

<!-- end child element -->
</div><!-- end parent element -->
```

I have added a parent division and inside that two children, another div and a paragraph.

The classes given are so that we can give attributes to parent and see the results to children.

Now below look at the some initial properties I've given to these elements:

```
<!-- STYLE SECTION -->

<style type="text/css">

body {
        font-family: "Aller","sans-serif";
}

.parent {
        width: 15em;
        height: 5em;
        color: gray;
        font-weight: bold;
        font-size: 2em;
        border: 0.1em solid red;
}

.parent span {
        color: green;
}

</style>
```

Now to see what what properties have been inherited look at the browser view:

Figure 1.1: Initial Test on Inheritance

Seen this result, we can state that the following elements get inherited properties automatically:

1. All elements `inherit` the `font-family` property from a higher level parent, which is body. Notice all three lines have the same font applied.
2. All child elements `inherit` the `color` (font-color) from the parent element. Notice all the lines have the same gray color applied (except span and link).
3. The parent `span` element is set to have a green color, but also the child span gets a green font-color. So the child inherits from parent automatically span elements too.

Think everything is inherited?

The `a` anchor element has not been styled, but it shows in blue and underlined. That's because these two properties are set to be default ones for all anchor tags over the page.

The link did not inherit the color from its parent element, because it has a different color (blue). Look at the border. It has only been applied to the parent element, and it shows only there. That's why we sometimes need to apply inheritance manually to make these element fit the desired view.

1.2.2 Forced/Applied Inheritance

Simple as that, refer to the `.child` class and inherit the `border` for the child and `color` property from parent for the link:

```
.parent a {                              /* you can refer even to the child class */
        color: inherit;          /* inherited color from the parent class */
}

.child {
        border: inherit;         /* inherited the border attributes to child and elements  ↩
            inside */
}
```

The browser view:

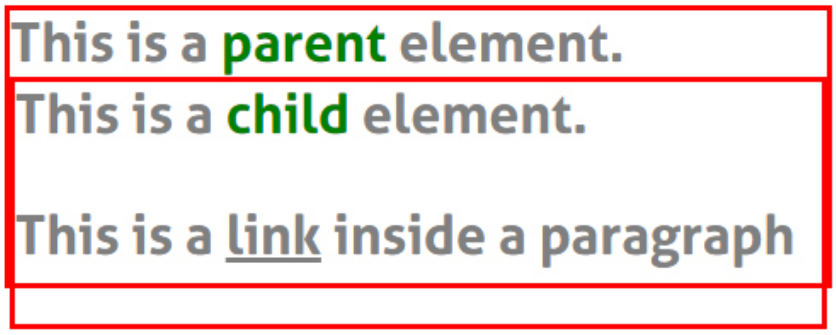

Figure 1.2: Inherit Value Applied to Color and Border

Notice that:

1. The link became the **same color as the parent** (its parent can be considered both the parent class element and the child class element, becuase the paragraph is nested inside the child element).
2. The border attributes got applied to the child element which contains another element inside, that's why you see the second smaller red border wrapping both the second and third line (child and paragraph elements).

That was a basic application of the inherit property value. Now let's see other cases and examples.

1.3 Cases and Examples

Here we look at some essential cases and examples where we can use the `inherit` property value.

1.3.1 Size and Background Inheritance

Change the html code to accommodate the following image and text elements:

```
<h4>Everything you can imagine is real.<br>-The Parent Image</h4>

<div class="child-image">
<h4>Glad to have looked after you.<br>-The Child Image</h4>

<!-- end child image -->
</div><!-- end parent image -->
```

and the css code accordingly:

```
.parent-image  {
        width: 30em;
        height: 30em;
        background-image: url("images/img1.jpg");
        background-repeat: no-repeat;
}

.parent-image h4 {
        color: gray;
        text-align: center;
        line-height: 1.5em;             /* vertically align text in the center */
        padding: 1em;                   /* just to differentiate the two elements */
```

```
}

.child-image {
        width: inherit;                                 /* inherited width */
        height: inherit;                                /* inherited height */
        background-image: inherit;              /* inherited background */
        background-repeat: inherit;
}
```

Now notice the child element inherits the background and size attributes from the parent:

Figure 1.3: Size and Background Values Inherited

1.3.2 Inheritance Manipulations

You can also use the inherit property value to avoid inheritance from one level higher parent.

Look at the codes below:

HTML

```
<!-- HTML SECTION -->



<div class="child1">
<h3>A title here</h3>
A paragraph text here
<button>A button here</button>
<!-- end child1 element -->


<h3>A title here</h3>
A paragraph text here
<button>A button here</button>
<!-- end child2 element -->

</div><!-- end parent element -->
```

CSS:

```
.parent h3 {
        font-variant: small-caps;
        color: red;
}
```

```
.parent p {
        text-indent: 1em;
        color: blue;
}

.parent button {
        padding: 1em 2em;
        background-color: gray;
        color: white;
        border-radius: 0.5em;
        border: 0.1em solid gray;
}

.child2 p {                                          /* declined inherited attributes  ↩
    from parent */
        text-indent: inherit;
        color: inherit;
}

.child2 h3 {                                    /* declined inherited attributes from  ↩
    parent */
        font-variant: inherit;
        color: inherit;
}

.child2 button {                                /* declined inherited attributes from  ↩
    parent */
        background-color: inherit;
        color: inherit;
}
```

Lets look at the browser view and then comment it:

Figure 1.4: Vice-Versa Inheritance

Notice that the parent attributes are applied to the first child but not to the second one.

That is because we chose to inherit attributes when they were already automatically inherited.

This case would result in an inheritance of the default attributes of a higher level parent (i.e body).

1.3.3 Tag and Location Inheritance

- **Tag Inheritance**

To explain this, take for example the `button` tag of html:

```
<button>Button</button>
```

With no styling at all, this element has already a view (attributes):

Figure 1.5: Tag Inheritance - Button Example

You can see it has a `border`, a gradient `background`, a `hover` state ect.

This is called tag inheritance, it means the element inherits default properties/attributes pre set by the browser.

Whenever you style a button, you just override the browser default properties for that element.

- **Location Inheritance**

Basically, location inheritance means using the same attributes for a set of elements under the same tag/class.

For example, if you want to have all titles styled with a green color and bold you could just refer to the following:

```
.title {
        color: green;
        font-weight: bold;
        font-size: 2em;
}
```

And apply this class to all elements wrapping a title like so:

```
2. Why us?

Just a paragraph to show that this is not a title.
<ul>
        <li>First</li>
        <li>Second</li>
        <li>Third</li>
</ul>

3. Help Center
```

Notice that elements given the `title` class now have a title look and feel.

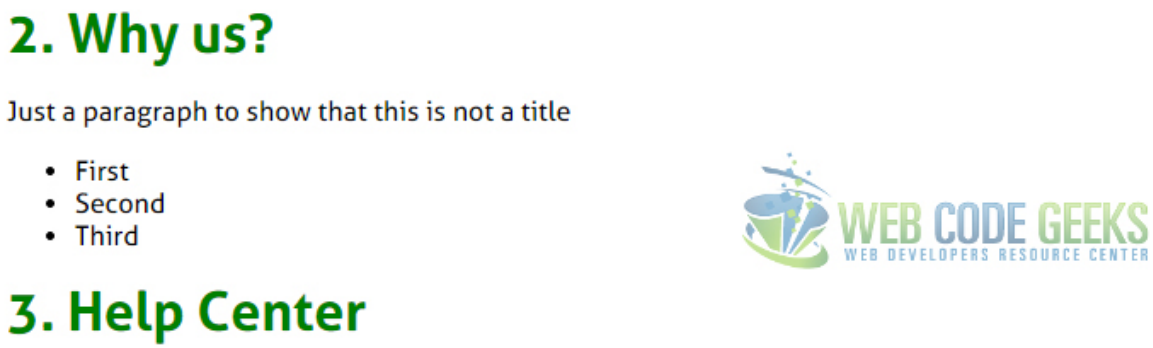

Figure 1.6: Location Inheritance - Title Example

1.3.4 Class Inheritance

Class inheritance is more and more being used in nowadays websites.

It means to apply certain styles from a predefined class, and other styles from another predefined class.

Take, for example, the multiple classes application in css, like below:

HTML

```
Just a paragraph here. It could be lorem ipsum.
<ul class="text-style-1">
        <li>This will be just a text line</li>
        <li class="link bold">Download Now</li>
        <li>Third line goes here</li>
</ul>
```

CSS

```
.text-style-1 {
        color: #50838e;
        font-weight: italic;
        font-size: 1.2em;
}

.link {
        color: blue;
        text-decoration: underline;
}

.bold {
        font-weight: bolder;
}
```

So above I have declared three classes and given them attributes.

Then, I have used them to style the elements by giving them these classes.

Figure 1.7: Class Inheritance - Text Example

Now all three lines will have the attributes of `text-style-1` but in addition to that, the second `li` is going to have added the attributes of the `link` class and the `bold` class.

We can say that the second line **inherits** all ul attributes and just keeps adding new (or overriding existing) attributes.

1.4 Conclusion

On a more professional approach, we can state that using inheritance is not of much usage as most elements will need specific styling and independence from parent elements they are in.

However, when using inheritance property value to give elements styling attributes, keep in mind that `inherit` can be applied just as a single value (i.e you can't have something like: border: inherit 1em #ccc;) so you don't get individual values inherited.

If you want to see the browser default styles (from which your element attributes are inherited), you can inspect element and check "Show Inherited Properties" in Chrome.

1.5 Download

Download You can download the full source code of this example here: **CSS Inheritance Example**

Chapter 2

CSS Multiple Classes

In this example we are going to have a look at multiple classes that you can apply on an element of html.

It has become more and more useful declaring some standard classes in CSS (actually referring to them, even though they do not exist) and using these classes whenever we need in our elements.

It basically is the inverse process of giving an element a class on html and then referring to that element in css to give attributes. In this case, you predefine some classes (that is, give them attributes) and then include them in your elements.

2.1 Predefine Classes & Attributes in CSS

In this step, let's create different kind of classes and give attributes to them to set up a group of classes that we can later use in html. So, first **create a basic html file** with a `style` tag in it where we can add the css, like below:

```
<!DOCTYPE html>
<html>
<head>
        <title>Multiple Classes on Elements</title>
</head>
<body>

        <!-- STYLE SECTION -->

        <style type="text/css">

        </style>

        <!-- HTML SECTION -->

</body>
</html>
```

2.1.1 Alignment Classes

Creating alignment classes will enable us to easy align elements on the web page. Let's name two alignment classes:

`.pull-left` - this will align an element to the left `.pull-right` - this will align an element to the right

The respective CSS code for these classes would be:

```
<style type="text/css">

  body{
```

```
   font-family: "Arial", "sans-serif";  }      /* just added custom font */

  /* ALIGNMENT CLASSES */

  .pull-left {                    /* pull-left class referred and given attributes */
    float: left !important;
  }

  .pull-right {                   /* pull-right class referred and given attributes */
    float: right !important;
  }

</style>
```

2.1.2 Color Classes

Creating color classes will let us color elements/backgrounds on the web page easier.

Let's name four color classes and three background color classes:

`.red` - this will give a text element or the border of a shape the red color. `.green` - this will give a text element or the border of a shape the green color. `.blue` - this will give a text element or the border of a shape the blue color. `.white` - this will give a text element or the border of a shape the white color.

`.bg-red` - this will give an element a red background color. `.bg-green` - this will give an element a green background color. `.bg-blue` - this will give an element a blue background color.

The respective CSS code for these classes would be:

```
  /* COLOR CLASSES */

  .red, .red li, .red a {               /* red class referred and given color attribute */
    color: #e85139;
  }

  .green, .green li, .green a {         /* green class referred and given color attribute  ↩
      */
    color: #4ca640;
  }

  .blue, .blue li, .blue a {            /* green class referred and given color attribute  ↩
      */
    color: #319cd6;
  }

  .white, .white li, .white a {           /* green class referred and given color attribute ↩
       */
    color: white;
  }

  .bg-red     {                  /* bg-red class referred and given backgrounf color attr  ↩
      */
      background-color: #e85139;
    }

  .bg-green  {                  /* bg-green class referred and given background color attr  ↩
      */
      background-color: #4ca640;
    }

  .bg-blue   {                  /* bg-blue class referred and given background color attr  ↩
      */
```

```
    background-color: #319cd6;
  }
```

Note that we also added the color attributes for `li` or `a` tags on the html.

That is so we can access li's or a's directly with these classes.

2.1.3 Size Classes

Creating size classes will let you control the elements sizes on the web page. Let's name three size classes:

`.small` - this will make an element look small on the screen. `.normal` - this will make an element look medium on the screen. `.large` - this will make an element look large on the screen.

The respective CSS code for these classes would be:

```
/* SIZE CLASSES */

.small {                    /* small class referred and given sizing attributes */
   font-size: 1em;
   padding: 0.2em;
   width: 50%;
   height: 50%;
}

.normal {                   /* normal class referred and given sizing attributes */
   font-size: 1.5em;
   padding: 0.4em;
   width: 100%;
   height: 100%;
}

.large {                    /* large class referred and given sizing attributes */
   font-size: 2em;
   padding: 0.6em;
   width: 150%;
   height: 150%;
}
```

2.1.4 Text Classes

Creating text classes will make it easier to control text appearance. Let's name four text classes:

`.decor1` - this will make the text `underline` and `uppercase`. `.decor2` - this will make the text `overline` and `lowercase`. `.decor3` - this will make the text `line-through` and `capitallize`. `.center` - this will make the text align center.

The respective CSS code for these classes would be:

```
/* TEXT CLASSES */

.decor1 {                       /* decor1 class referred and given text attributes */
 text-decoration: underline;
 text-transform: uppercase;
 font-weight: bold;
}

.decor2 {                       /* decor2 class referred and given text attributes */
 text-decoration: overline;
 text-transform: lowercase;
 font-weight: bold;
```

```
}

.decor3 {                          /* decor3 class referred and given text attributes */
 text-decoration: line-through;
 text-transform: capitalize;
 font-weight: bold;
}

 .center {                        /* center class referred and given attributes */
 position: relative;
 bottom: 0px;
 text-align: center;
}
```

2.1.5 Elements Classes

It is time to create some specific elements classes, which will represent the initial attributes of the elements we are going to give several classes. Here is what we will create:

`.button` - a very popular element which will be used to demonstrate. `.menu` - a four links menu with very little prestyled attributes. `.rectangular` - a prestyled fixed width and height element as example.

The respective CSS code for these classes would be:

```
/* ELEMENTS CLASSES */

.button {                       /* refers to a button class, given several attributes */
 border: 0.1em solid;
 border-radius: 0.3em;
 width: 5em;
 height: 2em;
 line-height: 2em;
 margin-top: 5em;
 margin-bottom: 2em;
}

.menu li {                     /* refers to a menu class, given several attributes */
 display: inline;
 padding-right: 1em;
 text-decoration: none;
}

.menu {
 margin-bottom: 2em;           /* just a margin for better view of the elements */
}

.rectangular {                 /* refers to a rectangular class, given several attributes ↩
    */
 border: 0.1em solid;
 width: 10em;
 height: 5em;
 line-height: 5em;
 margin-bottom: 2em;
}
```

2.2 Application of Multiple Classes in HTML

Here is where the magic happens, we will first create the basic elements structure in html

and then continue adding appropriate classes to each elements to see the results of what we;ve done so far!

2.2.1 Set up the HTML

Under your `tag` start the `html` code. Add four elements: a button, a menu, a rectangular and an image like so:

(Do not give these elements classes, we will do that next)

```
<!-- HTML SECTION -->

<!-- button element -->
<a href="#">Button</a>

<!-- menu element -->

<ul>
  <li class=""><a href="#">Home</a></li>
  <li class=""><a href="#">About</a></li>
  <li class=""><a href="#">Help</a></li>
  <li class=""><a href="#">Contact</a></li>
</ul>


<!-- rectangular element -->
I'm a rectangular

<!-- image element -->
<img src="images/image.png">
```

Because none of these elements is styled (or better say, not given a class) the view we would get is this:

Figure 2.1: Initial View of the Elements in HTML

Now let's give them the respective classes like this:

```
<!-- HTML SECTION -->

<!-- button element -->
<a href="#">Button</a>
```

```
<!-- menu element -->

<ul>
  <li class=""><a href="#">Home</a></li>
  <li class=""><a href="#">About</a></li>
  <li class=""><a href="#">Help</a></li>
  <li class=""><a href="#">Contact</a></li>
</ul>


<!-- rectangular element -->
I'm a rectangular

<!-- image element -->
<img src="images/image.png">
```

Note that the `img` element doesn't still have a class, because we did not create attributes especially for it.

Now lets see what we've got in the browser.

Figure 2.2: Basic classes applied on elements

We've created the most basic version of our final html.

2.2.2 Add multiple classes

Now lets go ahead and add classes that we've already given attributes to in css to make elements look better.

The way you apply multiple classes to an element is just by seperating them with a space inside the quotes like this:

`class="example class1 class2 class3 class4 class5"`

1. Adding Classes to the Button

I will add 4 more classes to our `button` element:

`center` - add this class to make the text align center. `bg-blue` - this class will make the button background blue. `white` - this will change the color of the text to white. `large` - add this class to make the whole element larger.

And this is how the code will look like:

```
<!-- button element -->
<a href="#">Button</a>
```

Now look at this nice button view we get:

Figure 2.3: Button Element - Classes Applied

2. Adding Classes to the Menu

Lets add 3 more classes, 2 of them to the `menu` and 1 to the `li`'s:

`red` - gives the text of this element a red color. `pull-right` - alignes the whole element on the right side of the screen.

`decor1` - makes the text in the list underlined, uppercase and bold.

This is how it should be in the code:

```
<!-- menu element -->

<ul>
  <li class="decor1"><a href="#">Home</a></li>
```

```
    <li class="decor1"><a href="#">About</a></li>
    <li class="decor1"><a href="#">Help</a></li>
    <li class="decor1"><a href="#">Contact</a></li>
  </ul>
```

And the expected view in the browser would be like this:

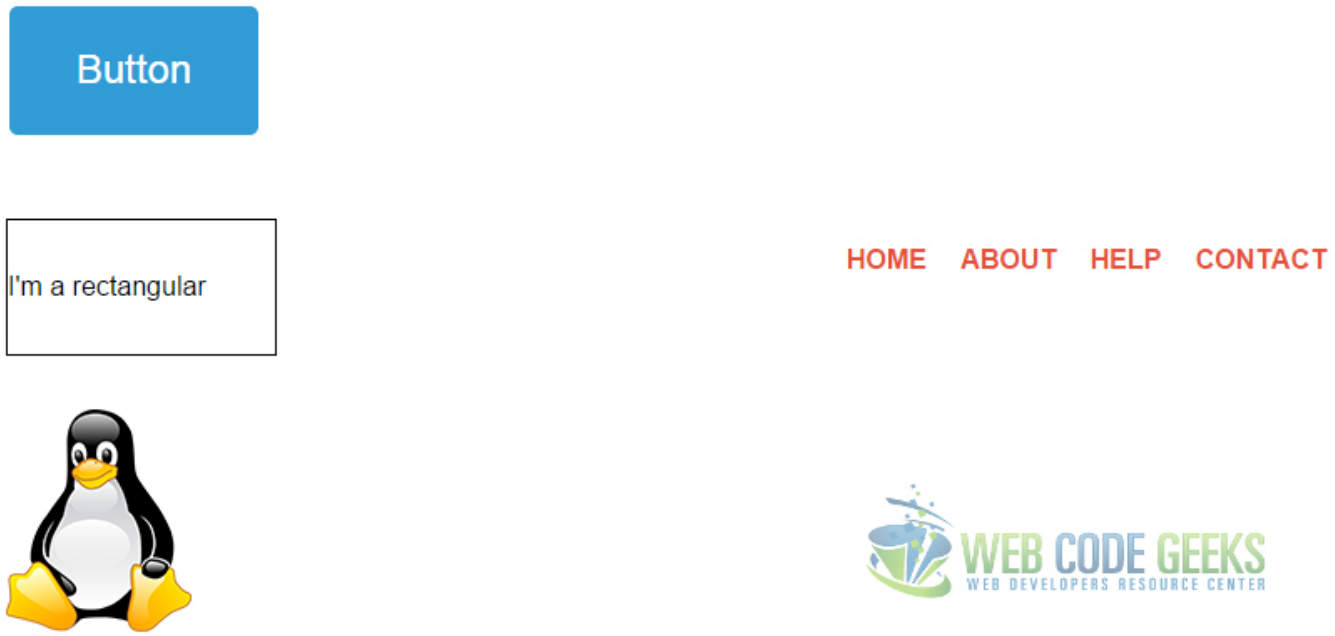

Figure 2.4: Menu Element - Classes Applied

Note that the `decor1` attributes also define underlined text, but you don't see it because of this code I added when dealing with the button element, which had the browser pre-styled any-link underlined attribute:

```
a:-webkit-any-link {
color: -webkit-link;
text-decoration: none;
}
```

It means do not underline any links by default, and that was needed for our button.

3. Adding Classes to the Rectangular

For the rectangular element, lets add 5 more classes after the `rectangular` basic class:

`center` - this will center the text inside the rectangular. `bg-green` - this will give it a green background color. `white` - this will make the text color white `decor3` - this decor will make the text `line-through`, `capitallize` and `bold`. `normal` - this will adjust the size of the rectangular to what we defined `normal`.

By now you should have learned how to add these classes, it will look like this in the code:

```
<!-- rectangular element -->
I'm a rectangular
```

Now notice the rectangular view:

Figure 2.5: Rectangular Element - Classes Applied

4. Adding Classes to the Image

There is not too much (from what we've done in css) that we can add to the image element, however lets see these:

`bg-red` - this will give the image (which is png) a red background color. `pull-right` - this will align the image on the right like the menu.

In the image element, first add a `class` tag and then add classes inside:

```
<!-- image element -->
<img class="bg-red pull-right" src="images/image.png">
<!-- added a class tag and then classes -->
```

And our final element would look this way:

Figure 2.6: Image Element - Classes Applied

You now have the concept of using multiple classes on elements that have been previously styled in css.

2.3 Some Considerations

The mupltiple classes application seems a great method for saving time and creating some standards for yourself, and it is, but there are also certain aspects and considerations you should keep in mind.

2.3.1 Do not Overwrite

Given the element specific classes for various types of look, alignment, color ect you want, do not write other/extra classes of the same property you have previously used. Depending on the case, it will sometimes apply the first/second class, or none of them at some cases. The code like below would be logically wrong:

```
<!-- rectangular element -->
```

And guess what, the rectangular is still the same in the browser view, nothing has changed.

So there is no reason to use multiple classes to apply the same property.

2.3.2 Avoid Cluttered Code

You might have noticed that we did not use all of the classes we referred to in html.

For example, we didn't use `pull-left` because by default all our elements were aligned on the left, we didn't also use the `small` class because it made elements we needed too small, and so on and so forth with come color classes. Knowing that, you are recommended to follow one of the practices below:

1. Do not use classes that you think will never be needed on the elements you are going to use.

2. Use any class you like or think will be needed, and clean up the unused classes when you finish your project.

Done that, you will have a more organised code suited to your needs.

2.3.3 Consider using Frameworks

You might get deep into this and create a set of classes (like, a lot of them) and this would be the approach of a css framework. If you have reached a level of professionalism that you feel you can do that, go ahead.

Otherwise, feel free to use popular and productive frameworks like Bootstrap (http://getbootstrap.com/) which have thousands of predefined classes you can use to get your project going.

2.4 Conclusion

To conclude, there is a lot you can do using multiple classes on elements, like easier and faster create forms, menus, buttons, tables, align, color elements ect, and there are advantages to using it in your projects, but always remember to give them names according to the main function they do, so that you won't get stuck in long code projects and have it easier to understand what has been used each class for, without having to search in the css code.

Play around with it to see how this works by yourself.

2.5 Download

Download You can download the full source code of this example here: **Multiple Classes Example**

Chapter 3

CSS Last Child

In this example we're going to have a look at the last-child css selector.

This selector of css is used to match every element that is the last child of its parent and give attributes and properties to specifically that element.

It is compatible with all modern browsers, except versions of IE under 9.

The usage will be necessary when taking into consideration giving attributes to a specific element in a group of elements of the same (or not) type.

Let's first see the basic application and then some cases and examples.

3.1 Basic Setup & Application

First, go ahead and create a blank html document and then add the basic syntax inside like so:

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS3 Image Rotate Example</title>
</head>
<body>
<!-- STYLE  SECTION -->
<style type="text/css">

</style>

<!-- HTML SECTION  -->

</body>
</html>
```

Now to apply the last-child selector, first add some elements like below:

```
<!-- HTML SECTION  -->

This is the first paragraph
This is the second paragraph
This is the third paragraph
This is the fourth paragraph
This is the fifth paragraph
```

Now the most basic application would be: `tag:last-child` or `class:last-child` like below:

```
<!-- STYLE  SECTION -->

<style type="text/css">

p:last-child{
        background-color: yellow;
}

</style>
```

In the browser, you can see the last line out of five having a yellow background color:

Figure 3.1: Last-Child Selector Basic Application

3.2 Cases and Examples

Another case would be when you change a text styling or boxes with text inside. Look at the html below:

```
<!-- HTML SECTION  -->

<div class="parent">
        <div>This will be a default size font
        Also this is going to be default text
        Random Text 3
</div>


        <div class="box">I am box 1
        I am box 2
        I am box 3

</div>
```

And applying the css for last-childs we'd get:

```
<!-- STYLE  SECTION -->

<style type="text/css">

.parent div:last-child{
        font-family: "Montserrat";
        font-size: 2em;
        width: 10em;
```

```
        height: 5em;
        color: red;
}

.box {
        width: 8em;
        height: 5em;
        border: 0.2em solid #ccc;
        margin-bottom: 1em;
}

.boxes .box:last-child {
        background-color: yellow;
        text-align: center;
        font-size: 2em;
        font-family: "Lato";
        padding-top: 3em;
}

</style>
```

In this case, the view of these *last children* would be:

Figure 3.2: Example of Last Child Application

Another usage of it can be noticed when also using the `nth-last-child(n)` like below:

```
<!-- HTML SECTION  -->

First line here
Second line here
Third line here
Fourth line here
Fifth line here
```

Applying the fourth child element means selecting the second, because the counting starts from the last element:

```
<!-- STYLE  SECTION -->

<style type="text/css">

p:nth-last-child(4){
        font-family: "Calibri";
        font-weight: bold;
```

```
        font-size: 2em;
}

</style>
```

As you'd expect, the second line has different attributes applied:

Figure 3.3: Nth-Last-Child Application

3.3 Conclusion

To conclude, the last-child css selector is good for targeting only the last element of a parent element.

However, if you want to select more than one element and counting from the bottom up, consider using nth-last-child.

It is up to you to decide if you need the first-child or last-child selector according to your needs.

3.4 Download

Download You can download the full source code of this example here: **CSS Last Child Selector**

Chapter 4

CSS Table Design

The aim of this example is to show how you can create tables with HTML and style them with CSS, including specific and various examples.

Since web pages are loaded with a lot of information, it sometimes becomes a neccesity organising data in a more easy-to-read and intuitive way.

That is where CSS can be of great help. Below you can find a complete guide to CSS Tables, from the html setup to the stylish viewpoint of table rows and cells.

4.1 The Initial Layout

First, go on and create a `html` file. This file is going to be the one we are putting both HTML and CSS code so as to keep it as simple as possible. However, when considering more complicated webpages with a lot of HTML, I recommend you separate the styling code in a `css` file.

4.1.1 Setting up the HTML

The initial layout consists of creating the basic structure of html we need.

To do this, let's first explain what kind of tags do we have for tables in html and what do they mean:

1. `table` - this tag defines a **table** in html.
2. `tr` - this tag divides the table into **table rows**.
3. `td` - this tags creates the cell, it means **table data**.
4. `th` - this tag creates a **table heading**.

With this said, let's create the basic table structure in html with randomly 3 rows and 3 columns, like below:

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS Tables Example</title>
</head>
<body>

</body>
</html>
```

As you can see, we created the `table` tag inside the `body` tag. After this, we added first the rows tag `tr` and inside each row the table data (or cells) with the `td` tag. Notice we also added a table heading row using the `th` tag, in the first row.

By default, you should not see anything shown in the browser until now, and that is because we only created the table, but did not put there any data or styling, so both information and layout are missing. We'll add them soon.

4.1.2 Understanding the Style Basics

Our table needs to be filled, so let's give it some random information in this example. To give it styling and layout attributes we create a `style` tag where all the css code is going to be put. To begin, I am giving this `table` two attributes:

1. `width` attribute of 40% (a normal width for having a pleasant view of the table, we will add custom width later)
2. `border` attribute of `0.1em solid` to the `table`, `th` and `td` tags.

The code now looks like this:

```
<style type="text/css">

        table {
        width: 40%;
        border: 0.1em solid;    }

        td {
        border: 0.1em solid;    }

        th {
        border: 0.1em solid;    }

</style>
```

```
<th>Name</th>
    <th>Surname</th>
    <th>Age</th>
```

And this is what the table in the browser looks like:

Name	Surname	Age
Fabio	Cimo	20
Jasmin	Brown	18

Figure 4.1: First Look of the Table View with inital attributes applied.

Notice that the table headings are set in a **bold** style and **centered** position by default by the browser.

So remember that whatever you put inside the `th` tag is styled this way everytime, but you can change it.

Other table data, like the cell information is just normal text and aligned left by default.

You probaby see that every cell has its own surrounding/separate border and that looks just bad.

We fix that with the `border-collapse:collapse` attribute given to the `table`.

(We are later in this article going to have a closer look to the `border-collapse` property)

That eleminates the need for `border` attributes inside the `table` tag.

Lets also align all the table data (`td` tag) in the `center`.

Now we have a code that looks like this in the style section:

```
<style type="text/css">

table {
width: 40%;                          /* the 'border' attribute is removed here */
border-collapse:collapse }

td {
border: 0.1em solid;
text-align: center;     }

th {
border: 0.1em solid;
text-align: center;     }

</style>
```

The result in the browser would be:

Name	Surname	Age
Fabio	Cimo	20
Jasmin	Brown	18

Figure 4.2: Table View with *text-align* and *border-collapse* attributes applied.

Now we can say we have created a simple table that looks fine. But there are lots of other styles we can apply to make this table look a lot more interesting and attractive.

4.2 Modifying Table Elements Styles

Up until now, we used a `40% width` attribute for our table.

Now we will see a more specific way for adjusting the width of the table, defined by the width of every cell.

4.2.1 Spacing

There are several spacing considerations when talking about table/row/cell sizes. Two of them are most popular and shown below.

1. Set the size of of your table according to the desired `width` of the **cells** and `height` of the **rows** like this:

```
<style type="text/css">

table {
                                                                    /* the 'width: 40%' attribute is ↩
                                                                        removed here */
border-collapse:collapse }

td {
border: 0.1em solid;           /* cells have their 'border' attribute */
text-align: center;
width: 10em;     }             /* added cell width */

th {
border: 0.1em solid;          /* cells have their 'border' attribute */
text-align: center;
width: 10em;     }             /* added cell width */

tr {
height: 2em;     }           /* added row height */


</style>
```

This would make the **cells 10em wide** and **rows 2em high**. And we'd get this view:

Name	Surname	Age
Fabio	Cimo	20
Jasmin	Brown	18

Figure 4.3: Table Width and Spacing using Cells Width and Rows Height

1. Set an inner space called `padding` to the cells (and/or headings of the table) by adding this attribute to the style like this:

```
<style type="text/css">

table {
border: 0.1em solid;
border-collapse:collapse }

td {
border: 0.1em solid;
text-align: center;
width: 10em;
```

```
padding: 1em;    }                                  /* added cell padding to the table data  ↩
    cells */

th {
border: 0.1em solid;
text-align: center;
width: 10em;
padding: 2em;    }                                  /* added cell padding to the heading cells  ↩
    */

tr {
height: 2em;     }


</style>
```

In this case, I added a **2em** `padding` to the heading cells and **1em** to the data cells.

Name	Surname	Age
Fabio	Cimo	20
Jasmin	Brown	18

Figure 4.4: Cell Spacing using the Padding attribute

4.2.2 Border Styling

Until now, we have set a `1px solid` border style for our table. And that is just good.

But lets see other options and customizations we can apply to the table border.

`dotted` - dotted border `dashed` - dashed border `solid` - solid border `double` - double border

The way we apply border styles is as in the code below:

```
<style type="text/css">

table {
border: 0.1em dashed;              /* added different border style: dashed */
border-collapse:collapse }

td {
```

```
border: 0.1em dashed;                          /* added different border style: dashed */
text-align: center;
width: 5em;
padding: 0.5em; }

th {
border: 0.1em dashed;                          /* added different border style: dashed */
text-align: center;
width: 5em;
padding: 1em;    }

tr {
height: 1em;     }


</style>
```

For the sake of displaying these four styles in one picture, I have reduced the width and padding of the cells.

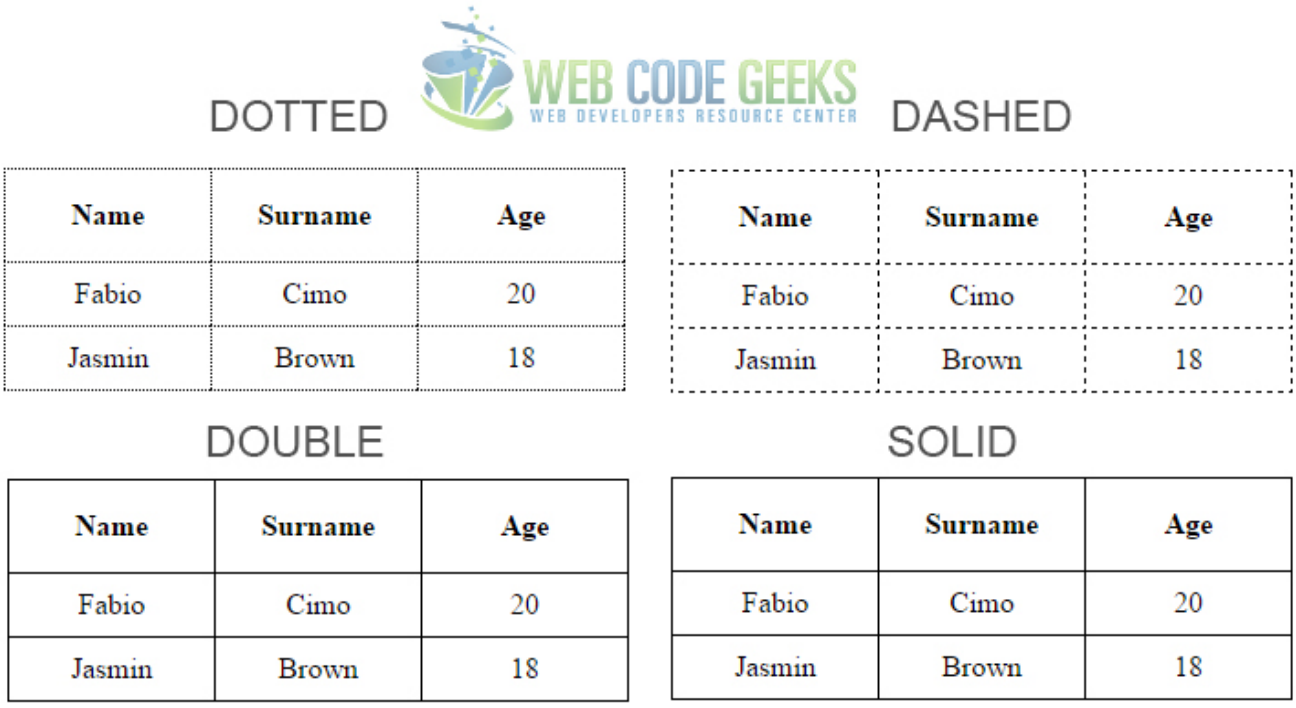

Figure 4.5: Fixed Border Styles

We can also have combinations of these styles, using the following:

`dotted solid` which gives a top and bottom style of `dotted` and left and right of `solid` `dotted solid double dashed` which gives a specific style to each border.

Additionally, borders can have different colors and border-widths than the default by applying:

1. For the color, the `border-color` attribute or simply adding the color code into the `border` attribute.
2. For `border-width` the attribute with the same name or right from the `border` tag.

All cases are shown below and commented:

```
<style type="text/css">

table {

border-collapse: collapse;       }

td {
border: solid #e74f3b;                          /* added border color inside the border  ↩
    attribute */
text-align: center;
width: 10em;
border-width: 0.2em;                            /* added border width as a separate  ↩
    attribute */
padding: 1em;                    }


th {
border: 0.2em solid;                            /* added border width inside the border  ↩
    attribute */
text-align: center;
width: 10em;
padding: 1em;
border-color: #329bd7;           }              /* added border color as a separate  ↩
    attribute */


tr {
height: 1em;     }

</style>
```

This is what the table looks like after these two modifications on the css code:

Name	Surname	Age
Fabio	Cimo	20
Jasmin	Brown	18

Figure 4.6: Border Width and Border Color Applied

4.2.3 Cell Styling

Adding a few more lines into our `css` code will give our cell backgrounds a new look. Let's remove some of the styling we did before like the border-width and border-color attributes to have a cleaner table.

In the code below, I am giving the **heading** cells a light gray background and **data** cells a light green background:

```
td {
border: 0.1em solid;                          /* resetted to a default of 0.1em width */
text-align: center;
width: 10em;
padding: 1em;
background-color: #d8ffc4;              }     /* added data cell background color (light  ↩
    green) */


th {
border: 0.1em solid;                          /* resetted to a default of 0.1em width */
text-align: center;
width: 10em;
padding: 1em;
background-color: #ededed;              }     /* added heading cell background color ( ↩
    light gray) */
```

Refreshing the browser, the result would be:

Name	Surname	Age
Fabio	Cimo	20
Jasmin	Brown	18

Figure 4.7: Table View after adding the *background-color* attribute to the cells

4.3 Advanced Layout & Style Customization

Even though we have covered a lot until now, there are still some interesting layout and styling changes that might be of great help to our specific cases. So we start with the layout changes we can use.

4.3.1 Layout Cases: Border-Collapse Property

`border-collapse` - this property specifies whether the browser should control the appearance of adjacent borders that touch each other or whether each cell should maintain its style. It can have two possible values:

`border-collapse:collapse` and `border-collapse:separate` The code below shows both cases: (duplicated our table code to have 2 tables in order to show both these two properties):

CSS

```
<style type="text/css">

table.one {
border-collapse: collapse;              /* added border collapse:collapse to the first  ↩
    table */
margin-bottom: 5em;     }

table.two {
border-collapse: separate;      }       /* added border collapse:separate to the second  ↩
    table */

caption {
padding-bottom: 1em;    }               /* added caption padding to make it distinctive */

td {
border: 0.1em solid;
text-align: center;
width: 10em;
padding: 1em;           }

th {
border: 0.1em solid;
text-align: center;
width: 10em;
padding: 1em;           }

tr {
height: 1em;    }

</style>
```

HTML

```
        <caption>Collapse Border Example</caption>  <!-- added table caption -->
        <th>Name</th>
    <th>Surname</th>
    <th>Age</th>

  |Fabio|Cimo|20
  |Jasmin|Brown|18

        <caption>Separate Border Example</caption>      <!-- added table caption -->
        <th>Name</th>
    <th>Surname</th>
    <th>Age</th>

  |Fabio|Cimo|20
  |Jasmin|Brown|18
```

In this example I also added table captions just by adding the `caption` tag under the `table` tag.

And referred to captions in css by giving the `caption` property attributes.

You can also have captions in the `top`, `bottom`, `left` or `right` of the table according to your needs.

You can achieve this by adding the `caption-side` attribute in your css code under the `caption` class.

The view we are going to get is this:

Collapse Border Example

Name	Surname	Age
Fabio	Cimo	20
Jasmin	Brown	18

Separate Border Example

Name	Surname	Age
Fabio	Cimo	20
Jasmin	Brown	18

Figure 4.8: Border-Collapse Property Cases

4.3.2 Layout Cases: Border-Spacing Property

`border-spacing` - specifies the distance that separates adjacent cells, borders. It can take one or two values according to the spacing you need on different sides.

The code below shows both cases:

```
table.one {
        margin-bottom: 5em;
        border-collapse:separate;
        border-spacing: 1em;  }                /* added the same top,bottom,right,left  ↩
            border spacing  */

table.two {
        border-collapse:separate;
        border-spacing: 0.5em 3em;      }      /* added different top,bottom and right, ↩
            left border spacing  */
```

In this case, we get the following view:

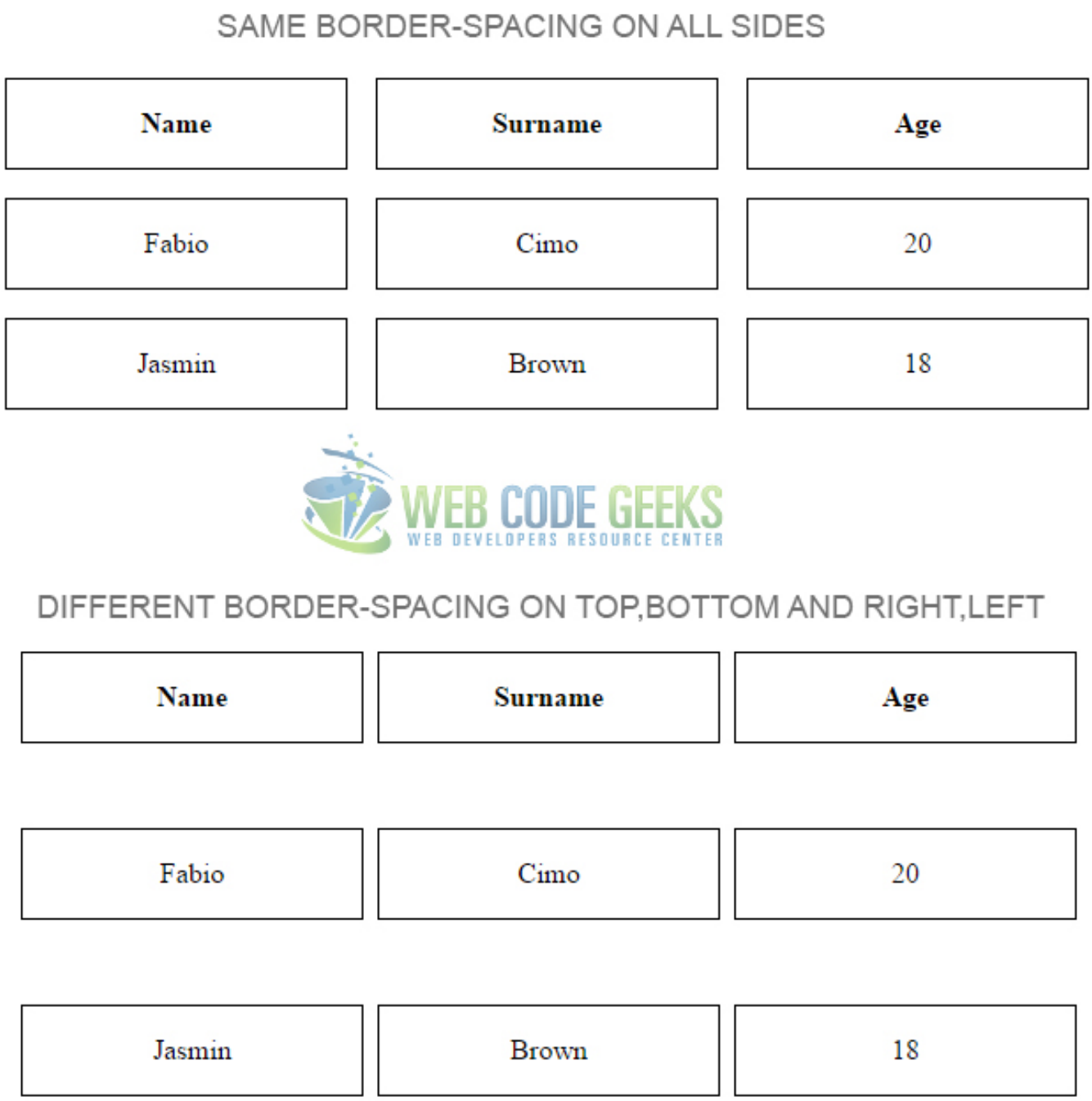

Figure 4.9: Border-Spacing Property

4.3.3 Layout Cases: Empty-Cells Property

`empty-cells` - used to indicate whether a cell without any content should have a border displayed. It can take 3 values:

`show` - given this attribute, all empty cells will show. `hide` - given this attribute, all empty cells will hide. `inherit` - given this attribute, all empty cells will by default show.

All three cases are shown in the code below with the respective comments:

```
table.one {
```

```
margin-bottom: 3em;
border-collapse:separate;
empty-cells: inherit;  }        /* added empty-cells property: inherit */

table.two {
margin-bottom: 3em;
border-collapse:separate;
empty-cells: show; }            /* added empty-cells property:show  */

table.three {
border-collapse:separate;
empty-cells: hide; }            /* added empty-cells property:hide  */
```

Note that the `empty-cells` property only works when the `border-collapse` attribute is set to `sepatare`.

Here are the 3 results we get:

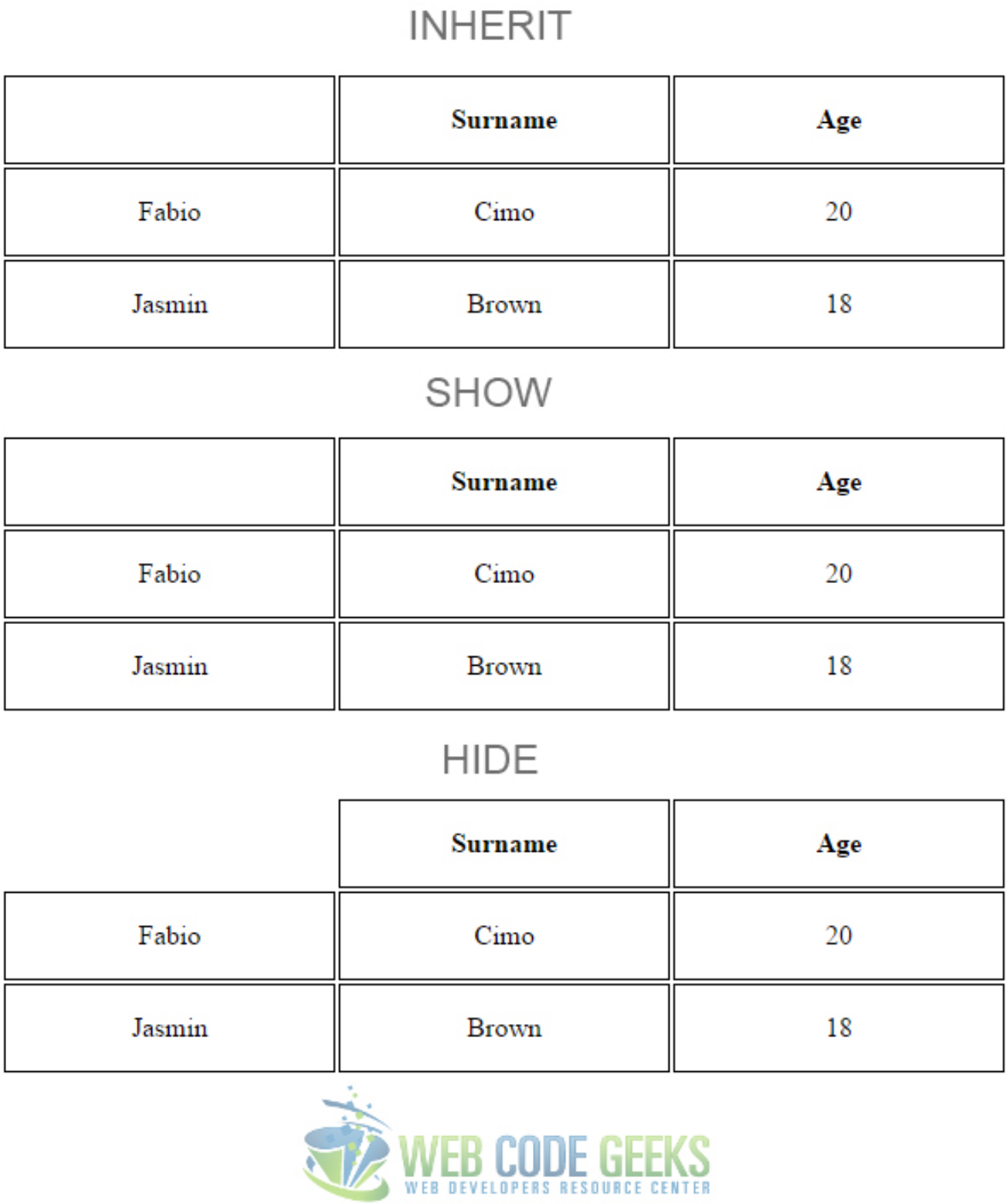

Figure 4.10: Empty-Cells Property

In this case, to demonstrate the empty-cells property, I removed the first table heading from the html as below:

```
<th></th>           <!-- removed text from here to have an empty cell -->
```

```
    <th>Surname</th>
    <th>Age</th>
```

4.3.4 The *colspan* Attribute

`colspan=""` - defines the number of columns a cell should span.

This attribute is given inside the `th` or `td` opening tag in html.

The number is put inside the quotes. It can be useful when you need different divisions of cells for different rows.

Application of the `colspan` attribute is as follows:

```
<th colspan="2"> FULL NAME</th>
    <!-- removed second th and added a colspan on the first -->
    <th>AGE</th>

<!-- other rows here -->
|Average Age<!-- removed second th and added a colspan on the first -->
    |22
```

In the browser, the table would looke like this:

FULL NAME		AGE
Fabio	Cimo	20
Jasmin	Brown	18
James	Wick	35
Donald	Dawn	29
Average Age		22

Figure 4.11: Colspan Attribute Applied

4.3.5 Table Design

Professional looking tables come with some simple and eye-catching design.

There is a pseudo-selector in css that we call went we want specific elements (the child) to be the same out of a larger element (the parent). This is the `nth-child()`. Inside the brackets we add a condition. Lets see the code below:

```
<style type="text/css">

table {
        font-family: "Lato","sans-serif";       }               /* added custom font-family ↩
             */

table.one {
        margin-bottom: 3em;
        border-collapse:collapse;       }

td {                                                    /* removed the border from the  ↩
    table data rows  */
        text-align: center;
        width: 10em;
        padding: 1em;           }

th {                                                      /* removed the border from the  ↩
    table heading row  */
        text-align: center;
        padding: 1em;
        background-color: #e8503a;          /* added a red background color to the heading ↩
             cells  */
        color: white;           }                               /* added a white font color  ↩
            to the heading text */

tr {
        height: 1em;    }

table tr:nth-child(even) {                     /* added all even rows a #eee color  */
        background-color: #eee;         }

table tr:nth-child(odd) {                      /* added all odd rows a #fff color  */
        background-color:#fff;          }

</style>
```

Other elements we changed to enhance the better-looking aspects:

1. `font-family` - adding a custom font of yours can be a significant improvement to the design of the table
2. `border` - removing the border of the table and table cells will give it a cleaner and flatter view.
3. `even` and `odd` row colors - it makes information inside the table easier to catch.

Now look at this pleasant table view we get:

NAME	SURNAME	AGE
Fabio	Cimo	20
Jasmin	Brown	18
James	Wick	35
Donald	Dawn	29
Maggie	Grace	15

Figure 4.12: A Final View at our Table

4.3.6 Conclusion

CSS Tables are a great way to organise information inside a webpage as long as we need to structure information.We use them heavily when we want to have our fully customized properties applied regarding the styles and design.

At other cases, you may also use a framework like Bootstrap to create tables faster and with a default style.For more practice on css tables and ideas on table design, check the source files below.

4.4 Download

Download You can download the full source code of this example here: **CSS Tables**

Chapter 5

CSS Button Style

In this article, we're going through CSS Buttons. As we all know, buttons are important elements on pages and we use them to visually beautify common actions like **download**, **upload**, **log in/out**, **submit** and so on. A good button design enhances usability.

Before we explain how to style buttons, keep in mind that buttons are not links. The main purpose of a link is to navigate between pages and views, whereas buttons allow you to perform an action (such as submit).

Make sure to follow each step to get the intended results. In the end you will be able to create your own custom styled buttons.

5.1 Prerequisites

Before we start creating and styling buttons please consider downloading Font Awsome icon font pack. These icons will be attached to buttons later when we have created the basics.

After downloading, add the folders **css** and **fonts** inside your project folder like this:

Figure 5.1: Folder View after adding the Font Awesome Folders

Also, create your basic HTML file that looks like this:

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS Buttons</title>
</head>
<body>

        <!-- STYLE SECTION -->
```

```
        <style type="text/css">

        </style>


        <!-- HTML SECTION -->

</body>
</html>
```

5.2 Basic Styling

In this section, we're going to create a basic button using html and css.

5.2.1 Setting up the HTML

There are 3 main ways you can create a button starting from html.

1. Use the `a` (anchor) tag to create the link and give it a class, which by default if not styled as a button.
2. Use the `button` tag that html5 offers and you have a basic styled button with no css at all.
3. Use the `input` tag and give it a class of button and a type of submit. That will create a pre-styled button.

Your code with these three lines would be:

```
<!-- HTML SECTION -->

<a class="button" href="#"<Anchor Button</a>
<button class="button">Button Tag</button>
<input class="button" type="submit">
```

This is what the basic buttons would look like with no css properties applied.

Figure 5.2: HTML - Basic Buttons

But for the sake of creating a button from scratch, we will only use the first method that is using an anchor tag, but add a `div` tag with a class of `btn-wrapper` because it will be useful as some properties cannot be applied over the anchor tag.

```
<a class="button" href="#">Button</a>
```

Until now, it looks like just a link, but we will change that with css.

5.2.2 Setting up the CSS

Lets give this `button` class some initial attributes to get the view of a button:

`border:0.1em #333336 solid` - gave the border respectively a width, a color and a style.

`border-radius:0.2em` - gave the border a radius of 0.2em (seems normal).

`text-decoration:none` - given this attribute, the text will not be underlined as the default browser link underline attribute.

`color:black` - this will give the text a black color, overriding the default blue color set by the browser

`padding:0.5em 1em` - gave the text inside the button a padding of 0.5em top and bottom and 1em left and right.

`background-color:#f3f3f3` - gave the button a light gray background color.

Lets also place it in a more pleasant position in order to have a better view.

We do that by giving the `btn-wrapper` class (from the div) margins.

The CSS code will look like this:

```
.btn-wrapper {
        margin-top: 5em;
        margin-left: 5em;
}

.button{
        border: 0.1em #333336 solid;
        border-radius: 0.2em;
        text-decoration: none;
        color: black;
        padding: 0.5em 1em;
        background-color: #f3f3f3;
}
```

Given these attributes, we have created a basic styled button that looks like this in the browser:

Figure 5.3: Basic Styled Button

5.2.3 Button States

In addition to the default state, buttons can have two other states: `hover` and `active`, which respectively mean *mouse over* and *clicked/pressed*. Below we will show the button how to act/change when the cursor is over it and when it is pressed.

1. Hover State

The hover state can be achieved by adding `:hover` pseudo class like below:

```
.button:hover {
        background-color: #cecece;  /* added an intense gray color in active state */
}
```

This would be the pressed button view:

Figure 5.4: Hover State of Button

2. Active State

The active state can be achieved by adding `:active` pseudo class like below:

```
.button:active {
        background-color: #a2b2bc;  /* added a blue color in active state */
}
```

This would be the active state (clicked) view of the button:

Figure 5.5: Active State of the Button

Note that when entering attributes about the other states rather than the default, you should only write attributes that are going to change when the button will be pressed, so it is not necessary to write again the padding or border-radius, these attributes will remain as the previous state.

As you can see, I gave it only the `background-color` attribute because that was what I needed to change, but you can also change the text color or border color when considering these states.

5.3 It's all about design!

From now on, we are going to use a custom font that is Lato and font icons we downloaded. Let's add the necessary links in the `head` section:

```
<!-- ADD CUSTOM FONT LATO -->
<link href='http://fonts.googleapis.com/css?family=Lato&subset=latin,latin-ext' rel=' ↩
    stylesheet' type='text/css'>

<!-- ADD FONTAWESOME ICONS -->
<link rel="stylesheet" href="css/font-awesome.min.css">
```

Also, add the css code to have one font for everything:

```
* {                                                                    /*  ↩
    means apply to everything on the page */
        font-family: "Lato", "sans-serif";              /* added cutsom font lato in css */
}
```

5.3.1 Start Small

You can have a pretty nice button view with color alternation in aspects of background and text.

One first advice is, you don't always need border for the default state.

Have a look at the following attributes:

```
.button{
        border-radius: 0.5em;                   /* increased border-radius */
        text-decoration: none;
        color: white;                                   /* changed text color to white */
        padding: 1em 3em;                               /* increased padding for a larger  ↩
            button */
        background-color: #329bd8;              /* changed background color to a nice blue  ↩
            */
        text-transform: uppercase;              /* made the text uppercase */
        font-weight: bold;                              /* gave the text a bold look */
}

.button:hover {
        background-color: transparent;  /* changed the bg-color to transparent */
        border: 0.15em #329bd8 solid;   /* set a border to a blue color */
        color: #329bd8;                                 /* set a text color to the same  ↩
            color */
}

.button:active {
        background-color: transparent;
        border: 0.15em #5e8ca5 solid;
        color: #5e8ca5;                                 /* minor text color change in a  ↩
            deeper blue */
}
```

This nice button would look this way in its 3 states:

Figure 5.6: Simple Button Design

5.3.2 Icons on Buttons

Going further into what's called a good user experience, this time with buttons, we will add an icon next to the text which will indicate what the button is for.

Icons are very easy to add, just find the one you want from here and copy the html code of the icon and paste it before the button text to make it sit right next to the text.

Look how I've added an `upload` icon to the button:

```
<!-- added <i class="fa fa-upload"></i> next to the button text-->
<<,<i class="fa fa-upload"></i>Upload>>
```

You will also need a few lines of css to make this look fine:

(only the commented lines are new or edited)

```
.button{
        border-radius: 0.5em;
        text-decoration: none;
        color: white;
        padding: 1em;                                   /* removed the right and left  ↩
            padding value */
        padding-right: 3em;                             /* added only padding-right to  ↩
            align the icon left      */
        background-color: #329bd8;
        text-transform: uppercase;
        font-weight: bold;
}

.fa-upload {
        padding-right: 2em;                             /* added padding-right to space it  ↩
            from the text */
        font-size: 1.2em;                               /* increased font-size to fit the  ↩
            button */
}
```

The button view in the browser:

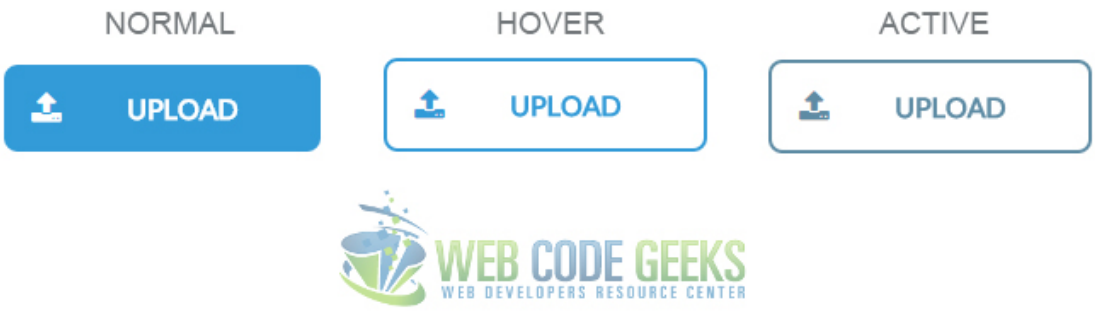

Figure 5.7: Button with Icon Attached

Icons can be a good point to consider for buttons. These are just some other buttons with icons:

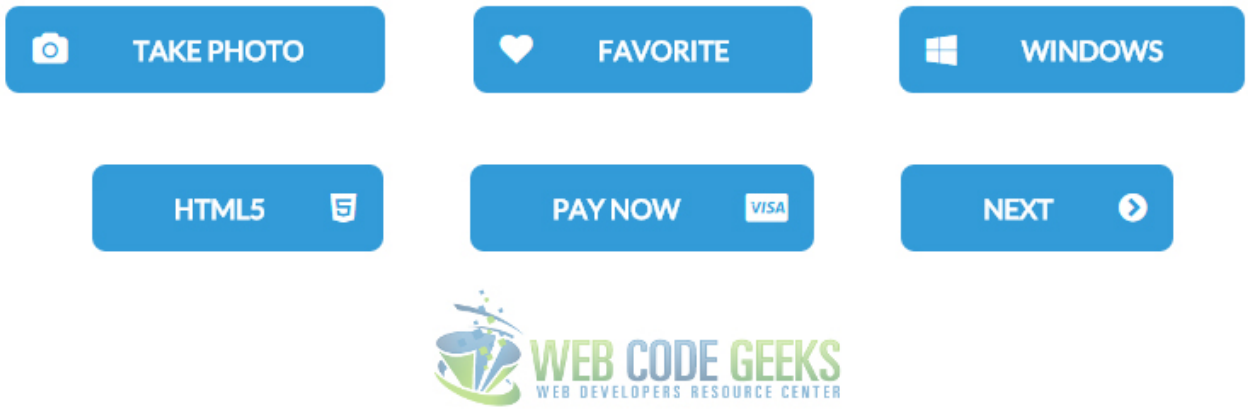

Figure 5.8: buttons10

You can change icon fonts attributes just like you do with text.

5.3.3 Gradients on Buttons

Just like you apply background colors to buttons, you can also apply *gradient*s on them.

Gradients make buttons look more like 3D rather than flat like we've seen until now.

Creating gradients yourselves is just time-wasting, so I suggest you generate gradients online in this website

Feel free to choose any gradient you like and copy the css. To be coherent with what we've started Imma choose a blue gradient.

Look at the code below:

```
/* just removed background-color attribute and added the custom css code */

.button{
        border-radius: 0.5em;
        text-decoration: none;
        color: white;
        padding: 1em;
        padding-right: 3em;
        text-transform: uppercase;
        font-weight: bold;
        text-shadow: -1px -1px 0 rgba(0,0,0,0.3);
        color: #FFFFFF;
        background-color: #49c0f0; background-image:
        -webkit-gradient(linear, left top, left bottom, from(#49c0f0), to(#2CAFE3));
        background-image: -webkit-linear-gradient(top, #49c0f0, #2CAFE3);
        background-image: -moz-linear-gradient(top, #49c0f0, #2CAFE3);
        background-image: -ms-linear-gradient(top, #49c0f0, #2CAFE3);
        background-image: -o-linear-gradient(top, #49c0f0, #2CAFE3);
        background-image: linear-gradient(to bottom, #49c0f0, #2CAFE3);
        filter:progid:DXImageTransform.Microsoft.gradient
                    (GradientType=0,startColorstr=#49c0f0, endColorstr=#2CAFE3);
                }


.button:hover {
         background-color: #1ab0ec; background-image:
         -webkit-gradient(linear, left top, left bottom, from(#1ab0ec), to(#1a92c2));
```

```
        background-image: -webkit-linear-gradient(top, #1ab0ec, #1a92c2);
        background-image: -moz-linear-gradient(top, #1ab0ec, #1a92c2);
        background-image: -ms-linear-gradient(top, #1ab0ec, #1a92c2);
        background-image: -o-linear-gradient(top, #1ab0ec, #1a92c2);
        background-image: linear-gradient(to bottom, #1ab0ec, #1a92c2);
        filter:progid:DXImageTransform.Microsoft.gradient
                    (GradientType=0,startColorstr=#1ab0ec, endColorstr=#1a92c2);
}
```

In the code above I did only consider the `normal` and `hover` state of the button.

It's up to you to add the active state if you think you need it.

This is the how the button we'd get:

Figure 5.9: Gradient Applied on Button

That's it on gradients, try more to see how you can get creative on this.

These are just some of them I made for you.

Figure 5.10: Gradients Button Design

5.3.4 Patterns on Buttons

In addition to gradient backgrounds, buttons can have **pattern** backgrounds.

You can find a great gallery of css patterns here, feel free to use any of them.

The way we use them is just like gradients, remove any background color we have and paste the css code for the pattern.

Look at the code below:

```
.button{
        border-radius: 0.5em;
        text-decoration: none;
        color: white;
        padding: 1em;
        padding-right: 3em;
        text-transform: uppercase;
        font-weight: bold;
        background-color: #6d695c;
        background-image:
        repeating-linear-gradient(120deg, rgba(255,255,255,.1), rgba(255,255,255,.1) 1px,  ↩
            transparent 1px, transparent 60px),
        repeating-linear-gradient(60deg, rgba(255,255,255,.1), rgba(255,255,255,.1) 1px,  ↩
            transparent 1px, transparent 60px),
        linear-gradient(60deg, rgba(0,0,0,.1) 25%, transparent 25%, transparent 75%, rgba ↩
            (0,0,0,.1) 75%, rgba(0,0,0,.1)),
        linear-gradient(120deg, rgba(0,0,0,.1) 25%, transparent 25%, transparent 75%, rgba ↩
            (0,0,0,.1) 75%, rgba(0,0,0,.1));
        background-size: 70px 120px;
}


.button:hover {
        background:
        radial-gradient(black 15%, transparent 16%) 0 0,
        radial-gradient(black 15%, transparent 16%) 8px 8px,
        radial-gradient(rgba(255,255,255,.1) 15%, transparent 20%) 0 1px,
        radial-gradient(rgba(255,255,255,.1) 15%, transparent 20%) 8px 9px;
        background-color:#282828;
        background-size:16px 16px;
}
```

In this code I just added pattern backgrounds for the normal and hover state of the button.

This is how the pattern styled button would look like:

Figure 5.11: Gradients on Buttons

You can play around with these and see a lot of good looking buttons you can create. Here are a few I created for you:

Figure 5.12: Getting Creative with Patterns on Buttons

5.4 Conclusion

To conclude, we can state that there are various ways you can style and fit buttons according to your needs.

You can add png images instead of icons or images as backgrounds into a button, but after some time practicing you will understand that keeping it simple and nice is a combinations you can achieve by trying either flat or gradient design, which will also make your project less cluttery from images on each button.

Below you can find some pre-styled and good designed buttons that you can use for your own projects.

5.5 Download

Download You can download the full source code of this example here: **CSS Buttons**

Chapter 6

CSS Input Type Text

In this example, we'll have a look at how we can style text inputs.

Text inputs are very common in websites nowadays like in sign up forms, contact forms, search boxes, survey answers and so on.

But you notice most of them are not styled, and you can see a bunch of text fields with the same default view accross different pages.

Well, the `input type text` already has a default/pre-styled view, but most of the times, you will be willing to change it.

As always, lets first look at the basics and then some custom stuff.

6.1 Prerequisites

First, go ahead and create a html file with its basic syntax inside like this:

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS3 Text Input Styling Example</title>
</head>
<body>
<!-- STYLE  SECTION -->
<style type="text/css">

</style>

<!-- HTML SECTION  -->

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

Set up an input with a type of text, a class of element and optionally a name of city:

```
<!-- HTML SECTION  -->

<input type="text" class="element" name="city">
```

Note that the name will be needed because the text input is going to be used under some form and referred to.

Look at the default styling of a text input:

```
input[type=text] {
  background-color: #ffffff;
  border: 1px solid #cccccc;
  -webkit-border-radius: 3px;
     -moz-border-radius: 3px;
          border-radius: 3px;
  -webkit-box-shadow: inset 0 1px 1px rgba(0, 0, 0, 0.075);
     -moz-box-shadow: inset 0 1px 1px rgba(0, 0, 0, 0.075);
          box-shadow: inset 0 1px 1px rgba(0, 0, 0, 0.075);
  -webkit-transition: border linear 0.2s, box-shadow linear 0.2s;
     -moz-transition: border linear 0.2s, box-shadow linear 0.2s;
      -ms-transition: border linear 0.2s, box-shadow linear 0.2s;
       -o-transition: border linear 0.2s, box-shadow linear 0.2s;
          transition: border linear 0.2s, box-shadow linear 0.2s;
}

input[type=text]:focus {
  border-color: rgba(82, 168, 236, 0.8);
  outline: 0;
  outline: thin dotted \9;
  /* IE6-9 */

  -webkit-box-shadow: inset 0 1px 1px rgba(0, 0, 0, 0.075), 0 0 8px rgba(82, 168, 236, 0.6) ↩
      ;
     -moz-box-shadow: inset 0 1px 1px rgba(0, 0, 0, 0.075), 0 0 8px rgba(82, 168, 236, 0.6) ↩
         ;
          box-shadow: inset 0 1px 1px rgba(0, 0, 0, 0.075), 0 0 8px rgba(82, 168, 236, 0.6) ↩
              ;
}
```

And the default view of a text input as you might know is:

Figure 6.1: Text Input Default Style

6.2 Styling a Text Input

In this section, we will style the above element with CSS3 to create a better looking input view. Look at the lines below:

```
<style type="text/css">

.element {
        font-family: "Montserrat", sans-serif;  /* custom font applied */
        width: 20em;                                                  /* custom width */
        height: 3em;                                                  /* custom height */
        border: 0.1em solid #ccc;                                  /* custom border */
        border-radius: 0.5em;                                      /* added border radius */
        background-color: #feffdc;                                 /* light yellow background  ↩
            */
        padding: 0em 1em;                                                 /* right and left  ↩
            padding */
```

```
}

</style>
```

Let's have a look at what we styled:

1. Broder - you will probably need a custom border for your input because the default one is really old inset shadow.
2. Border-Radius - most text inputs on pages have this non-zero radius on their borders, it makes them look better.
3. Background-Color - you may optionally use a bg color just to emphasize the input field, it is eye catching after all.
4. Padding - use paddings when width and height are changed and the text seems creepy starting from the very beginning.

We've pretty much given custom attributes to most of the properties. This would look like this:

Figure 6.2: Custom Styled Text Input

Now lets see another styling of the input text.

```
<style type="text/css">

.element {
        font-family: "Montserrat", sans-serif;
        width: 30em;
        height: 4em;
        border: 0.1em solid #ccc;
        border-radius: 0.5em;
        background-color: #efefef;
        padding: 0em 1em;
        font-size: 1.2em;
        box-shadow: 0em 0.1em 0.5em #ccc;
}

</style>
```

In this second example, we did some noticeable changes to the input:

1. We are having a much larger input box which can be considered as comfortable for full page forms.
2. We added a box-shadow which gave our inputs a 3D-like view which made it more attractive.
3. Basically, you can change the bg color to whatever suits your needs according to your page design.
4. Remember to always use custom fonts and font-sizes to adjust the right properties of the input.

Have a look at what this looks like in the browser:

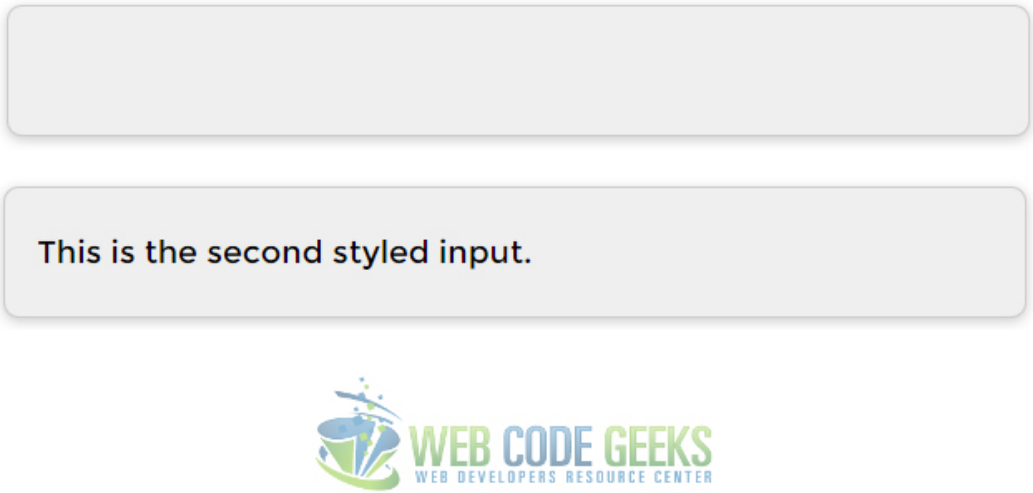

Figure 6.3: Another Styling for the Text Input

See how in the second example we added a box-shadow property on the input and that made it look more professional.

6.3 Advanced Input Styling

Notice that up until now, we have been styling the text input by giving it a class (`element`), but what if you want to style all text inputs the same way? Well, there is a way. You can select all input type text like this in css: `input[type=text]`. First lets add a few more lines create a form like view of the inputs and then apply styling using the selector.

```
<!-- HTML SECTION  -->
<h4>Please, fill the form below to continue:</h4>
<form class="form-group" method="post">
        <h4>First Name</h4>
        <input type="text" placeholder="First Name" name="first-name"><br />
        <h4>Last Name</h4>
        <input type="text" placeholder="Last Name" name="last-name"><br />
        <h4>E-Mail</h4>
        <input type="text" placeholder="E-Mail Address" name="email"><br />
        <h4>Country</h4>
        <input type="text" placeholder="Country" name="country"><br />
</form>
```

That gave a form view of several inputs. A more intense styling with advanced shadows and inset would be:

```
<!-- STYLE  SECTION -->

<style type="text/css">

input[type=text] {
        font-family: "Open Sans", sans-serif;
        width: 20em;
        height: 2em;
```

```
        border: 0.1em solid #ccc;
        border-radius: 0.2em;
        background-color: #f9f9f9;
        padding: 0em 1em;
        font-size: 1em;

        border: 5px solid white;
    -webkit-box-shadow:
      inset 0 0 8px  rgba(0,0,0,0.1),
            0 0 16px rgba(0,0,0,0.1);
    -moz-box-shadow:
      inset 0 0 8px  rgba(0,0,0,0.1),
            0 0 16px rgba(0,0,0,0.1);
    box-shadow:
      inset 0 0 8px  rgba(0,0,0,0.1),
            0 0 16px rgba(0,0,0,0.1);
    padding: 15px;
    background: rgba(255,255,255,0.5);
    margin: 0 0 10px 0;
}

h4 {
        margin-bottom: 0em;
        padding-bottom: 0.5em;
        font-family: "Montserrat";
}
```

The new, smart styled all inputs would look like this:

Figure 6.4: Styling all inputs with a type of Text using the CSS Selector

6.4 Conclusion

To conclude, there is a lot to explore while styling inputs of every type, especially text, and it is recommended to giveyour inputs a personalized view and keep a standard style that you create for later uses, just like we did with text inputs.

The default input view is kinda creepy and old-fashioned, so consider using your own styling when using inputs.

You could also take advantage of front end frameworks like Bootstrap to have pre styled inputs with nice animated hovers.

6.5 Download

Download You can download the full source code of this example here: **CSS3 Text Input Styling**

Chapter 7

CSS Text Decoration

The aim of this example is to show the usage of text-decoration property.

This is a very commonly used property to decorate text mainly with lines. It may also be used to style links in your own way using underline value.

All modern browsers support text-decoration property and you may use it without having to write extra lines of code like -webkit, -moz or -o.

As we will see, you can add two or more values to this property and still get things going well and have a customized styling that you want.

First, we have a look at the basics and then see more cases and examples.

7.1 Basic Setup & Application

Go ahead and create a new html document and add the basic sytnax in it like so:

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS3 Text Decoration Example</title>
</head>
<body>

<!-- STYLE SECTION -->

<style type="text/css">

</style>

<!-- HTML SECTION -->

</body>
</html>
```

Lets first explain how the property is used and values it can take:

1. Usage: `text-decoration:value;`
2. Values: `underline`, `overline`, `line-through`

A basic application of the text-decoration property would be the following. Create a some lines of text in html:

```
<!-- HTML SECTION  -->

<h1>this line is underlined</h1>
<h2>this line is overlined</h2>
<h3>this line has a line through</h3>
```

Applying the property in css would look like so:

```
<!-- STYLE  SECTION -->

<style type="text/css">

h1 {
        text-decoration: underline;
}

h2 {
        text-decoration: overline;
}

h3 {
         text-decoration: line-through;
}

</style>
```

The view in the browser for the basic application would be:

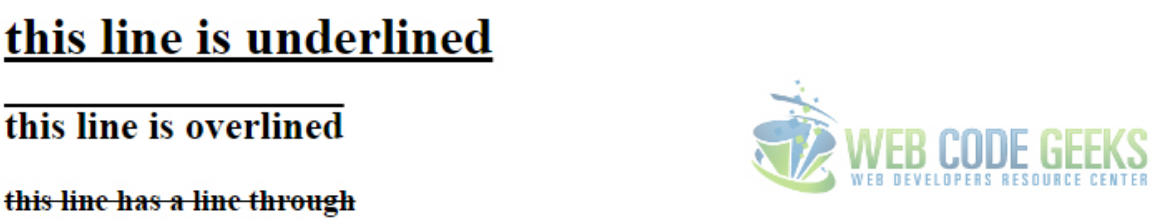

Figure 7.1: Basic Application of Text-Decoration Property

7.2 Cases and Examples

In this sections, we will cover things that you can do with text-decoration and some interesting ways how you can use it. Notice that the following examples will work best in Firefox, which has full support of them.

7.2.1 Multiple Values Example

In addition to a single attribute value, `text-decoration` can also take multiple values. Look at the code below:

-HTML

```
<h2 class="mix">this line will have multiple values</h2>
```

Lets add all three common values in css: underline, overline and line-through like below:

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

The view in the browser would be:

Figure 7.2: Multiple Common Values Applied

But you can also use multiple values to add other kind of styling like line style or line color.

-HTML

```
<h1 class="fancy">this line will have line styling</h1>
```

Lets add a dotted line style and red color to it to make this underline fancy.

```
.fancy {
        text-decoration: underline dotted red; /* works only on firefox */
}
```

Currently only supported in Firefox, this styling would look like this in Mozilla Firefox:

Figure 7.3: Line Styling using Multiple Attribute Values

7.2.2 The Text-Decoration Family

The text-decoration property is a shorthand to some specific related properties like text-decoration-color/line/style.

1. `text-decoration-color`

-HTML

```
<h2 class="color">This is the color styled line</h2>
```

-CSS

```
.color {
        text-decoration: underline;
        text-decoration-color: #40b7c2;
```

We did give the line a color, the view is:

Figure 7.4: Text-Decoration-Color Attribute

1. `text-decoration-line`

-HTML

```
<h2 class="line">This is the under/over/through line</h2>
```

-CSS

```
.line {
        text-decoration-line: underline;
}
```

We just underlined the sentence, but in the long way, the view is:

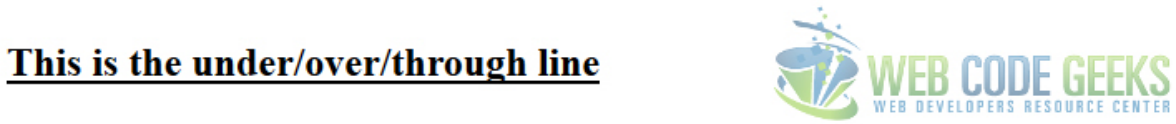

Figure 7.5: Text-Decoration-Line Attribute

1. `text-decoration-style`

-HTML

```
<h2 class="style">This is the styled line</h2>
```

-CSS

```
.style {
        text-decoration: underline;
        text-decoration-style: dotted;  /* try solid or wavy */
}
```

Here, we gave the line a dotted style, the view is:

Figure 7.6: Text-Decoration-Style Attribute

7.3 Conclusion

To conclude, we can state that `text-decoration` property is quite handy when working with text, consider it just like the bold or color property of text, it is obvious that it is part of the text styling family. The property is very easy to use in css.

Some elements, line the anchor tags include pre-styled text with a `text-decoration:underline;`, so you may want to do the contrary in those cases, I mean reset text-decoration to none. It always depends on you specific needs for your project.

7.4 Download

Download You can download the full source code of this example here: **CSS3 Text Decoration**

Chapter 8

CSS Text Shadow Example

Ever felt chaotic about text going into unsuitable button colors, boxes or simply sitting inside a bad contrast in the page?

Well, it's probably because you've used colors that won't match to give a good view of the text, (e.g make it readable).

With CSS, it is very easy to solve this using the `text-shadow` property to enhance the text readability and contrast.

In this example, we're going to use text shadows in some really essential cases which will later enable you to explore more.

8.1 Basic Set Up

First, go ahead and create a new html file and add the basic `html` and `css` like this:

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS Buttons</title>
</head>
<body>

        <!-- STYLE SECTION -->

        <style type="text/css">

        </style>


        <!-- HTML SECTION -->

</body>
</html>
```

On the html section, add a `<h1>` tag with a class of `text` like so:

```
<!-- HTML SECTION -->

<h1 class="text">Web Code Geeks</h1>
```

And this is the element we will be adding shadow on.

I have given this element some initial properties to make it look good on the screen:

```
<!-- STYLE SECTION -->
```

```
<style type="text/css">

body{
        font-family: "Arial", "sans-serif";          /* just a custom font */
}

h1 {                                /* centered the text for a better view */
        margin-left: 7em;
        margin-top: 5em;
}
</style>
```

Now let's give the text the text-shadow attribute.

Before doing that, let us explain what are the inputs of this attribute:

```
text-shadow: 4px 4px 4px #ccc;
```

`4px` - X (horizontal) offset `3px` - Y (vertical offset `2px` - blur amount `#ccc` - color

Basically, you can see it like this:

```
text-shadow: horizontal-offset vertical-offset blur color;
```

where the color can be either in `hex #ccc;` code or `rgba(0,0,0,0.3);` code.

Applying the shadow to our text in css would look like this:

```
<!-- STYLE SECTION -->

.text {
        font-size: 5em;                                                    /* just made text ↩
             larger */
        text-shadow: 4px 3px 2px #ccc;
}
```

The view in the browser of this text with shadow would be:

Figure 8.1: Basic Shadow Applied

8.2 Variations

From now on, the `body` background color will be subject to constant change.

This is because certain shadows need specific backgrounds to be noticed.

Also the text is going to be uppercase for better results.

The following will show some great text-shadows you can apply.

8.2.1 Pressed Effect

Set your text color to a slightly darker shade than your background color.

Next, apply a slight white text-shadow with a reduced opacity.

```
body {
        background: #222;
}

.text {
        font-size: 5em;
        color: rgba(0,0,0,0.6);                                                    /*  ↩
            added text-color */
    text-shadow: 2px 2px 3px rgba(255,255,255,0.1);              /* added shadow */
}
```

Using the `rgba` color code, you will be able to set the `opacity` of the color applied.

Notice that the text-color has an opacity of 60% (0.6) and shadow 10% (0.1).

Figure 8.2: Letterpress Shadow Effect

8.2.2 Hard Shadow Effect

Because of their retro nature, hard shadows don't always need to have blur applied.

Take for example this hard text shadow:

```
body {
        background: #fff3cd;                                  /* changed body background  ↩
            color */
}

.text {
        font-size: 5em;
        color: white;  ↩
                                                                                  /*  ↩
            changed text-color to white */
    text-shadow: 6px 6px 0px rgba(0,0,0,0.2);                 /* added retro-feel shadow  ↩
        */
}
```

Figure 8.3: Retro Hard Shadow Effect

8.2.3 Double Shadow Effect

It is interesting to know that you are not limited to one single shadow application.

You can use more than one shadow like this: `text-shadow:shadow1, shadow2, shadow3;`

Let's add two shadows, one of them with the color of the background, and the other one a slightly darker color than the background color:

```
.text {
        font-size: 5em;
    text-shadow: 4px 3px 0px #fff, 9px 8px 0px rgba(0,0,0,0.15);    /* given two shadows */
}
```

In this case, the background is white, so we don't need a custom color for it.

The view in the browser would be:

Figure 8.4: Double Shadow Effect

8.2.4 Distant Down Shadow Effect

This effect lies on the multi-shadow capability of css.

Below, you can see 4 shadows pointing down with various degrees.

```
body {
        background: #fff3cd;                    /* changed body background color */
}

.text {
        font-size: 5em;
        color: white;
    text-shadow: 0px 3px 0px #b2a98f,
                 0px 14px 10px rgba(0,0,0,0.15),
                 0px 24px 2px rgba(0,0,0,0.1),
                 0px 34px 30px rgba(0,0,0,0.1);
}
```

Figure 8.5: Distant Down Shadow Effect

8.2.5 Mark Dotto's 3D Text Effect

The following example is just as impressive as you might be wondering.

It comes from MarkDotto.com and utilizes an impressive 12 separate shadows to pull off a very believable 3D effect.

```
body {
        background: #3495c0;                    /* changed body background color */
}

.text {
        font-size: 5em;
        color: white;
    text-shadow: 0 1px 0 #ccc,
                 0 2px 0 #c9c9c9,
                 0 3px 0 #bbb,
                 0 4px 0 #b9b9b9,
                 0 5px 0 #aaa,
                 0 6px 1px rgba(0,0,0,.1),
                 0 0 5px rgba(0,0,0,.1),
                 0 1px 3px rgba(0,0,0,.3),
                 0 3px 5px rgba(0,0,0,.2),
                 0 5px 10px rgba(0,0,0,.25),
                 0 10px 10px rgba(0,0,0,.2),
                 0 20px 20px rgba(0,0,0,.15);
}
```

Now look at this:

Figure 8.6: 3D Shadow Effect

8.2.6 Gordon Hall's True Inset Text Effect

Gordon uses some serious CSS voodoo to pull off not only an outer shadow but a genuine inner shadow as well.

```
body {
        background: #cbcbcb;                    /* changed body background color */
}

.text {
        font-size: 5em;
        color: transparent;
        background-color: #666666;
                  -webkit-background-clip: text;
                  -moz-background-clip: text;
                  background-clip: text;
                  text-shadow: rgba(255,255,255,0.5) 0px 3px 3px;
}
```

And that gives and incredible true inset text effect.

Figure 8.7: Text Inset Effect

8.2.7 Glowing Text Shadow Effect

```
body {
        background: #992d23;                    /* changed body background color */
```

```
}

.text {
        font-size: 5em;
        color: white;
        text-shadow: 0px 0px 6px rgba(255,255,255,0.7);
}
```

This shadow will create the effect of a glowing text.

Figure 8.8: Glowing Text - Shadow Effect

8.2.8 Soft Emboss Shadow Effect

```
body {
        background: #629552;                    /* changed body background color */
}

.text {
        font-size: 5em;
        color: rgba(0,0,0,0.6);
        text-shadow: 2px 8px 6px rgba(0,0,0,0.2),
                0px -5px 35px rgba(255,255,255,0.3);
}
```

Figure 8.9: Soft Emboss - Text Shadow Effect

8.3 Conclusion

As you saw in this example, the text-shadow property is really easy to use.

What's interesting, is that you can come up with creative effects if you put yourself into it.

These effects are used all over the web (I did not *invent* them) so be sure you are using qualitative stuff.

You can also download the source html and just edit these examples that you saw above.

8.4 Download

Download You can download the full source code of this example here: **CSS Text Shadow**

Chapter 9

CSS Box Shadow Example

In this example, we will go through the box-shadow property of css.

Just like the text-shadow property, box-shadow will give an element of html a new look by adding different styled shadows.

As you can imagine, this is going to be more of an examples show rather than exaplanation because the property is pretty straight-forward.

The cases you'll see below, are of a very high usage all over the web, and I recommend you consider these shadow designs on your projects.

They will make content on boxes in your website more catchy and attractive, so do not hesitate to use them.

9.1 Prerequisites

Below, I will present you to the necessary knowledge you should have to get through this.

9.1.1 Basic Set Up

First, go ahead and create a new html file and add the basic `html` and `css` like this:

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS Buttons</title>
</head>
<body>

        <!-- STYLE SECTION -->

        <style type="text/css">

        </style>


        <!-- HTML SECTION -->

</body>
</html>
```

9.1.2 Pseudo-Elements :before and :after

A pseudo-element creates a phoney element and inserts it before or after the content of the element that you've targeted.

We will use the :before and :after pseudo-elements to add shadows left, right, top, bottom or combinations of these.

See this simple example of its usage so you have a better clue:

- HTML

```
<!-- HTML SECTION -->
<h1 class="text">This is the first item</h1>
```

- CSS

```
<!-- STYLE SECTION -->

.text:before {
        content:"1. ";    /* what to display before text element */
}

.text:after {
        content: "!";    /* what to display after text element */
}

</style>
```

This would give the number `1.` before the text and `!` after the text like this:

Figure 9.1: :before and :after pseudo elements

9.1.3 Extra properties to consider

Along with the pseudo-elements, there are also some properties I'd like you to understand:

`z-index` - specifies the stack order of an element.

`position` - allows you to position elements within a page.

`content` - used when appyling :before and :after pseudo-elements.

`left, right, top, bottom` - defines the left, right, top, bottom edge of an element.

`transform` - self-explanatory, transforms an object by degree, scale ect.

9.2 Basic Box-Shadow Property

Application of the box-shadow property is pretty simple. It goes like this:

```
box-shadow: horizontal-offset vertical-offset blur-radius spread-radius color;
```

9.2.1 Attribute values explanation

But what do these attributes mean?

1. The horizontal offset (required):

- A positive value will make the shadow lie on the right of the box.
- A negative offset will make it lie on the left of the box.

2. The vertical offset (required):

- A negative value makes the shadow lie above the box.
- A positive value make it lie below the box.

3. The blur radius (required):

- Sharp shadows come with a value of 0, and the higher the number, the more blurred it will be, and the further out the shadow will extend.

4. The spread radius (optional):

- Positive values increase the size of the shadow.
- Negative values decrease the size.

Default is 0 (the shadow is same size as blur).

5. Color (required):

- Using HEX color: e.g. #ccc, #9895AF ect.
- Using RGBA color: e.g. rgba(192,192,192,0.3);

9.2.2 Application of the basic property

Create a `div` with a `class` of `box`.

```
<!-- HTML SECTION -->

        <h3>WEB CODE GEEKS</h3>
```

We are using multiple classes on an element, in this case two, because the box element will be the same for all examples, while the other class will contain distinctive attributes according to the effect we want to create.

Now give the `body` a light gray color and `box` element these initial attributes to create a standard now.

```
<!-- STYLE SECTION -->
body {
        background:#E6E6E6;
}

.box {
    width:70%;
    height:200px;
    background:#FFF;
    margin:40px auto;
}
```

And the basic application of the `box-shadow` property would be:

```
 /* no horizontal shadow, 10px vertical shadow, 6px blur, -6px spread radius, gray color */

.basic{
    -webkit-box-shadow: 0 10px 6px -6px #777;
       -moz-box-shadow: 0 10px 6px -6px #777;
            box-shadow: 0 10px 6px -6px #777;
}
```

A basic shadow applied to a box like above will look like this:

Figure 9.2: Basic Box-Shadow Application

9.3 Advanced 3D Looking Box Shadows

Now, let's look at some popular designs with shadows.

For the first effect, every line of code is commented so that you know what we are doing.

9.3.1 Soft Box Shadow on Both Sides

```
 /* EFFECT 1 */
/* Here shadows are styled as elements to give impressive look */
/* However, the box shadow property is applied at the end to make final touches */
```

```
.effect1
{
  position: relative;   /* relative positioning referring to before and after */
}
.effect1:before, .effect1:after  /* apply some mutual properties before and after this  ↩
    element */
{
  z-index: -1;                  /* send this element backward */
  position: absolute;   /* absolute positioning referring to effect2 */
  content: "";                  /* just required, no need to put anything inside  */
  bottom: 15px;                 /* bottom shadow alignment, less is closer to bottom */
  left: 10px;                   /* consider this like a margin left */
  width: 50%;                   /* width of the shadow element background */
  top: 80%;                             /* consider this as margin top of the shadow  ↩
      element */
  max-width:300px;              /* restricts the max width of the shadow element  */
  background: #777;             /* gives a background color to the shadow element  */
  -webkit-box-shadow: 0 15px 10px #777; /* compatibility case application */
  -moz-box-shadow: 0 15px 10px #777;    /* compatibility case application */
  box-shadow: 0 15px 10px #777; /* applied the basic box-shadow property */
  /* rotation of shadows/elements gives that 3D look to the box */
  -webkit-transform: rotate(-3deg);
  -moz-transform: rotate(-3deg);
  -o-transform: rotate(-3deg);
  -ms-transform: rotate(-3deg);
  transform: rotate(-3deg);
}
.effect1:after
{
  -webkit-transform: rotate(3deg);      /* aligns the shadow right */
  -moz-transform: rotate(3deg);
  -o-transform: rotate(3deg);
  -ms-transform: rotate(3deg);
  transform: rotate(3deg);
  right: 10px;                                  /* consider this like a margin right */
  left: auto;                                   /* leave this auto to automatically set the ↩
       left */
}
```

View:

Figure 9.3: Soft Shadow on Both Sides

9.3.2 Left and Right Box Shadow

In this effect, we take a look at the left and right box-shadows.

```
/* LEFT SHADOW */

.effect2
{
  position: relative;
}
.effect2:before
{
  z-index: -1;
  position: absolute;
  content: "";
  bottom: 15px;
  left: 10px;
  width: 50%;
  top: 80%;
  max-width:300px;
  background: #777;
  -webkit-box-shadow: 0 15px 10px #777;
  -moz-box-shadow: 0 15px 10px #777;
  box-shadow: 0 15px 10px #777;
  -webkit-transform: rotate(-3deg);
  -moz-transform: rotate(-3deg);
  -o-transform: rotate(-3deg);
  -ms-transform: rotate(-3deg);
  transform: rotate(-3deg);
}

/* RIGHT SHADOW */

.effect3
{
  position: relative;
}
.effect3:after
{
  z-index: -1;
  position: absolute;
  content: "";
  bottom: 15px;
  right: 10px;
  left: auto;
  width: 50%;
  top: 80%;
  max-width:300px;
  background: #777;
  -webkit-box-shadow: 0 15px 10px #777;
  -moz-box-shadow: 0 15px 10px #777;
  box-shadow: 0 15px 10px #777;
  -webkit-transform: rotate(3deg);
  -moz-transform: rotate(3deg);
  -o-transform: rotate(3deg);
  -ms-transform: rotate(3deg);
  transform: rotate(3deg);
}
```

View:

Figure 9.4: Right and Left Box-Shadow Applied

9.3.3 Hard Box Shadow on Both Sides

In this example, we look at a 3D like effect on both sides box shadow.

```
/* EFFECT 4 */

.effect4
{
  position: relative;
}
.effect4:before, .effect4:after
{
  z-index: -1;
  position: absolute;
  content: "";
  bottom: 25px;
  left: 10px;
  width: 50%;
  top: 80%;
  max-width:300px;
  background: #777;
  -webkit-box-shadow: 0 35px 20px #777;
  -moz-box-shadow: 0 35px 20px #777;
  box-shadow: 0 35px 20px #777;
  -webkit-transform: rotate(-8deg);
  -moz-transform: rotate(-8deg);
  -o-transform: rotate(-8deg);
  -ms-transform: rotate(-8deg);
```

```
  transform: rotate(-8deg);
}
.effect4:after
{
  -webkit-transform: rotate(8deg);
  -moz-transform: rotate(8deg);
  -o-transform: rotate(8deg);
  -ms-transform: rotate(8deg);
  transform: rotate(8deg);
  right: 10px;
  left: auto;
}
```

View:

Figure 9.5: Both Sides Box Shadow

9.3.4 Soft Inset Shadow

You can also set inset shadows in a box element.

```
/* EFFECT 5 */

.effect5
{
    position:relative;
    -webkit-box-shadow:0 1px 4px rgba(0, 0, 0, 0.3), 0 0 40px rgba(0, 0, 0, 0.1) inset;
       -moz-box-shadow:0 1px 4px rgba(0, 0, 0, 0.3), 0 0 40px rgba(0, 0, 0, 0.1) inset;
            box-shadow:0 1px 4px rgba(0, 0, 0, 0.3), 0 0 40px rgba(0, 0, 0, 0.1) inset;
}
.effect5:before, .effect5:after
{
    content:"";
    position:absolute;
    z-index:-1;
    -webkit-box-shadow:0 0 20px rgba(0,0,0,0.8);
    -moz-box-shadow:0 0 20px rgba(0,0,0,0.8);
    box-shadow:0 0 20px rgba(0,0,0,0.8);
    top:50%;
    bottom:0;
    left:10px;
```

```
    right:10px;
    -moz-border-radius:100px / 10px;
    border-radius:100px / 10px;
}
.effect5:after
{
    right:10px;
    left:auto;
    -webkit-transform:skew(8deg) rotate(3deg);
       -moz-transform:skew(8deg) rotate(3deg);
        -ms-transform:skew(8deg) rotate(3deg);
         -o-transform:skew(8deg) rotate(3deg);
            transform:skew(8deg) rotate(3deg);
}
```

View:

Figure 9.6: Soft Inset Shadow

9.3.5 Top, Bottom and Left, Right Inset Shadow

These two examples show: top and bottom inset shadow, left and right inset shadow.

```
/* TOP AND BOTTOM INSET SHADOW */

.effect6
{
    position:relative;
    -webkit-box-shadow:0 1px 4px rgba(0, 0, 0, 0.3), 0 0 40px rgba(0, 0, 0, 0.1) inset;
       -moz-box-shadow:0 1px 4px rgba(0, 0, 0, 0.3), 0 0 40px rgba(0, 0, 0, 0.1) inset;
            box-shadow:0 1px 4px rgba(0, 0, 0, 0.3), 0 0 40px rgba(0, 0, 0, 0.1) inset;
}
.effect6:before, .effect6:after
{
    content:"";
    position:absolute;
    z-index:-1;
    -webkit-box-shadow:0 0 20px rgba(0,0,0,0.8);
    -moz-box-shadow:0 0 20px rgba(0,0,0,0.8);
    box-shadow:0 0 20px rgba(0,0,0,0.8);
    top:0;
```

```
    bottom:0;
    left:10px;
    right:10px;
    -moz-border-radius:100px / 10px;
    border-radius:100px / 10px;
}
.effect6:after
{
    right:10px;
    left:auto;
    -webkit-transform:skew(8deg) rotate(3deg);
       -moz-transform:skew(8deg) rotate(3deg);
        -ms-transform:skew(8deg) rotate(3deg);
         -o-transform:skew(8deg) rotate(3deg);
            transform:skew(8deg) rotate(3deg);
}

/* LEFT AND RIGHT INSET SHADOW */

.effect7
{
    position:relative;
    -webkit-box-shadow:0 1px 4px rgba(0, 0, 0, 0.3), 0 0 40px rgba(0, 0, 0, 0.1) inset;
       -moz-box-shadow:0 1px 4px rgba(0, 0, 0, 0.3), 0 0 40px rgba(0, 0, 0, 0.1) inset;
            box-shadow:0 1px 4px rgba(0, 0, 0, 0.3), 0 0 40px rgba(0, 0, 0, 0.1) inset;
}
.effect7:before, .effect7:after
{
    content:"";
    position:absolute;
    z-index:-1;
    -webkit-box-shadow:0 0 20px rgba(0,0,0,0.8);
    -moz-box-shadow:0 0 20px rgba(0,0,0,0.8);
    box-shadow:0 0 20px rgba(0,0,0,0.8);
    top:10px;
    bottom:10px;
    left:0;
    right:0;
    -moz-border-radius:100px / 10px;
    border-radius:100px / 10px;
}
.effect7:after
{
    right:10px;
    left:auto;
    -webkit-transform:skew(8deg) rotate(3deg);
       -moz-transform:skew(8deg) rotate(3deg);
        -ms-transform:skew(8deg) rotate(3deg);
         -o-transform:skew(8deg) rotate(3deg);
            transform:skew(8deg) rotate(3deg);
}
```

View:

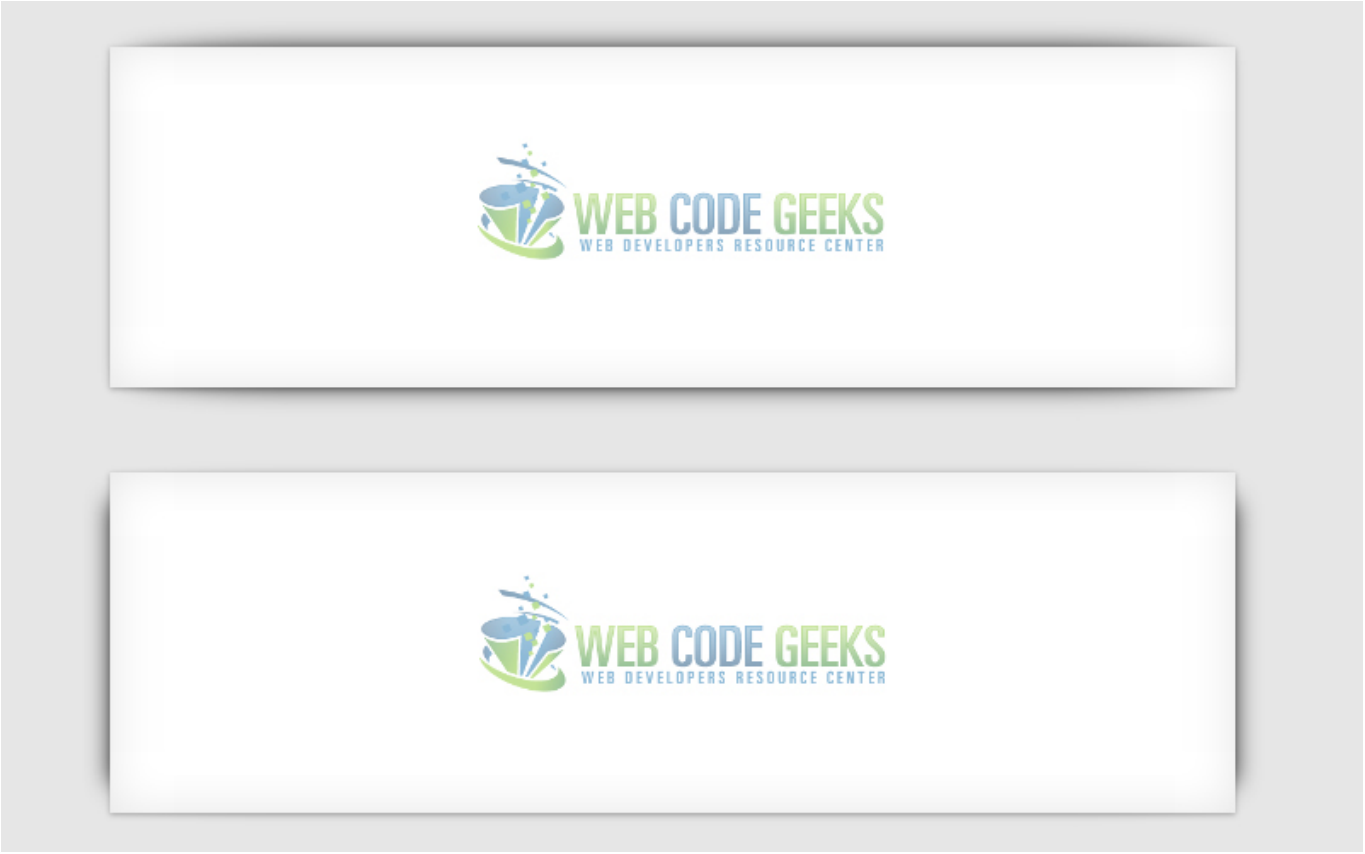

Figure 9.7: Top, Bottom & Left, Right Inset Shadow

9.4 Conclusion

Generally speaking, shadows are a great way to make some nice touches on elements.

They do create interesting look and feel of elements and that's why you should consider using them.

Sometimes the code will be complicated, but feel free to use effects we used in this example, (just copy and paste) if you like any of them just the way they are.

However, if you feel professional at this, you can change values according to your needs.

Below you can find all these shadow examples in one `html` file.

9.5 Download

Download You can download the full source code of this example here: **CSS Box-Shadow**

Chapter 10

CSS Horizontal Menu

The aim of this example is to show how we can create beautiful horizontal menus using a bit of html and more css to create some nice styling.

As you have seen on websites, menus are everywhere, it is an important part of the website, like a navigation toolbar for users to get essential links.

As long as only a few basic lines of html are needed and more lines on css, it is obvious that everything will be compatible accross browsers.

As always, we first create the basic example and then extend to a more advanced overview in css.

10.1 Basic Setup

First, go ahead and create a html file with its basic syntax inside like so:

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS3 Horizontal Menu Example</title>
</head>
<body>

<!-- STYLE SECTION -->

<style type="text/css">

</style>

<!-- HTML SECTION -->

</body>
</html>
```

10.2 Coding the Menu

Below, we will create the html and css for a basic horizontal menu.

1. HTML

The html structure of the menu is going to be like this:

1. A class named `menu` is going to wrap all menu items under a div.
2. The menu is going to be placed as an unordered list element, that is `ul`.
3. Each specific menu title is going to be under the `li` and then `a` tag.

Coming to the code, it would look like this together with some random menu items inside:

```
<!-- HTML SECTION  -->

<ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Profile</a></li>
        <li><a href="#">About</a></li>
        <li><a href="#">Contact</a></li>
</ul>
```

Well, it is still html, and we got a creepy view where the menu is vertical:

Figure 10.1: HTML Menu Unstyled

2. CSS CSS is going to make all the difference in this case, it is all about styling. First, lets remove some default styling.

```
a:-webkit-any-link {
        text-decoration: none;  /* no text underline to anchor elements */
}

.menu ul {
        list-style-type: none;  /* no bullet sign before items */
}
```

Then, we continue with making the menu items display inline, color, padding etc:

```
.menu li {
        display: inline;                        /* display horizontal */
        padding-right: 2em;                     /* item spacing */
        text-transform: uppercase;      /* ALL CAPS */
}

.menu a {
        color: black;
        transition: .5s;        /* given a fade transition when going to hover */
}

.menu a:hover {
        color: green;   /* given green color on hover */
}
```

That was pretty easy. We now have a better looking horizontal menu that looks like this:

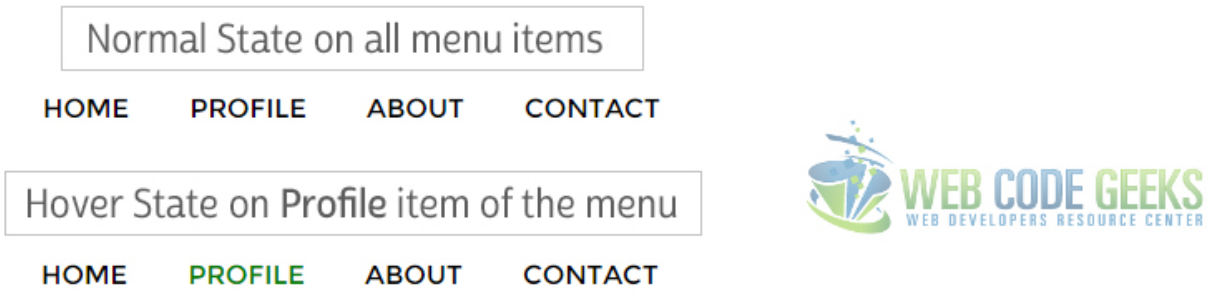

Figure 10.2: Basic Horizontal Menu

10.3 Advanced and Professional Menus

Now we will have a look at some more professional menus that have extra elements added on css.

10.3.1 Example 1

```
.menu a {
        color: #f2f2f2;
        transition: .5s;
}

.menu a:hover {
        color: white;
}

.menu {
        width: 28em;
        height: 3em;
        border-radius: 0.4em;
        padding: 0.5em;
        background-color: #56bce7;
}
```

Now this menu has a background and would look like this:

Figure 10.3: A Horizontal Menu with a Background

10.3.2 Example 2

Lets see another great styled menu which now has a more enhanced look.

HTML:

```
<head>
   <meta charset='utf-8'>
   <link rel="stylesheet" href="styles.css">
   <script src="http://code.jquery.com/jquery-latest.min.js" type="text/javascript"></ ↩
       script>
   <script src="script.js"></script>
   <title>CSS MenuMaker</title>
</head>
<body>


<ul>
   <li class='active'><a href='#'>Home</a></li>
   <li><a href='#'>Products</a></li>
   <li><a href='#'>Company</a></li>
   <li><a href='#'>Contact</a></li>
</ul>
```

CSS:

```
@import url(http://fonts.googleapis.com/css?family=Raleway);
#cssmenu,
#cssmenu ul,
#cssmenu ul li,
#cssmenu ul li a {
  margin: 0;
  padding: 0;
  border: 0;
  list-style: none;
  line-height: 1;
  display: block;
  position: relative;
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}
```

```
#cssmenu:after,
#cssmenu > ul:after {
  content: ".";
  display: block;
  clear: both;
  visibility: hidden;
  line-height: 0;
  height: 0;
}

#cssmenu {
  width: auto;
  border-bottom: 3px solid #47c9af;
  font-family: Raleway, sans-serif;
  line-height: 1;
}

#cssmenu ul {
  background: #ffffff;
}

#cssmenu > ul > li {
  float: left;
}
#cssmenu.align-center > ul {
  font-size: 0;
  text-align: center;
}
#cssmenu.align-center > ul > li {
  display: inline-block;
  float: none;
}
#cssmenu.align-right > ul > li {
  float: right;
}
#cssmenu.align-right > ul > li > a {
  margin-right: 0;
  margin-left: -4px;
}

#cssmenu > ul > li > a {
  z-index: 2;
  padding: 18px 25px 12px 25px;
  font-size: 15px;
  font-weight: 400;
  text-decoration: none;
  color: #444444;
  -webkit-transition: all .2s ease;
  -moz-transition: all .2s ease;
  -ms-transition: all .2s ease;
  -o-transition: all .2s ease;
  transition: all .2s ease;
  margin-right: -4px;
}

#cssmenu > ul > li.active > a,
#cssmenu > ul > li:hover > a,
#cssmenu > ul > li > a:hover {
  color: #ffffff;
}

#cssmenu > ul > li > a:after {
```

```
  position: absolute;
  left: 0;
  bottom: 0;
  right: 0;
  z-index: -1;
  width: 100%;
  height: 120%;
  border-top-left-radius: 8px;
  border-top-right-radius: 8px;
  content: "";
  -webkit-transition: all .2s ease;
  -o-transition: all .2s ease;
  transition: all .2s ease;
  -webkit-transform: perspective(5px) rotateX(2deg);
  -webkit-transform-origin: bottom;
  -moz-transform: perspective(5px) rotateX(2deg);
  -moz-transform-origin: bottom;
  transform: perspective(5px) rotateX(2deg);
  transform-origin: bottom;
}

#cssmenu > ul > li.active > a:after,
#cssmenu > ul > li:hover > a:after,
#cssmenu > ul > li > a:hover:after {
  background: #47c9af;
}
```

This would look like this:

Figure 10.4: 2nd Example of a Great Design and Professional Menu

You can continue to create or use templates of CSS & jQuery menus according to your needs.

One page that I recommend for you to browse is this.

10.4 Conclusion

To conclude, we can say that it is pretty easy to create basic menus with html and style with css. However, if you want professional and well designed menus consider browsing for templates or using frameworks which make this easier.

Note that to make interactive menus with submenus and dropdowns, you will probably need jQuery or Javascript in general to achieve animations, dropdowns, left and right menu expansion and so on and so forth.

10.5 Download

Download You can download the full source code of this example here: **CSS3 Horizontal Menu**

Chapter 11

CSS Rotate Image

In this example, we will consider image rotation using css.

CSS offers a specific property called `transform` to do this.

This property can take a lot of attributes, but we will only consider rotation.

Image rotation will be needed on certain occasions when you want to create your personalized view of image positioning and have a fancier look.

All browsers support the transform property, with their own prefixes: `-ms-transform` for IE, `-webkit-transform` for Chrome, Opera, Safari etc.

Lets first look at the basics and then something more on it.

11.1 Basic Setup & Application

First, go ahead and create a blank html document and then add the basic syntax inside like so:

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS3 Image Rotate Example</title>
</head>
<body>
<!-- STYLE  SECTION -->
<style type="text/css">

</style>

<!-- HTML SECTION  -->

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

Now add an `img` tag in the html section with an image inside like this:

```
<!-- HTML SECTION  -->

<img src="image.jpg">
```

To apply rotation, follow this syntax: `transform:rotate(10deg);` where:

1. **transform** is the attribute given to the img

2. **rotate** is the value of the attribute
3. **(10deg)** specifies at what scale should the value be applied.

With this said, the basic image rotation application would be:

```
<!-- STYLE  SECTION -->

<style type="text/css">

img {      transform: rotate(10deg);        }

</style>
```

Now you should see a rotated image on the browser, by 10 degrees:

Figure 11.1: Basic Image Rotation Application

11.2 Cases and Examples

In this section, we will go through some cases where you can use and how the image rotation.

In the html area, I will add 5 images so that we can create a collage like view of the images.

```
<!-- HTML SECTION  -->

<img class="img1" src="image.jpg">
<img class="img2" src="image1.jpg">
<img class="img3" src="image2.jpg">
<img class="img4" src="image3.jpg">
<img class="img5" src="image4.jpg">
```

Lets wrap images inside a border and a shadow and give odd numbers a -10deg rotation and even numbers 10deg:

1. `box-shadow` - each image is going to have a light gray shadow surrounding it.
2. `border` - borders are going to be placed around images in a gray color.
3. `transform:rotate` - first image will have a -10deg rotation, second a 10deg rotation and so on.

Coding the css part would be easy:

```
<!-- STYLE  SECTION -->

<style type="text/css">

.img1 {
        transform: rotate(-10deg);
        border: 0.8em solid #f4f4f4;
        box-shadow: 0.5em 0.5em 1em #ccc;
}


.img2 {
        transform: rotate(10deg);
        border: 0.8em solid #f4f4f4;
        box-shadow: 0.5em 0.5em 1em #ccc;
}

.img3 {
        transform: rotate(-10deg);
        border: 0.8em solid #f4f4f4;
        box-shadow: 0.5em 0.5em 1em #ccc;
}

.img4 {
        transform: rotate(10deg);
        border: 0.8em solid #f4f4f4;
        box-shadow: 0.5em 0.5em 1em #ccc;
}

.img5 {
        transform: rotate(-10deg);
        border: 0.8em solid #f4f4f4;
        box-shadow: 0.5em 0.5em 1em #ccc;
}

</style>
```

The view of this nice photo collage in the browser would be:

Figure 11.2: Photo Collage created using CSS Rotation

You choose positive values to rotate the images in the clockwise direction, and minus values to rotate counterclockwise.

Another example of the usage of image rotation would be a vertical gallery with inline descriptions. In this case, images are wrapped inside divs to display them in block view. Look at the html below:

```
<!-- HTML SECTION  -->

<img class="img1" src="image.jpg">
<img class="img2" src="image1.jpg">
<img class="img3" src="image2.jpg">
<img class="img4" src="image3.jpg">
<img class="img5" src="image4.jpg">
```

Now the CSS will have very slight changes (in this case, only the degree has changed and some size elements):

```
<!-- STYLE  SECTION -->

<style type="text/css">

body {
        font-family: "Montserrat";
}

.img1 {
        transform: rotate(7deg);
        border: 0.5em solid #f4f4f4;
        box-shadow: 0.5em 0.5em 0.5em #ccc;
}


.img2 {
        transform: rotate(-8deg);
        border: 0.5em solid #f4f4f4;
        box-shadow: 0.5em 0.5em 0.5em #ccc;
```

```
}

.img3 {
        transform: rotate(7deg);
        border: 0.5em solid #f4f4f4;
        box-shadow: 0.5em 0.5em 0.5em #ccc;
}

.img4 {
        transform: rotate(-8deg);
        border: 0.5em solid #f4f4f4;
        box-shadow: 0.5em 0.5em 0.5em #ccc;
}

.img5 {
        transform: rotate(7deg);
        border: 0.5em solid #f4f4f4;
        box-shadow: 0.5em 0.5em 0.5em #ccc;
}

</style>
```

So you can have a vertical view of these rotated images and place your description inline:

Figure 11.3: A Vertical Photo Gallery with Inline Description

Well, that is a clue of what you can do using the image rotation in css.

11.3 Conclusion

It seems like a small part of css, and it is somehow, but the transform property with rotate(xdeg) value applied can fundamentally change the way you think of images inside a webpage and rethink ways you can organise them.

It is very easy to use and create beautiful gallery-like sets of images placed on various degrees.

A 180 deg rotation would mean an upside down view while a 360 deg would mean no changes in the view, a full rotation.

11.4 Download

Download You can download the full source code of this example here: **CSS3 Image Rotate**

Chapter 12

CSS Hover Effects

In this example, we will go through performing hover effects on CSS.

You see hover effects on almost every webpage nowadays, it seems to be a good way to make elements eye-catching and enhance usability.

In a more functional note, it lets you know that (in most cases) something is clickable, like a link, an image or anything else.

The application is generally easy, and it is compatible with all modern browsers like Chrome, Firefox, Safari, Opera etc.

Lets first see how we can apply basic hover effects and then expand to more advanced usage with cases and examples.

12.1 Basic Setup & Application

First, go ahead and create a blank html document and then add the basic syntax inside like so:

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS3 Hover Effects Example</title>
</head>
<body>
<!-- STYLE  SECTION -->
<style type="text/css">

</style>

<!-- HTML SECTION  -->

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

To apply hover effects, you add `:hover` after the tag or class of html and then open the brackets:

-HTML (create an element, i.e a button, so that we can apply hover)

```
<button>This is me</button>
```

-CSS (respect the syntax `class/tag:hover { attributtes }` to apply hover effect)

```
<style type="text/css">
button:hover {
        background-color: red;
        color: white;
```

```
}
</style>
```

We expect this button to change background color to red and text color to white when the mouse is over it:

Figure 12.1: Hover on Button - Basic Application

So that is the most basic application of hover in a button, but there are more examples and elements that you can apply it.

12.2 Cases and Examples

Below, we will take elements of html, style them and give different hover effects to show examples.

12.2.1 Button Hover Effects

There are three most important aspects you should remember when stlying buttons with hovers:

1. Choose a light bg color on normal state and a deeper bg color on hover state or vice versa, that will look great.
2. Be careful to notice how the text inside the button looks like while on hover. If not right, change its color.
3. Always add a transition property with a value of .5s to make the hover effect animate (fade) beautifully.

Below, I have added three buttons on the HTML section, those will have the same styling but different hovers.

```
<!-- HTML SECTION  -->
<button class="btn1">button 1</button>
<button class="btn2">button 2</button>
<button class="btn3">button 3</button>
```

Now notice that I have given same styling by selecting the `button` tag and different hovers using the respective classes:

```
<!-- STYLE  SECTION -->

<style type="text/css">
button {
        width: 8em;
        height: 3em;
        border: none;
        border-radius: 0.5em;
        background-color: #919191;
        font-family: "Montserrat";
        text-transform: uppercase;
        color: white;
        margin-right: 2em;
        transition: .5s;
```

```
}

.btn1:hover {
        background-color:#8a3f3f;       /* deep red color */
        text-decoration: underline;
}

.btn2:hover {
        background-color: transparent; /* no background color */
        text-transform: lowercase;
        border: 0.2em solid #ccc;       /* added border color */
        color: black;
}

.btn3:hover {
        background-color:#3f708a;               /* deep blue color */
        box-shadow: .1em .1em 1em #ccc; /* added box shadow */
}
</style>
```

See how the buttons look in normal and hover state in the browser:

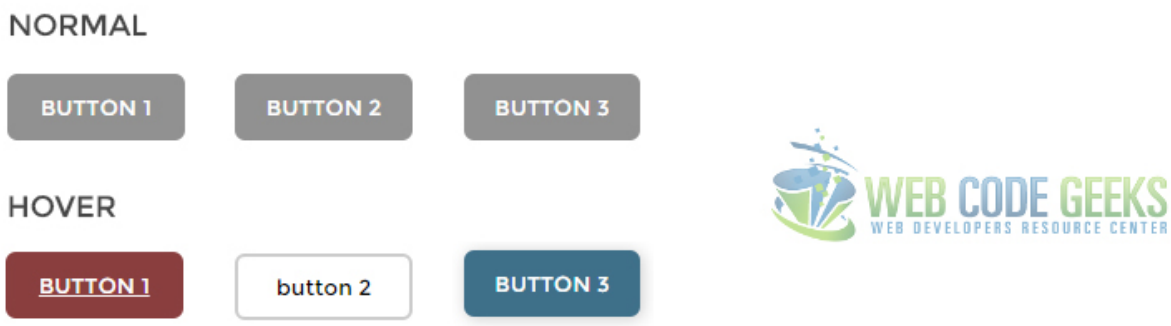

Figure 12.2: Buttons Hover Effects Examples

12.2.2 Other Elements Hovers

Lets take for example a simple html menu, and give the menu items hover effects:

```
    <a href="#1">Home</a>
    <a href="#2" class="current">Profile</a>
    <a href="#3">About</a>
    <a href="#4">Contact</a>
    <a href="#5">Sth Else Here</a>
```

Notice how you can add image effects using hover to the menu, in this case a gradient image:

```
div.menu2
{
    text-align:center;
    background-image: url(bg1.gif);
    border:1px solid black;
    font-size:0;
}
```

```
div.menu2 a
{
    display: inline-block;
    padding: 0 20px;
    background-image: url(bg.gif);
    color:White;
    text-decoration:none;
    font: bold 12px Arial;
    line-height: 32px;
}

div.menu2 a:hover, div.menu2 a.current
{
    background-position:0 -60px;
}
```

The menu would look like this:

Figure 12.3: Menu Image Hovers Example

12.2.3 Box Hover Animation and Added Text

THe following example is something advanced brought in a simple way to you. It is a box that is going to be given attributes for the normal state and then it is going to expand and show extra lines of text when on hover state.

```
<span class="element1">Hi Im not here at first, but only on hover.</span> I'm here for too ↩
    long now.
```

Applying some simple logics in css that would look like this:

```
.box {
        margin-top: 5em;
        width: 15em;
        height: 8em;
        font-size: 1.2em;
        font-family: "Montserrat";
        color: white;
        background-color: #464646;
        transition: width 2s;
}

span {
        display: none;  /* span should not be visible at first */
}
```

```
.box:hover{
        width: 25em;     /* box expanding on hover */
}

.box:hover .element1 {
        display: block;          /* span showing on hover */
}
```

The result, like you may have been wondering, would be:

Figure 12.4: Advanced Hover - Animation and Element Adding

It all comes beautifully animated and adjusts the text according to the new width.

12.3 Conclusion

To conclude, we can say that hovering elements should be used to have your website better designed.

It comes to usage in an easy form, only by adding the `:hover` in css, but when considering more advanced stuff like animation or adding elements after mouse over, it can get complicated.

However, keep in mind to use these hovering effects using only css and not javascript because it is the lightest way not to overload your website with js.

12.4 Download

Download You can download the full source code of this example here: **CSS3 Hover Effects**