CSS 3 – PART 1

CSS stands for Cascading Style Sheets. CSS is a standard style sheet language used for describing the presentation (i.e. the layout and formatting) of the web pages.

Prior to CSS, nearly all of the presentational attributes of HTML documents were contained within the HTML markup (specifically inside the HTML tags); all the font colors, background styles, element alignments, borders and sizes had to be explicitly described within the HTML.

As a result, development of the large websites became a long and expensive process, since the style information were repeatedly added to every single page of the website.

To solve this problem CSS was introduced in 1996 by the World Wide Web Consortium (W3C), which also maintains its standard. CSS was designed to enable the separation of presentation and content. Now web designers can move the formatting information of the web pages to a separate style sheet which results in considerably simpler HTML markup, and better maintainability.

CSS3 is the latest version of the CSS specification. CSS3 adds several new styling features and improvements to enhance the web presentation capabilities.

Advantages of Using CSS

The biggest advantage of CSS is that it allows the separation of style and layout from the content of the document. Here are some more advantages, why one should start using CSS?

- **CSS Save Lots of Time** — CSS gives lots of flexibility to set the style properties of an element. You can write CSS once; and then the same code can be applied to the groups of HTML elements, and can also be reused in multiple HTML pages.
- **Easy Maintenance** — CSS provides an easy means to update the formatting of the documents, and to maintain the consistency across multiple documents. Because the content of the entire set of web pages can be easily controlled using one or more style sheets.
- **Pages Load Faster** — CSS enables multiple pages to share the formatting information, which reduces complexity and repetition in the structural

contents of the documents. It significantly reduces the file transfer size, which results in a faster page loading.

- **Superior Styles to HTML** — CSS has much wider presentation capabilities than HTML and provide much better control over the layout of your web pages. So you can give far better look to your web pages in comparison to the HTML presentational elements and attributes.
- **Multiple Device Compatibility** — CSS also allows web pages to be optimized for more than one type of device or media. Using CSS the same HTML document can be presented in different viewing styles for different rendering devices such as desktop, cell phones, etc.

Including CSS in HTML Documents

CSS can either be attached as a separate document or embedded in the HTML document itself. There are three methods of including CSS in an HTML document:

- **Inline styles** — Using the `style` attribute in the HTML start tag.
- **Embedded styles** — Using the `<style>` element in the head section of a document.
- **External style sheets** — Using the `<link>` element, pointing to an external CSS file.

Inline Styles

Inline styles are used to apply the unique style rules to an element by putting the CSS rules directly into the start tag. It can be attached to an element using the `style` attribute.

The `style` attribute includes a series of CSS property and value pairs. Each `"property: value"` pair is separated by a semicolon (`;`), just as you would write into an embedded or external style sheets. But it needs to be all in one line i.e. no line break after the semicolon, as shown here:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Example of CSS Inline Styles</title>
  </head>
  <body>
    <h1 style="color: red; font-size: 30px">This is a heading</h1>
    <p style="color: green; font-size: 22px">This is a paragraph.</p>
    <div style="color: blue; font-size: 14px">This is some text content.</div>
  </body>
</html>
```

Using the inline styles are generally considered as a bad practice. As style rules are embedded directly inside the HTML tag, it causes the presentation to become mixed with the content of the document; which makes the code hard to maintain and negates the purpose of using CSS.

Embedded Style Sheets

Embedded or internal style sheets only affect the document they are embedded in.

Embedded style sheets are defined in the `<head>` section of an HTML document using the `<style>` element. You can define any number of `<style>` elements in an HTML document but they must appear between the `<head>` and `</head>` tags. Let's take a look at an example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Example of CSS Embedded Style Sheet</title>
    <style>
      body {
        background-color: YellowGreen;
      }
      p {
        color: #fff;
      }
    </style>
  </head>
  <body>
```

```
    <h1>This is a heading</h1>
    <p>This is a paragraph of text.</p>
  </body>
</html>
```

Tip: The `type` attribute of the `<style>` and `<link>` tag (i.e. `type="text/css"`) defines the language of the style sheet. This attribute is purely informative. You can omit this since CSS is the standard and default style sheet language in HTML5.

External Style Sheets

An external style sheet is ideal when the style is applied to many pages of the website.

An external style sheet holds all the style rules in a separate document that you can link from any HTML file on your site. External style sheets are the most flexible because with an external style sheet, you can change the look of an entire website by changing just one file.

You can attach external style sheets in two ways — *linking* and *importing*.

Linking External Style Sheets

Before linking, we need to create a style sheet first. Let's open your favorite code editor and create a new file. Now type the following CSS code inside this file and save it as "style.css".

```
body {
  background: lightyellow;
  font: 18px Arial, sans-serif;
}
h1 {
  color: orange;
}
```

An external style sheet can be linked to an HTML document using the `<link>` tag. The `<link>` tag goes inside the `<head>` section, as you can see in the following example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>My HTML Document</title>
    <link rel="stylesheet" href="css/style.css" />
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph of text.</p>
  </body>
</html>
```

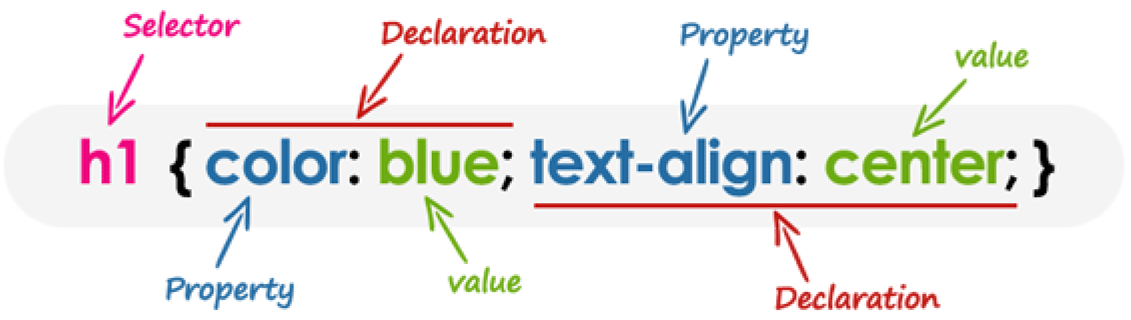
Tip: Among all the three methods, using external style sheet is the best method for defining and applying styles to the HTML documents. As you can clearly see with external style sheets, the affected HTML file require minimal changes in the markup.

CSS Syntax

Understanding CSS Syntax

A CSS stylesheet consists of a set of rules that are interpreted by the web browser and then applied to the corresponding elements such as paragraphs, headings, etc. in the document.

A CSS rule have two main parts, a selector and one or more declarations:



The selector specifies which element or elements in the HTML page the CSS rule applies to.

Whereas, the declarations within the block determines how the elements are formatted on a webpage. Each declaration consists of a property and a value separated by a colon (`:`) and ending with a semicolon (`;`), and the declaration groups are surrounded by curly braces `{}`.

The property is the style attribute you want to change; they could be font, color, background, etc. Each property has a value, for example color property can have value either `blue` or `#0000FF` etc.

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

To make the CSS more readable, you can put one declaration on each line, like this:

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

In the example above `h1` is a selector, `color` and `text-align` are the CSS properties, and the given `blue` and `center` are the corresponding values of these properties.

Note: A CSS declaration always ends with a semicolon "`;`", and the declaration groups are always surrounded by the curly brackets "`{}`".

Writing Comments in CSS

Comments are usually added with the purpose of making the source code easier to understand. It may help other developer (or you in the future when you edit the source code) to understand what you were trying to do with the CSS. Comments are significant to programmers but ignored by browsers.A CSS comment begins with `/*`, and ends with `*/`, as shown in the example below:

```
/* This is a CSS comment */
h1 {
  color: blue;
  text-align: center;
}
/* This is a multi-line CSS comment
that spans across more than one line */
p {
  font-size: 18px;
  text-transform: uppercase;
}
```

CSS Selectors

What is Selector?

A CSS selector is a pattern to match the elements on a web page. The style rules associated with that selector will be applied to the elements that match the selector pattern.

Selectors are one of the most important aspects of CSS as they allow you to target specific elements on your web page in various ways so that they can be styled.

Several types of selectors are available in CSS, let's take a closer look at them:

Universal Selector

The universal selector, denoted by an asterisk (`*`), matches every single element on the page.

The universal selector may be omitted if other conditions exist on the element. This selector is often used to remove the default margins and paddings from the elements for quick testing purpose.

Let's try out the following example to understand how it basically works:

```
* {
    margin: 0;
    padding: 0;
}
```

Note: It is recommended *not to use* the universal selector (`*`) too often in a production environment, since this selector matches every element on a web page that puts too much of unnecessary pressure on the browsers. Use element type or class selector instead.

Element Type Selectors

An element type selector matches all instance of the element in the document with the corresponding element type name. Let's try out an example to see how it actually works:

```
p {
    color: blue;
}
```

The style rules inside the `p` selector will be applied on every `<p>` element (or paragraph) in the document and color it blue, regardless of their position in the document tree.

Id Selectors

The id selector is used to define style rules for a *single* or *unique* element.

The id selector is defined with a hash sign (`#`) immediately followed by the id value.

```
#error {
    color: red;
}
```

This style rule renders the text of an element in red, whose `id` attribute is set to `error`.

Note: The value of an id attribute must be unique within a given document — meaning no two elements in your HTML document can share the same id value.

Class Selectors

The class selectors can be used to select any HTML element that has a `class` attribute. All the elements having that class will be formatted according to the defined rule.

The class selector is defined with a period sign (`.`) immediately followed by the class value.

```
.blue {
    color: blue;
}
```

The above style rules renders the text in blue of every element in the document that has `class` attribute set to `blue`. You can make it a bit more particular. For example:

```
p.blue {
    color: blue;
}
```

The style rule inside the selector `p.blue` renders the text in blue of only those `<p>` elements that has `class` attribute set to `blue`, and has no effect on other paragraphs.

Descendant Selectors

You can use these selectors when you need to select an element that is the descendant of another element, for example, if you want to target only those anchors that are contained within an unordered list, rather than targeting all anchor elements. Let's see how it works:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Example of CSS Descendant Selectors</title>
    <style>
      h1 em {
        color: green;
      }
      ul.menu {
        padding: 0;
        list-style: none;
      }
      ul.menu li {
        display: inline;
      }
      ul.menu li a {
        margin: 10px;
        text-decoration: none;
      }
    </style>
  </head>
  <body>
    <h1>This is a <em>heading</em></h1>
    <ul class="menu">
      <li><a href="#">Home</a></li>
      <li><a href="#">About</a></li>
      <li><a href="#">Services</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </body>
</html>
```

The style rules inside the selector `ul.menu li a` applied to only those `<a>` elements that contained inside an `<ul>` element having the class `.menu`, and has no effect on other links inside the document.

Similarly, the style rules inside the `h1 em` selector will be applied to only those `<em>` elements that contained inside the `<h1>` element and has not effect on other `<em>` elements.

Grouping Selectors

Often several selectors in a style sheet share the same style rules declarations. You can group them into a comma-separated list to minimize the code in your style sheet. It also prevents you from repeating the same style rules over and over again. Let's take a look:

```
h1 {
    font-size: 36px;
    font-weight: normal;
}
h2 {
    font-size: 28px;
    font-weight: normal;
}
h3 {
    font-size: 22px;
    font-weight: normal;
}
```

As you can see in the above example, the same style rule `font-weight: normal;` is shared by the selectors `h1`, `h2` and `h3`, so it can be grouped in a comma-separated list, like this:

```
h1, h2, h3 {
    font-weight: normal;
}
h1 {
    font-size: 36px;
}
h2 {
    font-size: 28px;
}
h3 {
    font-size: 22px;
}
```

CSS Color

Setting Color Property

The `color` property defines the text color (foreground color in general) of an element.

For instance, the `color` property specified in the `body` selector defines the default text color for the whole page. Let's try out the following example to see how it works:

```
body {
    color: #ff5722;
}
```

Note: The `color` property normally inherits the color value from their parent element, except the case of anchor elements. For example, if you specify `color` for the `body` element it will automatically be passed down to the headings, paragraphs, etc.

Defining Color Values

Colors in CSS most often specified in the following formats:

- a color keyword - like "red", "green", "blue", "transparent", etc.
- a HEX value - like "#ff0000", "#00ff00", etc.
- an RGB value - like "rgb(255, 0, 0)"

CSS3 has introduced several other color formats such as HSL, HSLA and RGBA that also support alpha transparency.

Color Keywords

CSS defines the few color keywords which lets you specify color values in an easy way.

These basic color keywords are: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. The color names are case-insensitive.

```
h1 {
    color: red;
}
p {
    color: purple;
}
```

Modern web browsers however practically support many more color names than what are defined in the CSS standard, but to be on the safer side you should use hex color values instead.

HEX Color Values

Hex (short for Hexadecimal) is by far the most commonly used method of defining color on the web.

Hex represent colors using a six-digit code, preceded by a hash character, like `#rrggbb`, in which `rr`, `gg`, and `bb` represents the red, green and blue component of the color respectively.

The value of each component can vary from 00 (no color) and FF (full color) in hexadecimal notation, or 0 and 255 in decimal equivalent notation. Thus `#ffffff` represents white color and `#000000` represents black color. Let's take a look the following example:

```
h1 {
    color: #ffa500;
}
p {
    color: #00ff00;
}
```

RGB Color Values

Colors can be defined in the RGB model (Red, Green, and Blue) using the `rgb()` functional notation.

The `rgb()` function accepts three comma-separated values, which specify the amount of red, green, and blue component of the color. These values are commonly specified as integers between 0 to 255, where 0 represent *no color* and 255 represent *full or maximum color*.

The following example specifies the same color as in the previous example but in RGB notation.

```
h1 {
    color: rgb(255, 165, 0);
}
p {
    color: rgb(0, 255, 0);
}
```

Note: You can also specify RGB values inside the `rgb()` function in percentage, where 100% represents full color, and 0% (*not simply 0*) represents no color. For example, you can specify the red color either as `rgb(255, 0, 0)` or `rgb(100%, 0%, 0%)`.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Setting Foreground Color for Elements</title>
    <style>
      p.one {
        color: #0000ff;
        border: 2px solid;
      }
      p.two {
        color: #00ff00;
        outline: 2px solid;
      }
    </style>
  </head>
  <body>
    <p class="one">
      The border color of this paragraph is same as the element's text color.
    </p>
    <p class="two">
```

```
      The outline color of this paragraph is same as the element's text color.
    </p>
  </body>
</html>
```

CSS Background

Setting Background Properties

Background plays an important role in the visual presentation of a web page.

CSS provide several properties for styling the background of an element, including coloring the background, placing images in the background and managing their positioning, etc.

The background properties are `background-color`, `background-image`, `background-repeat`, `background-attachment` and `background-position`.

In the following section we will discuss each of these properties in more detail.

Background Color

The `background-color` property is used to set the background color of an element.

The following example demonstrates how to set the background color of the whole page.

```
body {
    background-color: #f0e68c;
}
```

Color values in CSS are most often specified in the following formats:

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

Background Image

The `background-image` property set an image as a background of an HTML element.

Let's check out the following example which sets the background image for the whole page.

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

Note: When applying the background image to an element, make sure that the image you choose does not affect the readability of the element's text content.

Tip: By default browser repeats or tiles the background image both horizontally and vertically to fill the entire area of an element. You can control this with `background-repeat` property.

Background Repeat

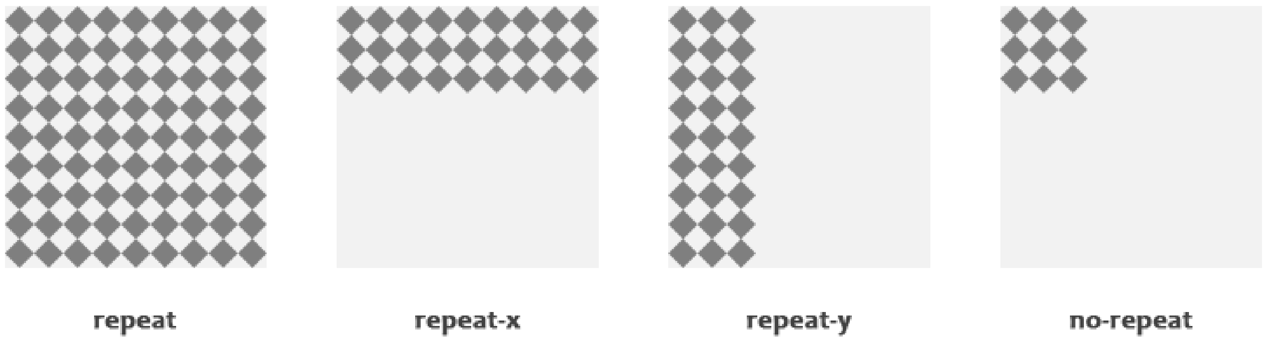
The `background-repeat` property allows you to control how a background image is repeated or tiled in the background of an element. You can set a background image to repeat vertically (y-axis), horizontally (x-axis), in both directions, or in neither direction. Let's try out the following example which demonstrates how to set the gradient background for a web page by repeating the sliced image horizontally along the x-axis.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Setting Horizontal background-repeat in CSS</title>
<style>
    body {
        background-image: url("/examples/images/gradient.png");
        background-repeat: repeat-x;
    }
</style>
</head>
<body>
    <h1>Background Repeat Demo</h1>
    <p>some text</p>
</body>
</html>
```

Similarly, you can use the value `repeat-y` to repeat the background image vertically along the y-axis, or the value `no-repeat` to prevent the repetition altogether.

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

Let's take a look at the following illustration to understand how this property actually works.

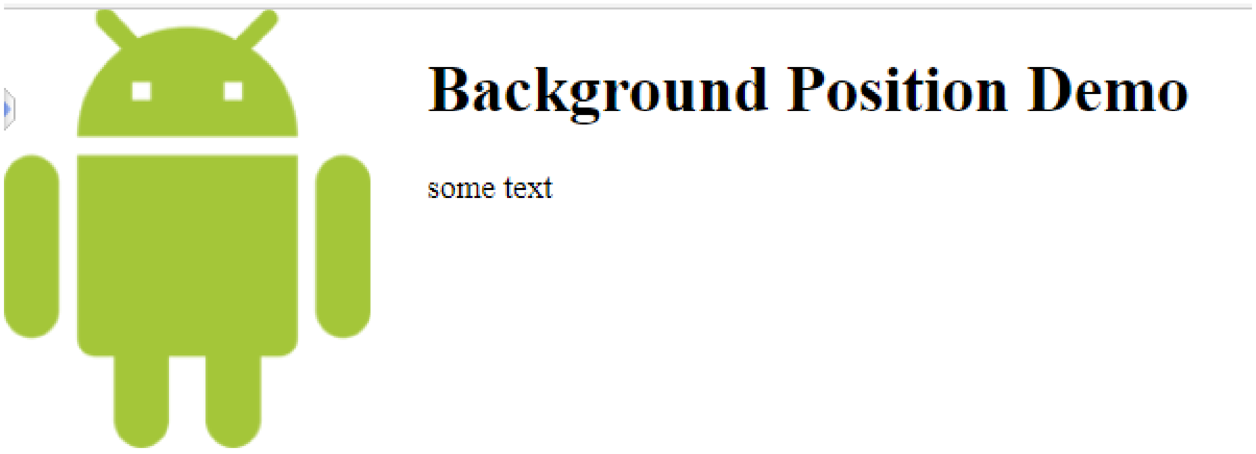


Background Position

The `background-position` property is used to control the position of the background image.

If no background position has been specified, the background image is placed at the default top-left position of the element i.e. at `(0,0)`, let's try out the following example:

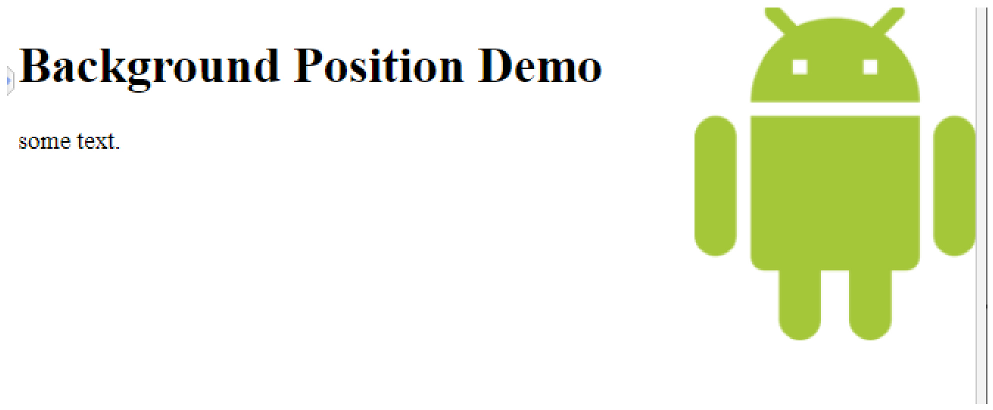
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Default background-position of Images in CSS</title>
<style>
    body {
        background-image: url("/examples/images/robot.png");
        background-repeat: no-repeat;
    }
    h1, p {
        margin-left: 200px;
    }
</style>
</head>
<body>
    <h1>Background Position Demo</h1>
    <p>some text </p>
</body>
</html>
```



Background Position Demo

some text

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Setting Custom background-position of Images in CSS</title>
<style>
    body {
        background-image: url("/examples/images/robot.png");
        background-repeat: no-repeat;
        background-position: 100% top;
    }
    h1, p {
        margin-right: 200px;
    }
</style>
</head>
<body>
    <h1>Background Position Demo</h1>
    <p>some text.</p>
</body>
</html>
```



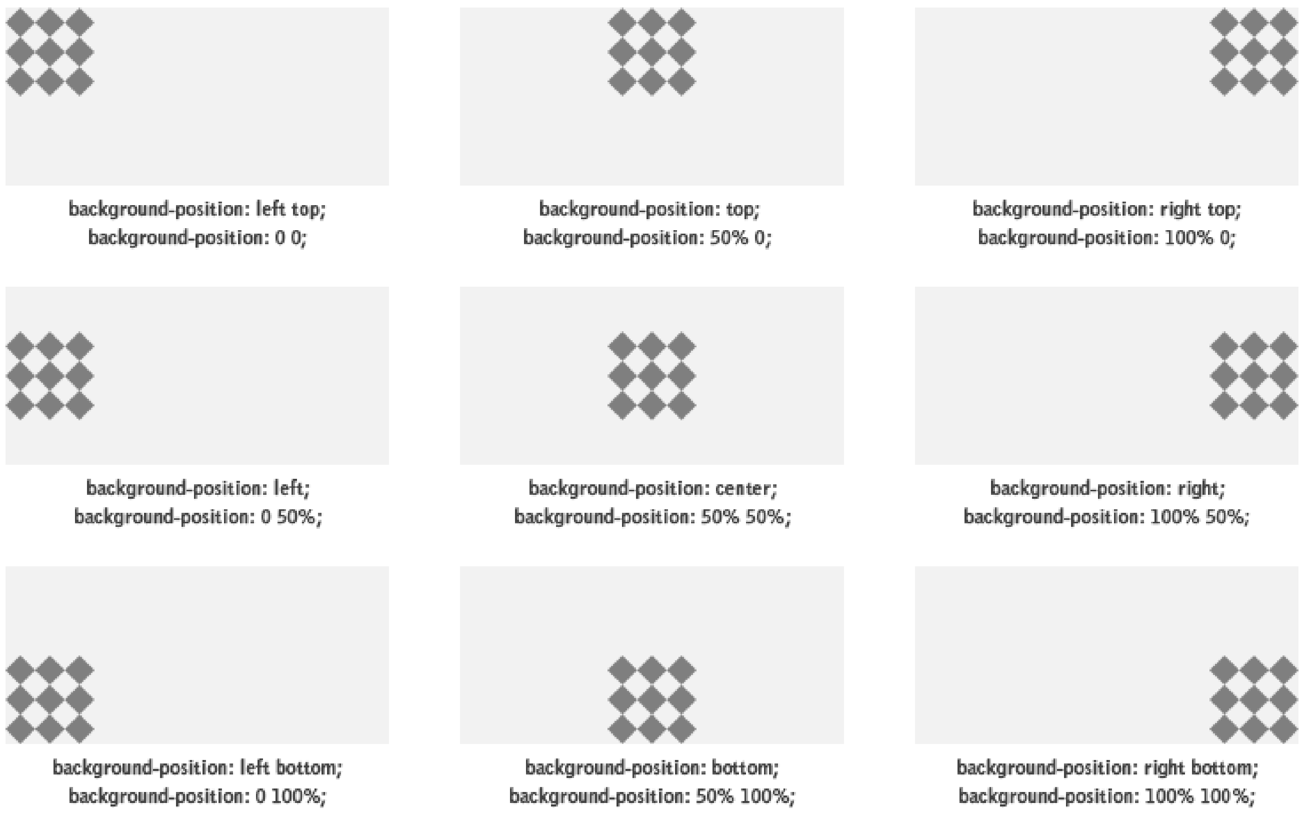
Background Position Demo

some text.

Note: If two values are specified for the `background-position` property, the first value represents the horizontal position, and the second represents the vertical position. If only one value is specified, the second value is assumed to be center.

Besides keywords, you can also use percentage or length values, such as `px` or `em` for this property.

Let's take a look at the following illustration to understand how this property actually works.



```
background-position: left top;
background-position: 0 0;

background-position: top;
background-position: 50% 0;

background-position: right top;
background-position: 100% 0;

background-position: left;
background-position: 0 50%;

background-position: center;
background-position: 50% 50%;

background-position: right;
background-position: 100% 50%;

background-position: left bottom;
background-position: 0 100%;

background-position: bottom;
background-position: 50% 100%;

background-position: right bottom;
background-position: 100% 100%;
```

Background Attachment

The `background-attachment` property determines whether the background image is fixed with regard to the viewport or scrolls along with the containing block.

Let's try out the following example to understand how it basically works:

```
body {
    background-image: url("images/bell.png");
    background-repeat: no-repeat;
    background-attachment: fixed;
}
```

The Background Shorthand Property

As you can see in the examples above, there are many properties to consider when dealing with the backgrounds. However, it is also possible to specify all these properties in one single property to shorten the code or avoid extra typing. This is called a shorthand property.

The `background` property is a shorthand property for setting all the individual background properties, i.e., `background-color`, `background-image`, `background-repeat`, `background-attachment` and the `background-position` property at once. Let's see how this works:

```
body {
    background-color: #f0e68c;
    background-image: url("images/smiley.png");
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: 250px 25px;
}
```

Using the shorthand notation, the above example can be written as:

```
body {
    background: #f0e68c url("images/smiley.png") no-repeat fixed 250px 25px;
}
```

When using the `background` shorthand property the order of the property values should be.

```
background: color image repeat attachment position;
```

CSS Fonts

Styling Fonts with CSS

Choosing the right font and style is very crucial for the readability of text on a page.

CSS provide several properties for styling the font of the text, including changing their face, controlling their size and boldness, managing variant, and so on.

The font properties are: `font-family`, `font-style`, `font-weight`, `font-size`, and `font-variant`.

Font Family

The `font-family` property is used to specify the font to be used to render the text.

This property can hold several comma-separated font names as a *fallback* system, so that if the first font is not available on the user's system, browser tries to use the second one, and so on.

Hence, list the font that you want first, then any fonts that might fill in for the first if it is not available. You should end the list with a generic font family which are five — `serif`, `sans-serif`, `monospace`, `cursive` and `fantasy`. A typical font family declaration might look like this:

```
body {
    font-family: Arial, Helvetica, sans-serif;
}
```

Note: If the name of a font family contains more than one word, it must be placed inside quotation marks, like `"Times New Roman"`, `"Courier New"`, `"Segoe UI"`, etc.

The most common font families used in web design are *serif* and *sans-serif*, because they are more suitable for reading. While *monospace* fonts are commonly used to display code, because in this typeface each letter takes up the same space which looks like typewritten text.

Font Style

The `font-style` property is used to set the font face style for the text content of an element.

The font style can be `normal`, `italic` or `oblique`. The default value is `normal`.

Let's try out the following example to understand how it basically works:

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

Font Size

The `font-size` property is used to set the size of font for the text content of an element.

There are several ways to specify the font size values e.g. with keywords, percentage, pixels, ems, etc.

Setting Font Size with Pixels

Setting the font size in pixel values (e.g. 14px, 16px, etc.) is a good choice when you need the pixel accuracy. Pixel is an absolute unit of measurement which specifies a fixed length.

Let's try out the following example to understand how it basically works:

```
h1 {
    font-size: 24px;
}
p {
    font-size: 14px;
}
```

Defining the font sizes in pixel is not considered very accessible, because the user cannot change the font size from the browser settings. For instance, users with limited or low vision may wish to set the font size much larger than the size specified by you.

Therefore, you should avoid using the pixels values and use the values that are relative to the user's default font size instead if you want to create an inclusive design.

Tip: The text can also be resized in all browsers using the *zoom feature*. However, this feature resizes the entire page, not just the text. The W3C recommends using the em or percentage (%) values in order to create more robust and scalable layouts.

Setting Font Size with EM

The `em` unit refers to the font size of the parent element. When defining the `font-size` property, `1em` is equal to the size of the font that applies to the *parent of the element*.

So, if you set a `font-size` of 20px on the body element, then `1em = 20px` and `2em = 40px`.

However, if you haven't set the font size anywhere on the page, then it is the browser default, which is normally 16px. Therefore, by default `1em = 16px`, and `2em = 32px`.

Let's take a look at the following example to understand how it basically works:

```
h1 {
    font-size: 2em;    /* 32px/16px=2em */
}
p {
    font-size: 0.875em;    /* 14px/16px=0.875em */
}
```

Using the Combination of Percentage and EM

As you've observed in the above example the calculation of em values doesn't look straightforward. To simplify this, a popular technique is to set the `font-size` for the body element to `62.5%` (that is 62.5% of the default 16px), which equates to 10px, or 0.625em.

Now you can set the `font-size` for any elements using em units, with an easy-to-remember conversion, by dividing the `px` value by 10. This way `10px = 1em`, `12px = 1.2em`, `14px = 1.4em`, `16px = 1.6em`, and so on. Let's take a look at the following example:

```
body {
    font-size: 62.5%;    /* font-size 1em = 10px */
}
p {
    font-size: 1.4em;    /* 1.4em = 14px */
}
p span {
    font-size: 2em;    /* 2em = 28px */
}
```

Setting Font Size with Root EM

To make things even more simpler CSS3 has introduced `rem` unit (short for "root em") which is always relative to the font-size of the root element (`html`), regardless of where the element lies in the document (unlike `em` which is relative to parent element's font size).

This means that `1rem` is equivalent to the font size of the `html` element, which is `16px` by default in most browsers. Let's try out an example to understand how it actually works:

```
html {
    font-size: 62.5%;    /* font-size 1em = 10px */
}
p {
    font-size: 1.4rem;    /* 1.4rem = 14px */
}
p span {
    font-size: 2rem;    /* 2rem = 20px (not 28px) */
}
```

Setting Font Size with Keywords

CSS provide several keywords that you can use to define font sizes.

An absolute font size can be specified using one of the following keywords: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`. Whereas, a relative font size can be specified using the keywords: `smaller` or `larger`. Let's try out an example and see how it works:

```
body {
    font-size: large;
}
h1 {
    font-size: larger;
}
p {
    font-size: smaller;
}
```

Note: The keyword `medium` is equivalent to the browsers default font-size, which is normally 16px. Likewise, xx-small is the equivalent of 9 pixels, x-small is 10 pixels, small is 13 pixels, large is 18 pixels, x-large is 24 pixels, and xx-large is 32 pixels.

Tip: By setting a font size on the body element, you can set the relative font sizing everywhere else on the page, giving you the ability to easily scale the font size up or down accordingly.

Setting Font Size with Viewport Units

The font sizes can be specified using viewport units such as `vw` or `vh`.

Viewport units refer to a percentage of the browser's viewport dimensions, where 1vw = 1% of viewport width, and 1vh = 1% of viewport height. Hence, if the viewport is 1600px wide, 1vw is 16px.

Try out the following example by resizing the browser window and see how it works:

```
body {
    font-size: 1vw;
}
h1 {
    font-size: 3vw;
}
```

However, there is a problem with the viewport units. On small screens fonts become so small that they are hardly readable. To prevent this you can utilize CSS `calc()` function, like this:

```
html {
    font-size: calc(1em + 1vw);
}
h1 {
    font-size: 3rem;
}
```

In this example, even if the viewport width becomes 0, the font-size would be at least 1em or 16px.

Font Weight

The `font-weight` property specifies the weight or boldness of the font.

This property can take one of the following values: `normal`, `bold`, `bolder`, `lighter`, `100`, `200`, `300`, `400`, `500`, `600`, `700`, `800`, `900` and `inherit`.

- The numeric values `100`-`900` specify the font weights, where each number represents a weight greater than its predecessor. `400` is same as `normal` & `700` is same as `bold`.
- The `bolder` and `lighter` values are relative to the inherited font weight, while the other values such as `normal` and `bold` are absolute font weights.

Let's try out an example to understand how this property basically works:

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

CSS Text

Formatting Text with CSS

CSS provides several properties that allows you to define various text styles such as color, alignment, spacing, decoration, transformation, etc. very easily and effectively.

The commonly used text properties are: `text-align`, `text-decoration`, `text-transform`, `text-indent`, `line-height`, `letter-spacing`, `word-spacing`, and more. These properties give you precise control over the visual appearance of the *characters*, *words*, *spaces*, and so on.

Let's see how to set these text properties for an element in more detail.

Text Color

The color of the text is defined by the CSS `color` property.

The style rule in the following example will define the default text color for the page

```
h1 {
        color: #ff0000;
    }
    p {
        color: green;
    }
```

Text Alignment

The `text-align` property is used to set the horizontal alignment of the text.

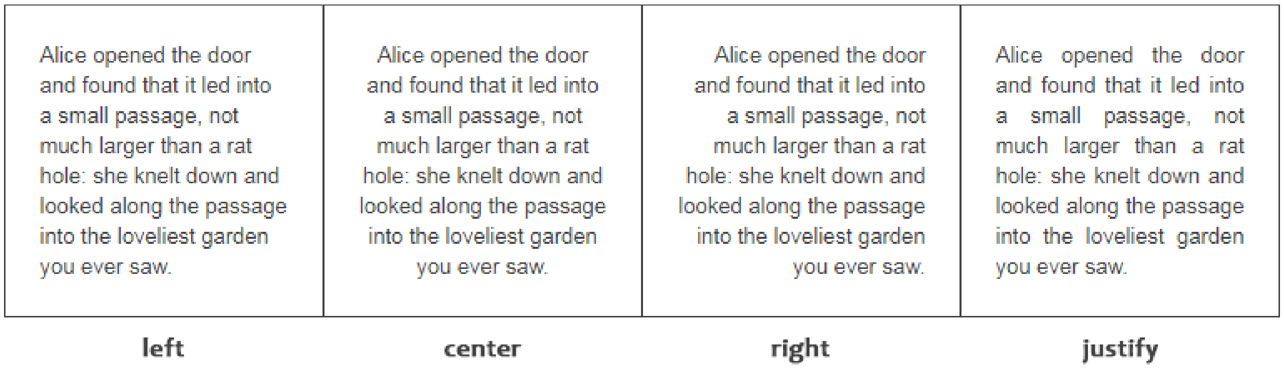
Text can be aligned in four ways: to the left, right, centre or justified (straight left and right margins).

Let's take a look at an example to understand how this property basically works.

```
h1 {
    text-align: center;
}
p {
    width: 300px;
    text-align: justify;
}
```

Note: When `text-align` is set to `justify`, each line is stretched so that every line has equal width (except the last line), and the left and right margins are straight. This alignment is generally used in publications such as magazines and newspapers.

Let's take a look at the following illustration to understand what these values actually mean.


Alice opened the door and found that it led into a small passage, not much larger than a rat hole: she knelt down and looked along the passage into the loveliest garden you ever saw.
left
center
right
justify

Text Decoration

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

This property typically accepts one of the following values: `underline`, `overline`, `line-through`, and `none`. You should avoid underline text that is not a link, as it might confuse the visitor.

Let's try out the following example to understand how it basically works:

```
h1 {
    text-decoration: overline;
}
h2 {
    text-decoration: line-through;
}
h3 {
    text-decoration: underline;
}
```

Text Transformation

The `text-transform` property is used to set the cases for a text.

Text are often written in mixed case. However, in certain situations you may want to display your text in entirely different case. Using this property you can change an element's text content into uppercase or lowercase letters, or capitalize the first letter of each word without modifying the original text.

Let's try out the following example to understand how it basically works:

```
h1 {
    text-transform: uppercase;
}
h2 {
    text-transform: capitalize;
}
h3 {
    text-transform: lowercase;
}
```

CSS Links

Styling Links with CSS

Links or hyperlinks are an essential part of a website. It allows visitors to navigate through the site. Therefore styling the links properly is an important aspect of building a user-friendly website.

See the tutorial on HTML links to learn more about links and how to create them.

A link has four different states — `link`, `visited`, `active` and `hover`. These four states of a link can be styled differently through using the following anchor pseudo-class selectors.

- **a:link** — define styles for normal or unvisited links.
- **a:visited** — define styles for links that the user has already visited.
- **a:hover** — define styles for a link when the user place the mouse pointer over it.
- **a:active** — define styles for links when they are being clicked.

You can specify any CSS property you'd like e.g. `color`, `font`, `background`, `border`, etc. to each of these selectors to customize the style of links, just like you do with the normal text.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Styling Different Link States using CSS</title>
<style>
```

```
    /* unvisited link */
    a:link {
        color: #ff0000;
        text-decoration: none;
        border-bottom: 1px solid;
    }
    /* visited link */
    a:visited {
        color: #ff00ff;
    }
    /* mouse over link */
    a:hover {
        color: #00ff00;
        border-bottom: none;
    }
    /* active link */
    a:active {
        color: #00ffff;
    }
</style>
</head>
<body>
    <p><a href="https://www.tutorialrepublic.com/" target="_top">Visit Tutorial
Republic</a></p>
</body>
</html>
```

The order in which you are setting the style for different states of links is important, because what defines last takes precedence over the style rules defined earlier.

Note: In general, the order of the pseudo classes should be the following — `:link`, `:visited`, `:hover`, `:active`, `:focus` in order for these to work properly.

Making Text Links Look Like Buttons

You can also make your ordinary text links look like button using CSS. To do this we need to utilize few more CSS properties such as `background-color`, `border`, `display`, `padding`, etc. You will learn about these properties in detail in upcoming chapters.

Let's check out the following example and see how it really works:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Customize a Link as Button using CSS</title>
<style>
    a:link, a:visited {
        color: white;
        background-color: #1ebba3;
        display: inline-block;
        padding: 10px 20px;
        border: 2px solid #099983;
        text-decoration: none;
        text-align: center;
        font: 14px Arial, sans-serif;
    }
    a:hover, a:active {
        background-color: #9c6ae1;
        border-color: #7443b6;
    }
</style>
</head>
<body>
    <p><a href="#">CSS Link Button</a></p>
</body>
</html>
```

CSS Lists

Types of HTML Lists

There are three different types of list in HTML:

- **Unordered lists** — A list of items, where every list items are marked with bullets.
- **Ordered lists** — A list of items, where each list items are marked with numbers.
- **Definition list** — A list of items, with a description of each item.

Styling Lists with CSS

CSS provides the several properties for styling and formatting the most commonly used unordered and ordered lists. These CSS list properties typically allow you to:

- Control the shape or appearance of the marker.
- Specify an image for the marker rather than a bullet point or number.
- Set the distance between a marker and the text in the list.
- Specify whether the marker would appear inside or outside of the box containing the list items.

Changing the Marker Type of Lists

By default, items in an ordered list are numbered with Arabic numerals (1, 2, 3, 5, and so on), whereas in an unordered list, items are marked with round bullets (•).

But, you can change this default list marker type to any other type such as roman numerals, latin letters, circle, square, and so on using the `list-style-type` property.

Let's try out the following example to understand how this property actually works:

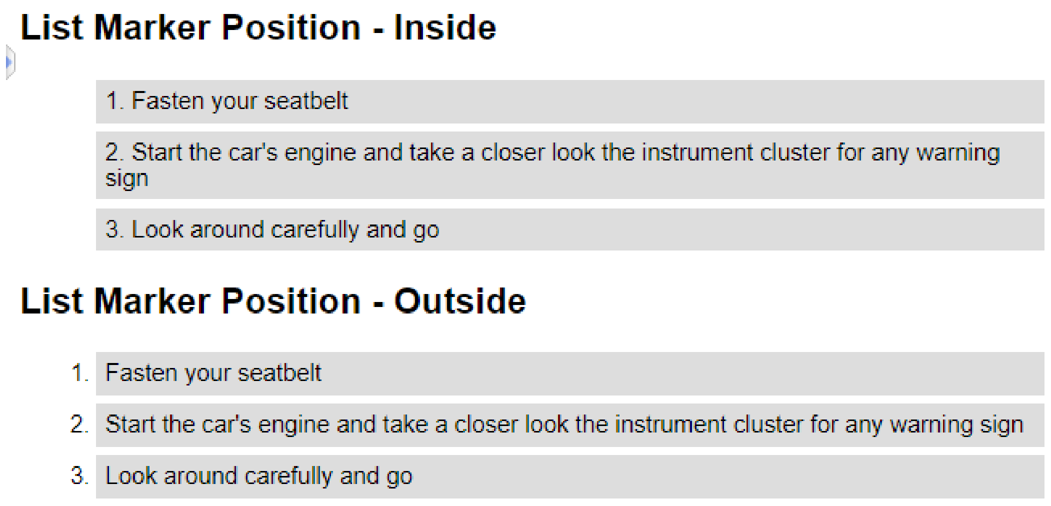
```css
ul {
    list-style-type: square;
}
ol {
    list-style-type: upper-roman;
}
```

Changing the Position of List Markers

By default, markers of each list items are positioned `outside` of their display boxes.

However, you can also position these markers or bullet points inside of the list item's display boxes using the `list-style-position` property along with the value `inside`. In this case the lines will wrap under the marker instead of being indented. Here's an example:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Setting the Position of List Marker</title>
<style>
    body{
        font-size: 14px;
        font-family: Arial,sans-serif;
    }
    ol li {
        background: #ddd;
        padding: 5px;
        margin: 5px;
    }
    ol.in li {
        list-style-position: inside;
    }
    ol.out li {
        list-style-position: outside;
    }
</style>
</head>
<body>
    <h2>List Marker Position - Inside</h2>
    <ol class="in">
        <li>Fasten your seatbelt</li>
        <li>Start the car's engine and take a closer look the instrument cluster
for any warning sign</li>
        <li>Look around carefully and go</li>
    </ol>
    <h2>List Marker Position - Outside</h2>
    <ol class="out">
        <li>Fasten your seatbelt</li>
        <li>Start the car's engine and take a closer look the instrument cluster
for any warning sign</li>
        <li>Look around carefully and go</li>
    </ol>
</body>
</html>
```



List Marker Position - Inside

1. Fasten your seatbelt
2. Start the car's engine and take a closer look the instrument cluster for any warning sign
3. Look around carefully and go

List Marker Position - Outside

1. Fasten your seatbelt
2. Start the car's engine and take a closer look the instrument cluster for any warning sign
3. Look around carefully and go

CSS Tables

Styling Tables with CSS

Tables are typically used to display tabular data, such as financial reports.

But when you create an HTML table without any styles or attributes, browsers display them without any border. With CSS you can greatly improve the appearance your tables.

CSS provides several properties that allow you to control the layout and presentation of the table elements. In the following section you will see how to use CSS to create elegant and consistent tables.

Adding Borders to Tables

The CSS `border` property is the best way to define the borders for the tables.

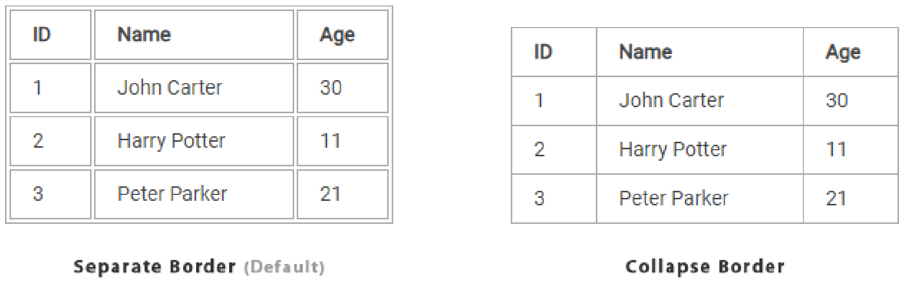
The following example will set a black border for the `<table>`, `<th>`, and `<td>` elements.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
```

```
<title>Example of Setting Table Borders</title>
<style>
    table, th, td {
        border: 1px solid black;
    }
</style>
</head>
<body>
    <table>
        <tr>
            <th>ID</th>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Email</th>
        </tr>
        <tr>
            <td>1</td>
            <td>John</td>
            <td>Carter</td>
            <td>johncarter@mail.com</td>
        </tr>
        <tr>
            <td>2</td>
            <td>Peter</td>
            <td>Parker</td>
            <td>peterparker@mail.com</td>
        </tr>
        <tr>
            <td>3</td>
            <td>John</td>
            <td>Rambo</td>
            <td>johnrambo@mail.com</td>
        </tr>
        <tr>
            <td>4</td>
            <td>Harry</td>
            <td>Potter</td>
            <td>harrypotter@mail.com</td>
        </tr>
    </table>
</body>
</html>
```

By default, browser draws a border around the table, as well as around all the cells, with some space in-between, which results in double border. To get rid of this double border problem you can simply collapse the adjoining table cell borders and create clean single line borders.

Let's take a look at the following illustration to understand how a border is applied to a table.



ID	Name	Age
1	John Carter	30
2	Harry Potter	11
3	Peter Parker	21

Separate Border (Default)

ID	Name	Age
1	John Carter	30
2	Harry Potter	11
3	Peter Parker	21

Collapse Border

Collapsing Table Borders

There are two distinct models for setting borders on table cells in CSS: *separate* and *collapse*.

In the separate border model, which is the default, each table cell has its own distinct borders, whereas in the collapsed border model, adjacent table cells share a common border. You can set the border model for an HTML table by using the CSS `border-collapse` property.

The following style rules will collapse the table cell borders and apply one pixel black border.

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

ID	First Name	Last Name	Email
1	John	Carter	johncarter@mail.com
2	Peter	Parker	peterparker@mail.com
3	John	Rambo	johnrambo@mail.com
4	Harry	Potter	harrypotter@mail.com

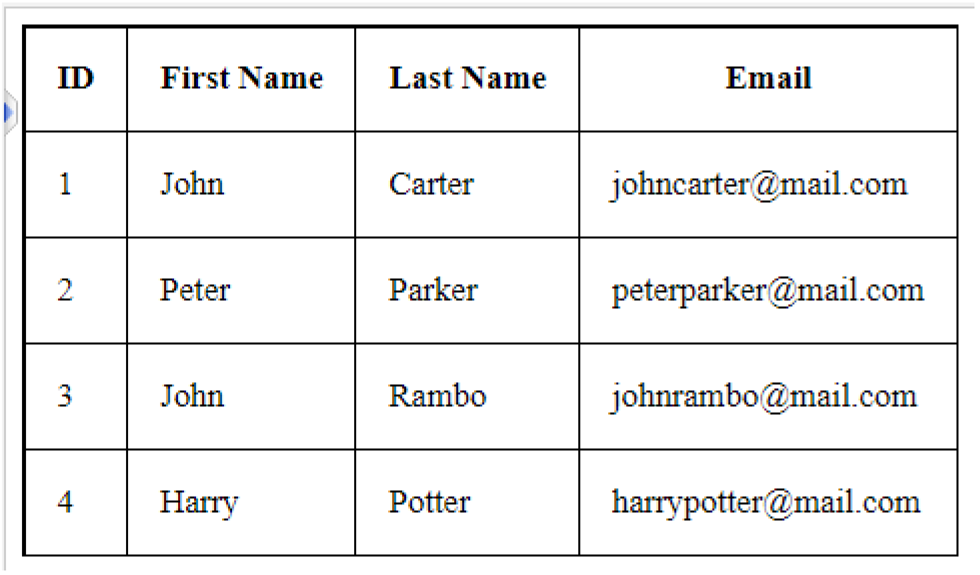
Note: You can also remove the space between the table cell borders through setting the value of CSS `border-spacing` property to 0. However, it only removes the space but do not merge the borders like when you set the `border-collapse` to `collapse`.

Adjusting Space inside Tables

By default, the browser creates the table cells just large enough to contain the data in the cells.

To add more space between the table cell contents and the cell borders, you can simply use the CSS `padding` property. Let's try out the following example and see how it works:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Example of Setting Table Cell Padding</title>
<style>
    table {
        border-collapse: collapse;
    }
    table, th, td {
        border: 1px solid black;
    }
    th, td {
        padding: 15px;
    }
</style>
</head>
<body>
    <table>
        <tr>
            <th>ID</th>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Email</th>
        </tr>
        <tr>
            <td>1</td>
            <td>John</td>
            <td>Carter</td>
            <td>johncarter@mail.com</td>
        </tr>
        <tr>
            <td>2</td>
            <td>Peter</td>
            <td>Parker</td>
            <td>peterparker@mail.com</td>
        </tr>
        <tr>
            <td>3</td>
            <td>John</td>
            <td>Rambo</td>
            <td>johnrambo@mail.com</td>
        </tr>
        <tr>
            <td>4</td>
            <td>Harry</td>
            <td>Potter</td>
            <td>harrypotter@mail.com</td>
        </tr>
    </table>
</body>
</html>
```

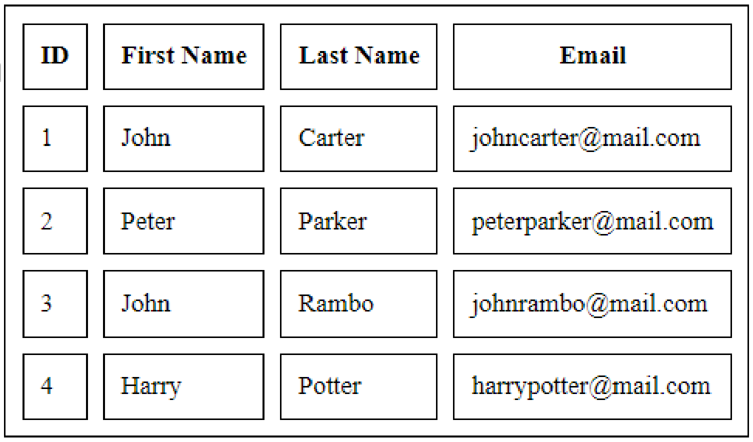


ID	First Name	Last Name	Email
1	John	Carter	johncarter@mail.com
2	Peter	Parker	peterparker@mail.com
3	John	Rambo	johnrambo@mail.com
4	Harry	Potter	harrypotter@mail.com

You can also adjust the spacing between the borders of the cells using the CSS `border-spacing` property, if the borders of your table are separated (which is default).

The following style rules apply the spacing of 10 pixels between all borders within a table:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Example of Setting Table Cell Spacing</title>
<style>
    table {
        border-spacing: 10px;
    }
    table, th, td {
        border: 1px solid black;
    }
    th, td {
        padding: 10px;
    }
</style>
</head>
<body>
    <table>
        <tr>
            <th>ID</th>
            <th>First Name</th>
            <th>Last Name</th>
            <th>Email</th>
        </tr>
        <tr>
            <td>1</td>
            <td>John</td>
            <td>Carter</td>
            <td>johncarter@mail.com</td>
        </tr>
        <tr>
            <td>2</td>
            <td>Peter</td>
            <td>Parker</td>
            <td>peterparker@mail.com</td>
        </tr>
        <tr>
            <td>3</td>
            <td>John</td>
            <td>Rambo</td>
            <td>johnrambo@mail.com</td>
        </tr>
        <tr>
            <td>4</td>
            <td>Harry</td>
            <td>Potter</td>
            <td>harrypotter@mail.com</td>
        </tr>
    </table>
</body>
</html>
```



ID	First Name	Last Name	Email
1	John	Carter	johncarter@mail.com
2	Peter	Parker	peterparker@mail.com
3	John	Rambo	johnrambo@mail.com
4	Harry	Potter	harrypotter@mail.com

Setting Table Width and Height

By default, a table will render just wide and tall enough to contain all of its contents.

However, you can also set the width and height of the table as well as its cells explicitly using the `width` and `height` CSS property. The style rules in the following example will sets the width of the table to 100%, and the height of the table header cells to 40px.

```
table {
    width: 100%;
    border-collapse: collapse;
}
th, td {
    padding: 8px;
    border: 1px solid #dee2e6;
}
th {
    height: 40px;
    text-align: left;
}
```

Controlling the Table Layout

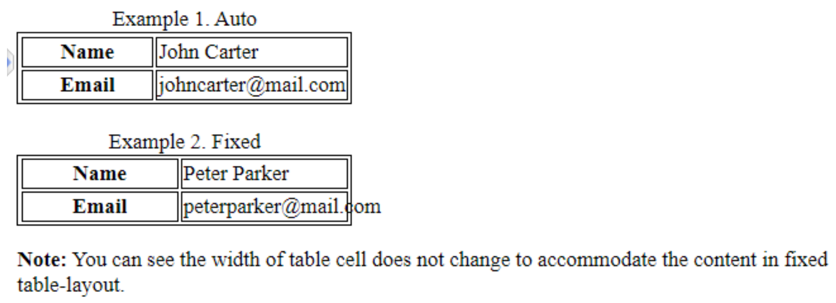
A table expands and contracts to accommodate the data contained inside it. This is the default behavior. As data fills inside the table, it continues to expand as long as there is space. Sometimes, however, it is necessary to set a fixed width for the table in order to manage the layout.

You can do this with the help of CSS `table-layout` property. This property defines the algorithm to be used to layout the table cells, rows, and columns. This property takes one of two values:

- **auto** — Uses an automatic table layout algorithm. With this algorithm, the widths of the table and its cells are adjusted to fit the content. This is the default value.
- **fixed** — Uses the fixed table layout algorithm. With this algorithm, the horizontal layout of the table does not depend on the contents of the cells; it only depends on the table's width, the width of the columns, and borders or cell spacing. It is normally faster than auto.

The style rules in the following example specify that the HTML table is laid out using the fixed layout algorithm and has a fixed width of 300 pixels. Let's try it out and see how it works:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Example of CSS table-layout property</title>
<style>
    table {
        width: 250px;
        border-collapse: separate;
    }
    table, tr, th, td{
        border: 1px solid #000000;
    }
    .auto {
        table-layout: auto;
    }
    .fixed {
        table-layout: fixed;
    }
    td{
        width: 50%;
    }
</style>
</head>
<body>
    <table class="auto">
        <caption>Example 1. Auto</caption>
        <tr>
            <th>Name</th>
            <td>John Carter</td>
        </tr>
        <tr>
            <th>Email</th>
            <td>johncarter@mail.com</td>
        </tr>
    </table>
    <br>
    <table class="fixed">
        <caption>Example 2. Fixed</caption>
        <tr>
            <th>Name</th>
            <td>Peter Parker</td>
        </tr>
        <tr>
            <th>Email</th>
            <td>peterparker@mail.com</td>
        </tr>
    </table>
  <p><strong>Note:</strong> You can see the width of table cell does not change
to accommodate the content in fixed table-layout.</p>
</body>
</html>
```



Example 1. Auto

Name	John Carter
Email	johncarter@mail.com

Example 2. Fixed

Name	Peter Parker
Email	peterparker@mail.com

Note: You can see the width of table cell does not change to accommodate the content in fixed table-layout.

Tip: You can optimize the table rendering performance by specifying the value `fixed` for the `table-layout` property. Fixed value of this property causes the table to be rendered one row at a time, providing users with information at a faster pace.

Note: Without `fixed` value of the `table-layout` property on large tables, users won't see any part of the table until the browser has rendered the whole table.

Aligning the Text Inside Table Cells

You can align text content inside the table cells either horizontally or vertically.

Horizontal Alignment of Cell Contents

For horizontal alignment of text inside the table cells you can use the `text-align` property in the same way as you use with other elements. You align text to either left, right, center or justify.

The following style rules will left-align the text inside the `<th>` elements.

```
table {
    width: 100%;
    border-collapse: collapse;
}
th, td {
    padding: 8px;
    border: 1px solid #dee2e6;
}
th {
    text-align: left;
}
```

Note: Text inside the `<td>` elements are left-aligned by default, whereas the text inside the `<th>` elements are center-aligned and rendered in bold font by default.

Vertical Alignment of Cell Contents

Similarly, you can vertically align the content inside the `<th>` and `<td>` elements to top, bottom, or middle using the CSS `vertical-align` property. The default vertical alignment is middle.

The following style rules will vertically bottom-align the text inside the `<th>` elements.

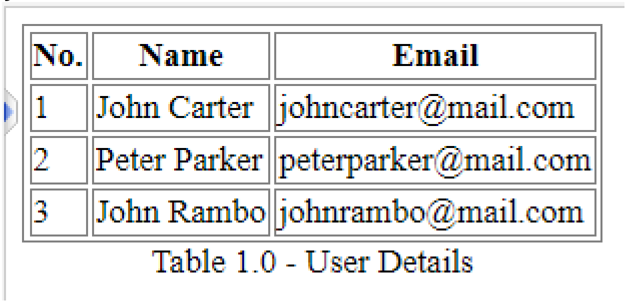
```
table {
    width: 100%;
    border-collapse: collapse;
}
th, td {
    padding: 8px;
    border: 1px solid #dee2e6;
}
th {
    height: 40px;
    vertical-align: bottom;
}
```

Controlling the Position of Table Caption

You can set the vertical position of a table caption using the CSS `caption-side` property.

The caption can be placed either at the top or bottom of the table. The default position is top.

```
table, td, th {
    border: 1px solid gray;
}
caption {
    caption-side: bottom;
}
```



No.	Name	Email
1	John Carter	johncarter@mail.com
2	Peter Parker	peterparker@mail.com
3	John Rambo	johnrambo@mail.com

Table 1.0 - User Details

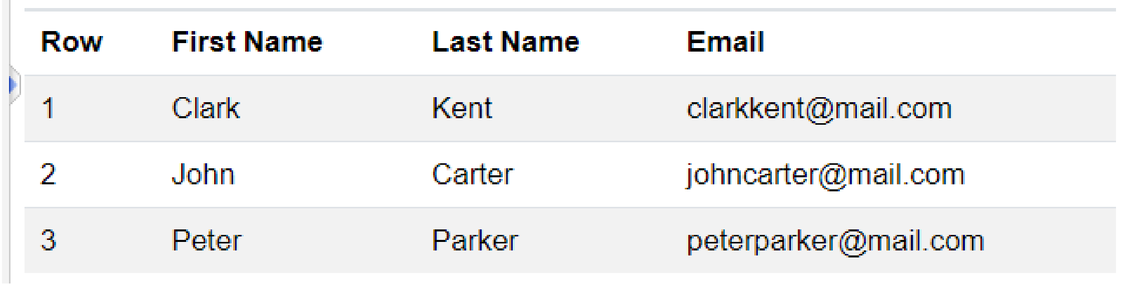
Creating Zebra-striped Tables

Setting different background colors for alternate rows is a popular technique to improve the readability of tables that has large amount of data. This is commonly known as zebra-striping a table.

You can simply achieve this effect by using the CSS `:nth-child()` pseudo-class selector.

The following style rules will highlight every odd rows within the table body.

```css
table {
    width: 100%;
    font-family: arial, sans-serif;
    border-collapse: collapse;
}
th, td {
    padding: 8px;
    text-align: left;
    border-top: 1px solid #dee2e6;
}
tbody tr:nth-child(odd) {
    background-color: #f2f2f2;
}
```



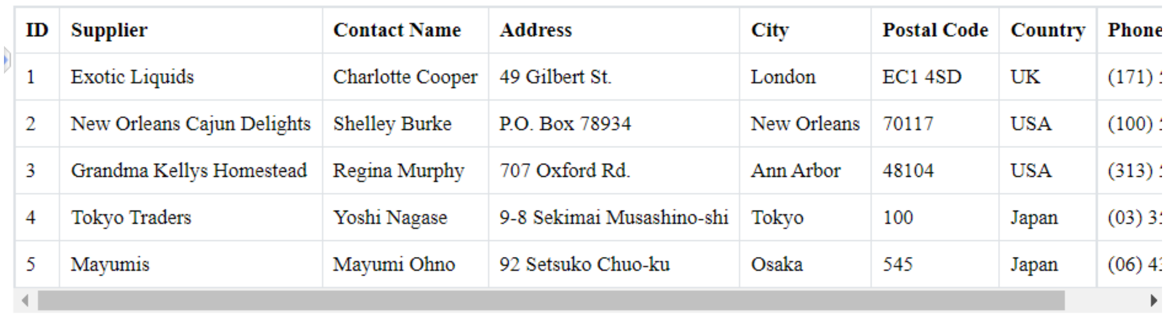
Row	First Name	Last Name	Email
1	Clark	Kent	clarkkent@mail.com
2	John	Carter	johncarter@mail.com
3	Peter	Parker	peterparker@mail.com

Note: The `:nth-child()` pseudo-class select elements based on their position in a group of siblings. It can take a number, a keyword even or odd, or an expression of the form xn+y where x and y are integers (e.g. 1n, 2n, 2n+1, ...) as an argument.

Making a Table Responsive

Tables are not responsive in nature. However, to support mobile devices you can add responsiveness to your tables by enabling horizontal scrolling on small screens. To do this simply wrap your table with a `<div>` element and apply the style `overflow-x: auto;` as shown below:

```
table {
    width: 100%;
    border-collapse: collapse;
}
th, td {
    padding: 8px;
    text-align: left;
    border: 1px solid #dee2e6;
    white-space: nowrap; /* to prevent text wrapping */
}
.responsive-table {
    overflow-x: auto;
}
```



ID	Supplier	Contact Name	Address	City	Postal Code	Country	Phone
1	Exotic Liquids	Charlotte Cooper	49 Gilbert St.	London	EC1 4SD	UK	(171) :
2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA	(100) :
3	Grandma Kellys Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA	(313) :
4	Tokyo Traders	Yoshi Nagase	9-8 Sekimai Musashino-shi	Tokyo	100	Japan	(03) 3:
5	Mayumis	Mayumi Ohno	92 Setsuko Chuo-ku	Osaka	545	Japan	(06) 4:

```
<body>
  <div class="responsive-table">
      <table>
          <thead>
              <tr>
                  <th>ID</th>
                  <th>Supplier</th>
                  <th>Contact Name</th>
                  <th>Address</th>
                  <th>City</th>
                  <th>Postal Code</th>
                  <th>Country</th>
                  <th>Phone</th>
              </tr>
          </thead>
          <tbody>
              <tr>
                  <td>1</td>
                  <td>Exotic Liquids</td>
                  <td>Charlotte Cooper</td>
                  <td>49 Gilbert St.</td>
```

```
        <td>London</td>
        <td>EC1 4SD</td>
        <td>UK</td>
        <td>(171) 555-2222</td>
    </tr>
    <tr>
        <td>2</td>
        <td>New Orleans Cajun Delights</td>
        <td>Shelley Burke</td>
        <td>P.O. Box 78934</td>
        <td>New Orleans</td>
        <td>70117</td>
        <td>USA</td>
        <td>(100) 555-4822</td>
    </tr>
    <tr>
        <td>3</td>
        <td>Grandma Kellys Homestead</td>
        <td>Regina Murphy</td>
        <td>707 Oxford Rd.</td>
        <td>Ann Arbor</td>
        <td>48104</td>
        <td>USA</td>
        <td>(313) 555-5735</td>
    </tr>
    <tr>
        <td>4</td>
        <td>Tokyo Traders</td>
        <td>Yoshi Nagase</td>
        <td>9-8 Sekimai Musashino-shi</td>
        <td>Tokyo</td>
        <td>100</td>
        <td>Japan</td>
        <td>(03) 3555-5011</td>
    </tr>
    <tr>
        <td>5</td>
        <td>Mayumis</td>
        <td>Mayumi Ohno</td>
        <td>92 Setsuko Chuo-ku</td>
        <td>Osaka</td>
        <td>545</td>
        <td>Japan</td>
        <td>(06) 431-7877</td>
    </tr>
```

```
          </tbody>
      </table>
  </div>
  <p><strong>Note:</strong> Toggle the editor layout or open the output in a
blank window and resize it to understand how responsive table works.</p>
</body>
```