Length, Size Types and URL Values

In this lesson, we'll see the different ways to specify size and lengths of various HTML elements in CSS and how to achieve cross-platform size and length compatibility. In addition, we will touch upon URL values.
Let's begin!

WE'LL COVER THE FOLLOWING

- Size type options
- URL values

When designing the layout of a page, you often need to specify sizes, such as the width or height of a box, top and bottom margins of paragraphs, and so on.

To indicate size, CSS offers a number of type size options including:

- pixels
- points
- picas
- inches

- centimeters
- millimeters
- em-heights
- ex heights
- percentages

When you define a style property that specifies a size or length value, you always need to specify the unit of measure as well

The above point is shown in the example below:

```
p {
    margin-top: 4px;
    margin-right: 1in;
    margin-bottom: 0.5em;
    margin-left: 0;
}
```

using size types to specify the unit of measure of style properties

The only value that does not need a unit of measure is zero, for it is exactly the same independent of the unit used to calculate it.

The table below summarizes the size type options:

Size type options

Unit	Description
`px`	`px` stands for *pixels*. Pixels are most often used when you want to precisely align elements to images because images are measured in pixels.
`in`	`in` stands for *logical inches*. The epithet "logical" refers to the fact that the actual physical size depends on the monitor and settings chosen by the operating system or the user. The size of dots of a monitor determines the physical size of its pixels, and thus the physical size of the logical inch. Be careful when using logical inches—and all other fixed units of measure—, because they do not scale well on systems with different dot-per-inch settings.
`pt`	This unit specifies *points*. A point is 1/72 of a logical inch.
`pc`	This unit stands for *picas*. A pica is 12 points or 1/6 of a logical inch.
`cm`	It stands for *centimeters*. One logical inch is 2.54 centimeters.
`mm`	This unit specifies *millimeters*. One logical inch is 25.4 millimeters.
`em`	While all entries above are fixed units of measure, `em` is a flexible unit; it is the `font-size` value assigned to an element. The only exception is the `font-size` property itself, when you assign a value to it, `em` represents the `font-size` value of the parent element. For example, `3em` is three times the `font-size`. Use `em`s when you want to size an element relative to the size of its text. This allows the layout of your documents to flex with the size of the text.
`ex`	`ex` is also a flexible unit, is the height of the letter "x" of an element's current font. This measurement is rarely used.

Logical units (in, pt, pc, cm, mm) do not scale well on systems with different *dot-per-inch* settings. What may seem right on Windows 8 at $96dpi$, may be too large or too small on other systems.

Instead of these size types:

*use **percentages** or **ems** if you want to provide the best **cross-platform compatibility**.*

To understand better how `ems` work, take a look at this sample:

HTML Output

```html
<!DOCTYPE html>
<html>
<head>
  <title>Using ems</title>
  <style>
    body { font-size: 30px; }
    p {
        margin-top: 0.5em;
        margin-bottom: 0.5em;
        font-size: 0.8em;
    }
    h1 { font-size: 2em; }
  </style>
</head>
<body>
  <h1>I'm a heading</h1>
  <p>I'm paragraph</p>
</body>
</html>
```

example usage of ems

Here, `body` defines that the font size should be $30pixels$. The `<p>` tag within `<body>` uses the p style rule, which specifies all sizes in `ems`. The `font-size` value is specified as $0.8em$, and it is calculated by the `font-size` of the parent element (`<body>`), so the paragraph will use a $30px * 0.8 = 24px$ font.

The `margin-top` and `margin-bottom` values are also specified with `ems`, but they use the `<p>` font size ($24px$) as the base of height calculation. So, `margin-top` and `margin-bottom` is set to $24px * 0.5 = 12px$. Using the same logic, the `font-size` of the `<h1>` tag is to $60pixels$.

There is another flexible unit of measure you can use: ***percentages.***

CSS uses this unit for many different purposes, like determining the **width** or **height** of an element, **sizing text**, specifying the **placement of an image** in the background of a style, and so on.

What is considered a percentage varies from property to property.

For example, for font sizes, the percentage is calculated based on the text's inherited value. When applied to width, however, percentages are calculated based on the width of the page or on the width of the nearest parent element.

URL values

A few properties in CSS use URL values. For example, the `background-image` property accepts a URL that points to a file on the web. This file is used to assign an image as a background for a related page element.

The syntax of specifying URLs is simple:

```
body {
  background-image: url(Images/pageBkg.png)
}
```

syntax of specifying URLs

As you expect, this property value gets the **pageBkg.png** file from the Images folder.

You can use optional single quotes or double quotes to wrap URLs, such as in these examples:

```
url('Images/pageBkg.png')
url("Images/pageBkg.png")
```

wrapping URLs in quotes

Now that we have sufficiently gotten acquainted with length and size types, along with URL values in CSS, in the *next lesson*, we will meet the baseline style sheet.