

# CSS

## What is it?

They hammered out thin *sheets* of gold and cut strands to be worked into the blue, purple and scarlet yarn and fine linen—the work of skilled hands. [Exodus 39:3 NIV](#)

**CSS** stands for **C**ascading **S**tyle **S**heet. It specifies the layout, color, fonts, and spacing of the web elements in a web page.

It was developed by Håkon Wium Lie (<http://people.opera.com/howcome/>) while working for [W3C](#) in 1994. He is the current Chief Technology Officer of [Opera Software](#).

## Why should I care?

**CSS** is also used to style the web pages of a Chrome Application.

You link the CSS file with the HTML file using the link element like this:

```
<link rel="stylesheet" href="styles.css">
```

The link element must go inside the head element.

Since the link element can also be used for other things, we need to tell the browser that this is a link to a style sheet. That is done with the rel attribute.

The href attribute indicates the location of the CSS file. If it is in the same folder as the HTML file, you can just put the file name.

# CSS

Each styling element in the CSS file starts with a comma-separated list of selectors, followed by a semicolon separated list of statements in curly brackets like this:

```
selector[, selector...] {  
  property: value;  
  [property: value;  
  ...  
}
```

Example

```
* {  
  font-family: sans-serif;  
}
```

## Details

### Fonts

In the example above we are setting the `font-family` for every element to `sans-serif`.

This text is in sans-serif

This text is in serif

# CSS

## Selectors

The selector is the pointer to elements in the HTML page.

The \* selector points to every element.

By specifying the name of an HTML element, you select every element with that name. For instance, if the selector is `div`, every `div` element in the web page is styled according to the statements you give inside the curly brackets following the selector.

A selector that is prefixed with `#` will point to elements with an `id` attribute value of the selector.

HTML:

```
<div id="div-1">Content</div>
```

CSS:

```
#div-1 {  
    height: 10px;  
}
```

A selector that is prefixed with `.` will point to elements with a `class` attribute value of the selector.

HTML:

```
<div class="smallish">Content</div>
```

CSS:

```
.smallish {  
    height: 10px;  
}
```

# CSS

You can also combine the class selector with other selectors like this:

```
div.smallish {  
    height: 10px;  
}
```

This is now only selecting the `div` elements that have a class attribute with a value of `smallish`.

If you specify two selectors separated by space, it means that the right selector has to be a descendant, or inside the other.

Example:

```
div a {  
    font-family: sans-serif;  
}
```

This means that links (`a` elements) inside a `div` element will be `sans-serif`.

## Colors

You specify the color of text with the `color` property, the color of the background is specified with the `background-color` property, and the border color with the `border-color` property.

Example:

```
div {  
    color: white;  
    background-color: blue;  
    border-color: black;  
}
```

# CSS

Look at my colors!

The value of a color property can be specified as the name of the color. There are more than 150 color names that are supported.

The basic colors that must be supported by all browsers are specified here:

<http://www.w3.org/TR/css3-color/#html4>

Most popular browsers also support these color

names: <http://www.w3.org/TR/css3-color/#svg-color>

You can also specify the color as a numeric value. It is specified as a 6-digit hexadecimal value with a pound-sign (#) in front. The first two digits indicate the amount of red in the color, the next two digits the amount of green, and the last two indicate the amount of blue in the color. Since we add the colors together, and white contains all colors, you would specify white as #FFFFFF. The color red would be specified as #FF0000, the color green as #00FF00, and the color blue as #0000FF. This way of specifying a color as the red, green, and blue components is called RGB (**R**ed **G**reen **B**lue).

Example:

```
div {  
  color: #FFFFFF;  
  background-color: #0000FF;  
  border-color: #000000;  
}
```

# CSS

Look at my colors!

The Web site W3Schools.com has a color mixer page that can help you find the numeric values of colors:

[http://www.w3schools.com/tags/ref\\_colormixer.asp](http://www.w3schools.com/tags/ref_colormixer.asp)

Be careful when you choose your colors. Some colors look great together, and some look terrible. Choosing a *color scheme* that has been created by a designer is a good idea. The Web site [Paletton \(http://paletton.com/\)](http://paletton.com/) is great to put a color scheme together.

Consider also that some people are color blind, and choosing your colors wisely will help them use your application.

Finally if you are not familiar with hexadecimal values, you can also specify the numeric color values as a comma separated list like this: `rgb(255, 255, 255)`.

Example:

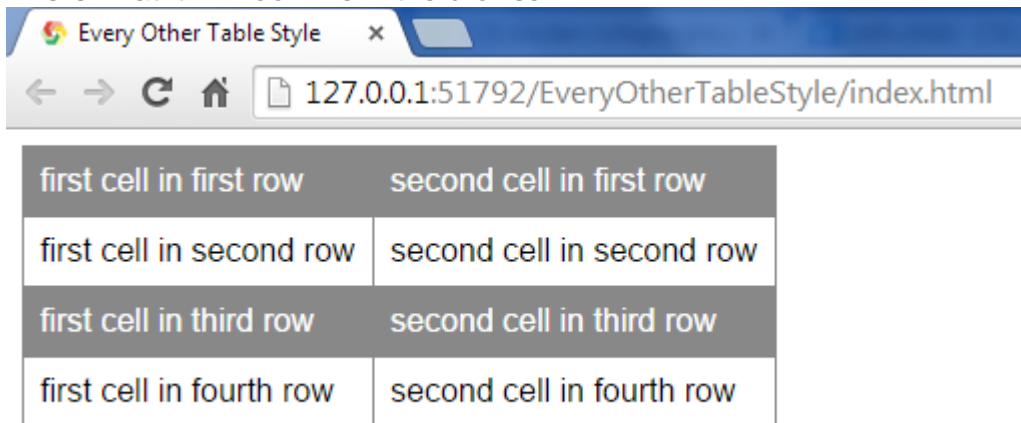
```
div {  
  color: rgb(255, 255, 255);  
  background-color: rgb(0, 0, 255);  
  border-color: rgb(0, 0, 0);  
}
```

# CSS

Look at my colors!

What if we wanted to color every other row in a table differently. This is a very usual effect, and of course this is possible to do with CSS.

This is what it will look like in the browser:



A screenshot of a web browser window. The title bar says "Every Other Table Style". The address bar shows the URL "127.0.0.1:51792/EveryOtherTableStyle/index.html". Below the browser window is a table with 4 rows and 2 columns. The first, third, and fourth rows have a dark gray background, while the second row has a white background. The text in the cells is as follows:

first cell in first row	second cell in first row
first cell in second row	second cell in second row
first cell in third row	second cell in third row
first cell in fourth row	second cell in fourth row

Here is the HTML:

```
<table>
  <tr>
    <td>first cell in first row</td>
    <td>second cell in first row</td>
  </tr>
  <tr>
    <td>first cell in second row</td>
    <td>second cell in second row</td>
  </tr>
  <tr>
    <td>first cell in third row</td>
    <td>second cell in third row</td>
  </tr>
  <tr>
    <td>first cell in fourth row</td>
    <td>second cell in fourth row</td>
  </tr>
```

# CSS

```
<td>first cell in third row</td>
<td>second cell in third row</td>
</tr>
<tr>
  <td>first cell in fourth row</td>
  <td>second cell in fourth row</td>
</tr>
</table>
```

You have to put it inside the body element of your index.html file.

Here is the CSS:

```
* {
  font-family: sans-serif;
}

table {
  border-collapse: collapse;
}

td {
  padding: 0.5em;
  border-width: thin;
  border-style: solid;
  border-color: #888888;
}

tr:nth-child(2n+1) {
  background-color: #888888;
  color: #ffffff;
}
```



# CSS

```
}
```

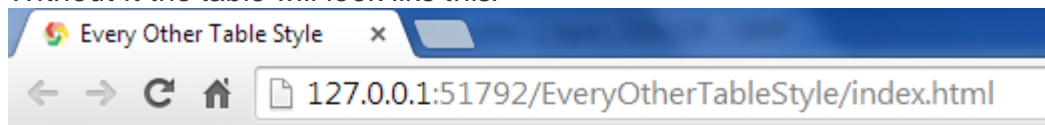
First we use the `*` selector to select every element in the web page, and set the font-family to `sans-serif`.

```
* {  
  font-family: sans-serif;  
}
```

Next we use the `table` selector to select every table element, and set the `border-collapse` property to `collapse`. This just means that if two elements are next to each other, and they both have a border, only one border will be shown.

```
table {  
  border-collapse: collapse;  
}
```

Without it the table will look like this:



first cell in first row	second cell in first row
first cell in second row	second cell in second row
first cell in third row	second cell in third row
first cell in fourth row	second cell in fourth row

The `td` selector is used to set padding and border properties for every cell.

```
td {  
  padding: 0.5em;  
  border-width: thin;
```

# CSS

```
border-style: solid;
border-color: #888888;
}
```

The padding property value has the effect of adding some space between the border and the content of the cell. The value `0.5em` means half a character size, so if the font size changes, so does this value. The border properties gives us a border that is a `solid` thin line, with a gray color.

Now comes the interesting part; the `:nth-child` selector.

```
tr:nth-child(2n+1) {
  background-color: #888888;
  color: #ffffff;
}
```

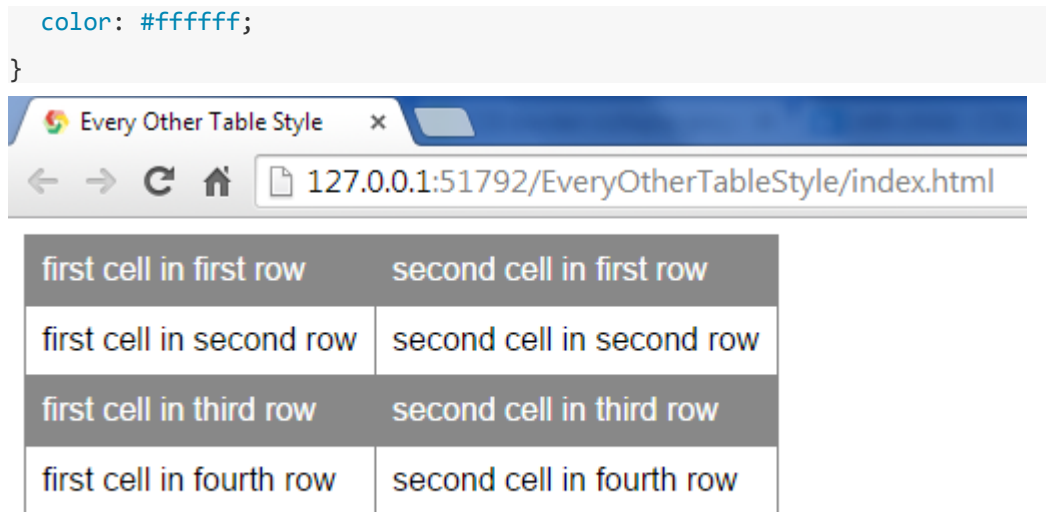
The argument in parentheses;  $2n+1$  indicates index into the sibling elements in the parent element. The first child has index 1. So for any value of  $n$ ; 0, 1, 2, ... we multiply by 2 and add 1. The result becomes 1, 3, 5, ..., so all the odd numbers. The syntax is  $a_n+b$ . We could also have specified just  $2n$ , and then it would have been all the even indexes. Since this is kind of complicated and hard to remember, you can also just give the argument `odd` or `even`.

Suppose we want to change just one particular element? That is also possible by not including the  $n$  in the argument. So if we wanted all the odd rows to be gray, but the very first row to be blue, we can do it like this:

```
tr:nth-child(1) {
  background-color: blue;
  color: #ffffff;
}

tr:nth-child(odd) {
  background-color: #888888;
```

# CSS



Why did that not work? The first row is not blue.

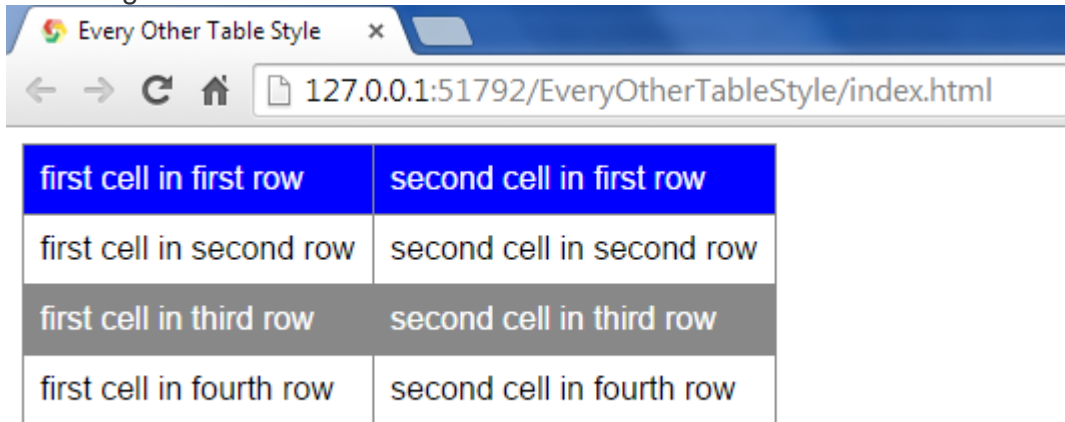
Oh yeah, CSS means Cascading Style Sheets. So if something comes later it effectively adds to the effects already done in a cascading fashion. We first set the style for index 1, and then for all the odd indexes, and since 1 is an odd number, it overrides the first effect for the first row.

What we need to do is reverse the order like this:

```
tr:nth-child(odd) {  
  background-color: #888888;  
  color: #ffffff;  
}  
  
tr:nth-child(1) {  
  background-color: blue;  
  color: #ffffff;  
}
```

# CSS

Now we got the desired effect:



first cell in first row	second cell in first row
first cell in second row	second cell in second row
first cell in third row	second cell in third row
first cell in fourth row	second cell in fourth row

You can read more about this selector here:

<https://developer.mozilla.org/en-US/docs/Web/CSS/:nth-child>

## Text Effects

In the previous section we saw that text can have different colors, but using CSS you can also make other effects on the text in you web page.

You can underline text using the `text-decoration` property with the `underline` value.

Example:

```
div {  
  text-decoration: underline;  
}
```

## CSS

Look at my effects!

You can make it italics using the `font-style` property with the `italic` value.

Example:

```
div {  
  font-style: italic;  
}
```

*Look at my effects!*

You can make it bold using the `font-weight` property with the `bold` value.

Example:

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

**Look at my effects!**

You can make shadows using the `text-shadow` property. The value is specified as 4 arguments. The first is the horizontal distance of the shadow, the second is the vertical distance of the shadow, the third is the blur distance, and the fourth is the color of the shadow.

Example:

```
div {
```

# CSS

```
text-shadow: 0.1em 0.1em 0.1em black;  
}
```

Look at my effects!

## Layout

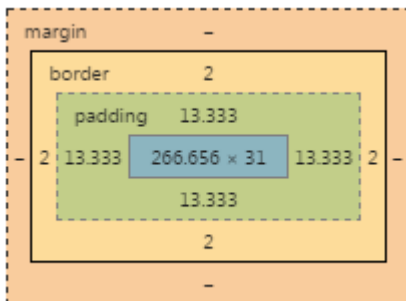
Each HTML element has the following areas:

Content

Padding

Border

Margin



The content is the innermost box. Around the content is the padding, then the border, and finally the margin. The margin gives space to surrounding elements.

# CSS

Use the following properties to describe the size of the different areas:

Property	Description
width	The width of the content area
height	The height of the content area
padding	The width of the padding area. Specify as 4 values separated by space. They indicate <i>top</i> , <i>right</i> , <i>bottom</i> , and <i>left</i> respectively. You can also specify the padding for individual directions by using the properties <code>padding-top</code> , <code>padding-right</code> , <code>padding-bottom</code> , and <code>padding-left</code>
border-width	The width of the border. As with the padding, you can specify it as 4 values separated by space indicating <i>top</i> , <i>right</i> , <i>bottom</i> , and <i>left</i> respectively. You can also specify the border width for individual directions by using the properties <code>border-top-width</code> , <code>border-right-width</code> , <code>border-bottom-width</code> , and <code>border-left-width</code>
margin	The width of the margin. As with the padding, you can specify it as 4 values separated by space indicating <i>top</i> , <i>right</i> , <i>bottom</i> , and <i>left</i> respectively. You can also specify the margin for individual directions by using the properties <code>margin-top</code> , <code>margin-right</code> , <code>margin-bottom</code> , and <code>margin-left</code>

Example:

HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Examples</title>
    <link rel="stylesheet" href="styles.css">
  </head>
  <body>
```

# CSS

```
<div id="first">First</div>
<div id="second">Second</div>
</body>
</html>
```

Styles.css:

```
div {
  color: white;
  background-color: blue;
  border-color: black;
  border-style: solid;
}

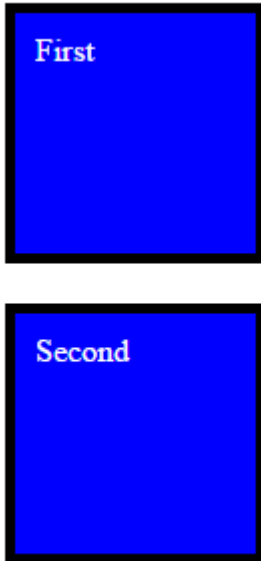
div {
  width: 100px;
  height: 100px;
  padding: 10px;
  border-width: 5px;
  margin: 20px;
}
```

Render:



# CSS

---



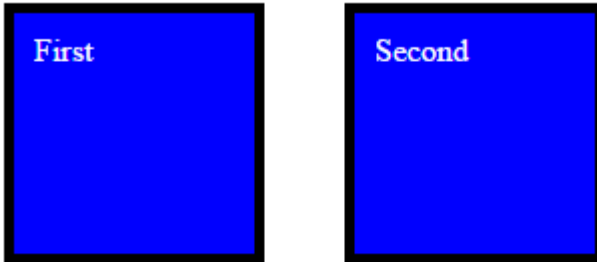
Did you notice how the two `div` elements were laid out vertically one after the other? We can change that using the `float` property. It can have the values `left`, `right`, or `none`, where `none` is the default. If we set `float` to `left` it means that the element should float over to the left inside the element it is in. If you float another element left, it will float as far to the left as it can. We already floated one element, so it will be placed to the right of that one. Let's try it.

We will add the following to the `Styles.css` file:

```
div {  
  float: left;  
}
```

It now renders like this:

# CSS



Normally the elements in a web page renders in one layer, but guess what? We can also change that.

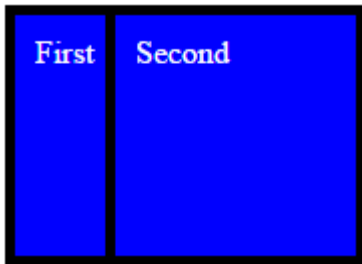
The `z-index` property is used to indicate the elements elevation. The higher the number, the higher the elevation. You can imagine that elements with a higher elevation could cover other elements with lower elevation.

Let's try it.

```
div#first {  
  z-index: 1;  
  position: absolute;  
  left: 0;  
}  
  
div#second {  
  z-index: 2;  
  position: absolute;  
  left: 50px;  
}
```

It renders like this:

# CSS



## Units

Have you noticed that I have put `px` after each size? It means that I have given the size in pixels. A pixel is a tiny dot on your screen. Usually there will be 72 such pixels in one inch. We say that the pixel density of the screen is 72 PPI (**P**ixels **P**er **I**nc). Some people use DPI (**D**ots **P**er **I**nc) for the same thing. Some displays have higher PPI to be able to create a clearer picture in a smaller area. Therefore if you specify your layout using `px`, it will look smaller on some devices, and bigger on others.

Fortunately there are other ways to specify sizes than `px`. You can use `in` or `%`. The `in` unit indicate size in inches. In this way you are guaranteed that the size will be the same no matter what display the web page is being shown on. One risk here is that the page will be too big then to fit on some devices, and too small to look good on other devices.

The `%` unit to the rescue! This unit is used to indicate a size as a percentage of the parent element's size. So if you want one element that fills the whole screen, you can give it a `width` and a `height` property with the value `100%`.

# CSS

Let's try it!

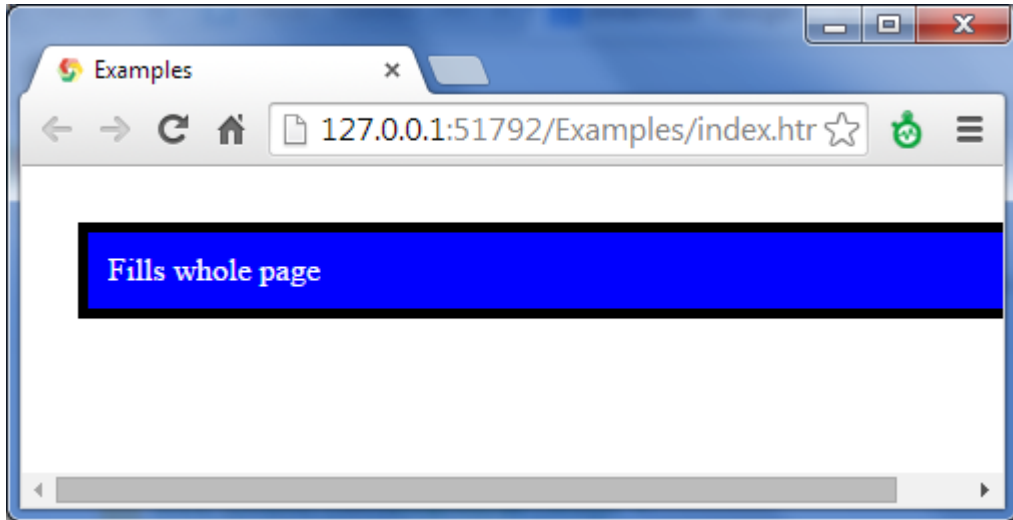
HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Examples</title>
    <link rel="stylesheet" href="styles.css">
  </head>
  <body>
    <div id="whole">Fills whole page</div>
  </body>
</html>
```

CSS:

```
div#whole {
  width: 100%;
  height: 100%;
}
```

# CSS



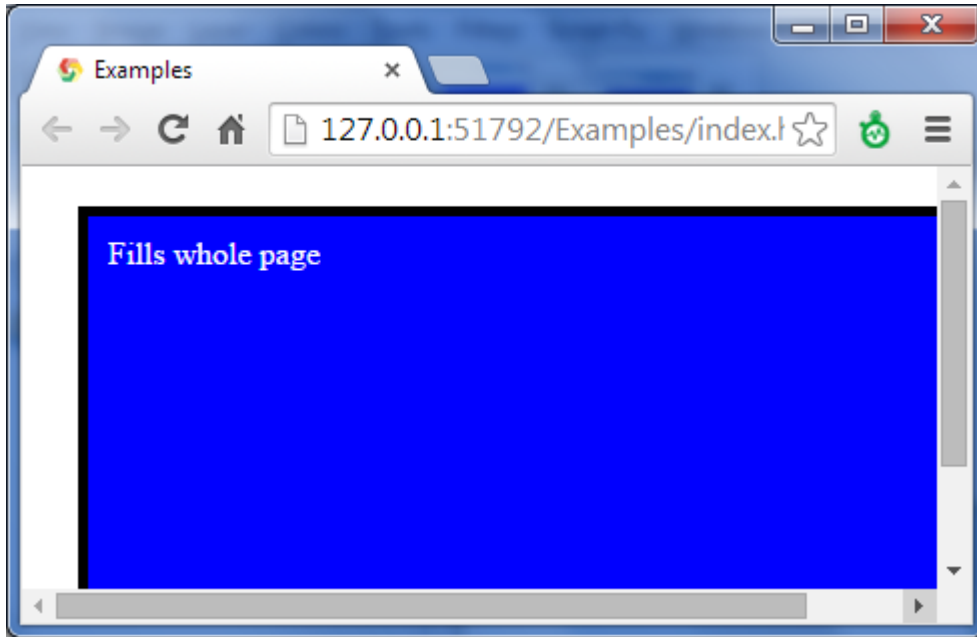
Oops! What happened?

Well the `height: 100%` means 100% of the height of the parent. Since the parent is the `body` element that doesn't have any height defined it will have `auto` by default, which means to fit to content.

The solution is to set `height: 100%` on every parent element like this:

```
html, body {  
  height: 100%;  
}
```

# CSS



It is better, but still not what we want. If our content is 100% of the parent, and adding the margin, border, and padding the total area of our element is going to be bigger than the parent, and the scroll-bars appear.

Final solution:

```
* {
  padding: 0;
  border-width: 0;
  margin: 0;
}

div {
  color: white;
  background-color: blue;
  border-color: black;
```

# CSS

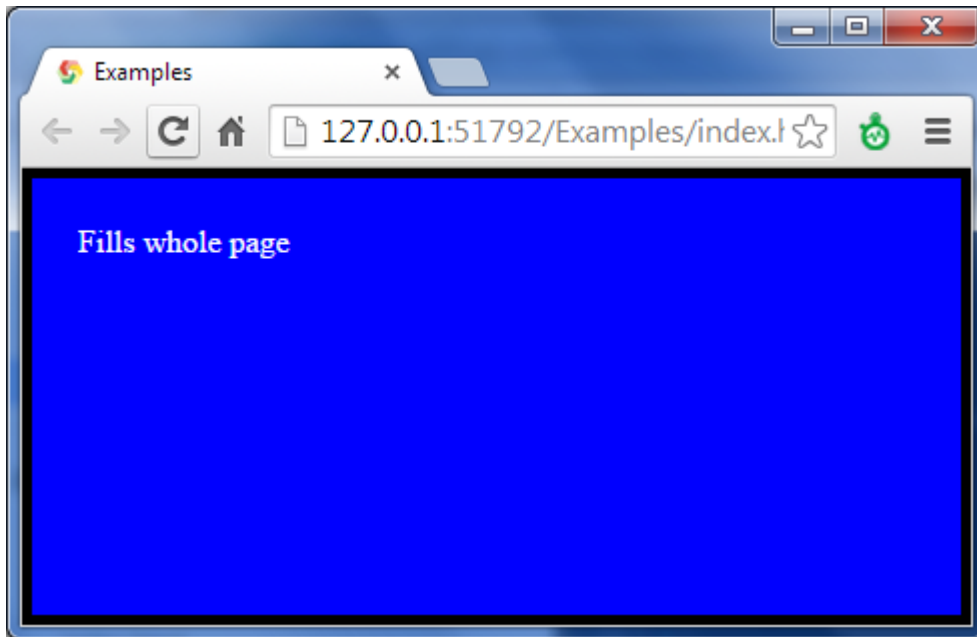
```
border-style: solid;
}

html, body {
  height: 100%;
  width: 100%;
}

body {
  padding: 1%;
  box-sizing: border-box;
  background-color: black;
}

div#whole {
  width: 100%;
  height: 100%;
  padding: 5%;
  box-sizing: border-box;
}
```

# CSS



There are a couple of tricks here. First we are using the `*` selector to select every element in the page and set padding, border-width, and margin to 0.

```
* {  
  padding: 0;  
  border-width: 0;  
  margin: 0;  
}
```

The next thing to note is that the `div#whole` element has a new property `box-sizing` that we haven't talked about before. The property describes how the size of a box is to be calculated. The default value is `content-box`. It means that width and height only indicate the content. The `border-box` value that we specified means that the width and height now are taken to include padding and border.



# CSS

```
div#whole {  
  width: 100%;  
  height: 100%;  
  padding: 5%;  
  box-sizing: border-box;  
}
```

The final trick is that the border now scales with the size of the window. Since we cannot specify a border-width in percent - I didn't tell you this before, but now you know - we specify the padding of the parent element instead, and set the background-color of the parent to the desired border color; black.

```
body {  
  padding: 1%;  
  box-sizing: border-box;  
  background-color: black;  
}
```

## Where can I find more information?

CSS Tutorial (<http://www.w3schools.com/css/>)

CSS | MDN (<https://developer.mozilla.org/en-US/docs/Web/CSS>)

Cascading Style Sheets (<http://www.w3.org/Style/CSS/Overview.en.html>)