CSS: part 4

**Kalob**

**4.1 CSS syntax**

Selector {property: value; property: value }

**4.2 different way to style CSS**

* Inline: <tag style=" property: value ">
* Inside HTML: inside <style></style>
* Separate CSS file: in .css file. <link >

<link href="color\_STYL2.css" rel="stylesheet" type="text/css">

* Use "id" and "class": ***id*** is unique and different element can have same ***class***.

In style sheet: use ***#id\_name*** for an ***id="id\_name"***.

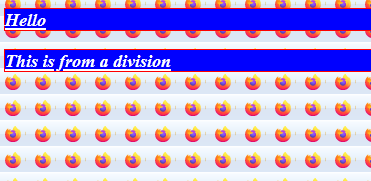
**4.3 Different CSS styles**

* Background ***color***; Background ***image***; Background ***repeat***; Background ***Attachment***; Background ***Position***

body{

    background-image: url(./mz.PNG);

}



body{

    background-image: url(./mz.PNG);

    background-repeat: repeat-x;

}



body{

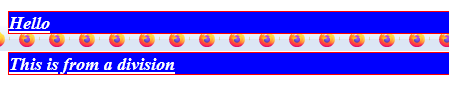
    background-image: url(./mz.PNG);

    background-repeat: repeat-x;

    background-attachment: scroll;

    background-position: 10px;

}



**4.4 Text styling**

* color; for cloring the text
* font-style; italic
* font-weight; bold, normal
* text-decoration; underline, linethrough
* font-size; bigger/smaller fonts
* font-family; serif, sans-serif or other fonts
* text-align;
* text-indent;
* text-overflow;
* text-transform;
* Including fonts from file:

/\* using local font \*/

@font-face{

    font-family: myFirstFont;

    src: url(abd.otf);

}

div{

    font-family: myFirstFont;

}

**4.5 Styling Links**

We'll use pseudo-classes: "**:link**", "**:visited**", "**:hover**", "**:active**".

a:hover{

    color: aqua;

}

a:link{

    color: rgb(251, 255, 0);

}

a:active{

    color: rgb(174, 0, 255);

}

a:visited{

    color: rgb(255, 0, 0);

}

* Links with classes:

a.className:pseudoclass

**4.6 lists Styling**

div{

    list-style-type: circle;

    display: inline;

}

/\* custom list imaging \*/

li{

    background-image: url(./mz.PNG);

    background-repeat: no-repeat;

    list-style-type: none;

    padding-left: 15%;

}

**4.7 Width and Height**

Use 50% to take half of the screen. "auto" is default.

**4.8 Borders**

border: size solid black

div{

    border-width: thick;

    /\*    up right bottom left\*/

    border-width: 2px 10px 5px;

    border-style: dashed;

    border-color: bisque;

    /\*Individual side\*/

    border-left-width: thick;

    /\*    top right bottom left\*/

    border-left-width: 2px 10px 5px;

    border-left-style: dashed;

    border-left-color: bisque;

}

​

/\*Short-hand\*/

selector{border: width style color;}

**4.9 Margin and Padding**

* Margin: space outside the element
* Padding: space inside the element

div{

    margin: 10px;

    margin: 10px 20px;

    margin: 10px 20px 30px 60px;

    /\* short hnd \*/

    border: 1p0x solid red;

}

**4.10 changing cursor**

There will be times when you want to change your cursor. Maybe make it ***disapear*** look like it’s ***loading***, or a ***custom image***

div{

    cursor: url("./mz.PNG");

}

* Cursor: URL, auto, crosshair, default, e-resize, help, move, n-resize, ne-resize, rtw-resize, pointer, progress, s-resize, se-resize, sw-resize, text, w-resize, wait, inherit auto is the default value URL is a list of urls to try
* Notably, the most popular ones are URL. for a custom cursor image, default which is just the regular cursor, pointer, which is die cursor when you have your mouse over a link

**4.11 Display**

* Hide an element: **display: none;**
* Transform any inline as Block: **display: block;**
* Transform any block as Inline: **display: inline;**
* Transform any inline-Block: **display: inline-block;** There are many other values.

**4.12 Overflow**

overflow: auto;

**4.13 Grouping and Nesting**

* Grouping: Group styling elements

/\* Grouping using "," \*/

div, #MyId, .blue, h1{

    color: aqua;

    overflow: auto;

}

* Nesting: selecting nested elements

/\* Nesting using " " \*/

div ol li{

    list-style: none;

    color: red;

}

.green #uniqId div{

    background-color: bisque;

    font-family: sans-serif;

}

**4.14 Max/Min-width and height**

div{

    max-width: 1300px;

    min-width: 420px;

    max-height: 75%;

}

**4.15 Positions**

* Absolute:

div{

    position: absolute;

    top: 0;

    left: 0;

}

a{

    position: relative;

    left: -30px;

}

h1{

    position: static;

    background-color: brown;

}

h2{

    position: fixed;

    background-color: wheat;

}

**4.16 Relative & z-index**

div{

    position: absolute;

    z-index: 10;

}

* Negative numbers works too.

**4.17 clear the float**

div{

    float: left;

}

/\* clear will remove floating \*/

div h1{

    clear: both;

}

.div2{

    float: right;

}

**4.18 Pseudo Selector**

:first-child

:first-letter

:first-line

:focus

:lang(value)

div:lang(awesome){}

:before

strong:before{}

:after

strong:after{}

**4.19 Alignment**

* To center: This will align x-axis center.

div{

    margin: 0 auto 0 auto;

    /\* or;

    margin: 0 auto; \*/

}

* Text align: We can also use:

.div{

    text-align: center;

}

* sub/sup:

.vertical{

    vertical-align: middle;

    /\* sub = <sub>

    sup = <sup>

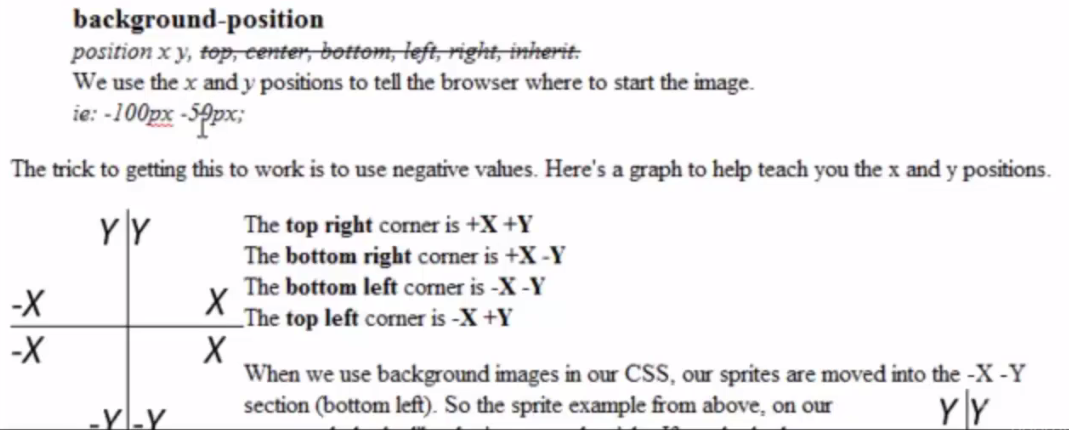
    default = baseline \*/

}

**4.20 Sprites**

Use *sprites* instead of *images*.

|  |  |
| --- | --- |
|  | .sprite{      background-image: url("sprite.png");      background-position: -17px -373px;      background-repeat: no-repeat;      width: 16px;      height: 15px;  } |



**4.21 Opacity**

.div2{

    opacity: 50%;

}

**4.22 Media Types**

@media screen {

    /\* all computer screen stuff goes in here \*/

}

@media handheld{

    /\* mobile devices like iPod touch and cell phone styling would go In here \*/

}

@media print{

    /\* page styling for a printable page. They are usually pretty bland pages to save the user some ink \*/

}

@media tv{

    /\* this styling is for television-type devices \*/

}

**4.23 Attribute selector and targeting by attribute**

[lang=abc]{

    border: 1px solid black;

}

* Targeting by attribute:

/\* Targetting elements \*/

option[name = dark]{

    color: white;

    background: bisque;

}

input[type = text]{

    /\* style \*/

}

**4.24 Browser prefix**

Some old browser doesn't support some styling, for example:

-webkit-font-weight: bold;

Use:

**-o-** for opera

**-webkit-** for chome & safari

**-ms-** for internet explorer

**-moz-** for firefox

* General rule: -prefix-property: value
* Most developers put their prefix after the regular properties.

**4.25 !important**

It will force the style. And this style is not changeable.

Don’t use it too much. It can destucture your styles.

option[name = dark]{

    color: white !important;

    background: bisque;

}

* The only way to overwrite, or remove, and !important statement is to use another !important statement.
* I highly recommend not using the !important statement just to change the styling of one class or element. Think of this as a backup plan for when there is no other option available
* Because this will create a vicious circle of !important statements all over your CSS page.
* Instead, write more code and specify a new class or ID for the element you are trying to style
* Or, better yet, take some time and look at your CSS and restructure it so you don't need any !important statements at all.

**CSS3 properties**

**4.26 Border Radius**

* Border Radius: Border-radius is how we curve the corners of a border. Similar to margin and padding , we can choose which side, or in this case a corner, to smooth

<style>

.topRight     { border-top-right-radius: 10px; }

.bottomtRight { border-bottom-right-radius: 10px; }

.bottomLeft  { border-bottom-left-radius: 10px; }

.topLeft     { border-top-left-radius: 10px; }

/\* IS THE SAME AS \*/

.radius{ border-radius: 10px; }

</style>

* Shorthand: When you use shorthand, instead of going from top-right-bottom-left, we go from top-left to top-right to bottom- right to bottom-left. It’s still in the clockwise direction, you start at 10 O'clock instead of 12 O'clock.

.shortHand {

    border:1px solid red;

    border-radius: 10px 0px 20px 10px;

    /\* READS LIKE THIS border-radius: top-left top-right bottom-right bottom-left; \*/

    }

* Border Image:

.url { border-image : url(BorderImage.gif), . . . }

* Slicing The Image: *This can be tricky to comprehend*. The slice value accepts 1-4 answers starting with the top . If you had four values, you'd be writing *TOP% RIGHT% BOTTOM% LEFT%*. Here's the trick : You do not need to specify a pixel value, but you need to specify the % value .

*example:*

When we slice an image, it gets broken down into 9 parts : . Here's an example with some code

<style>

.url { border-image: url(BorderImage.gif) 20% 20% 20% 20% . . . . .}

</style>

* Repeat/Round/Stretch: Repeat will tile your image sections over and over. Just like background-repeat. Each section from our slices (above) are applied accordingly. So the white corners would be our corners, the pink sides will tile from top to bottom and then the top and bottom section would tile from left to right. But with repeat it will repeat your image over and over and potentially end the repeat in *die wrong place* .So to fix that we’d use round .
* Round is like repeat, but instead of tiling our sections continuously, our sections will only use full tiles. So if the top row' can fit 3.9 tiles, it will use 3 ties and stretch them over the span of the row. Now if we wanted to stretch our image, we'd apply just that. Round, if not supported in your browser, will act as repeat .
* Stretch is how we do not repeat any tiles and simply stretch our one tile (section) over the span of the row or column. Now let’s apply some code.

<style>

.border {

border-image: url(Borderlmage.png) 20 20 20 20 repeat;

border-width: 20px;

}

</style>

The border-image is

TL = Top Left

T = Top

TR = Top Right

R = Right. You get the point!!!

* So if you look at the code, we split the image into 9 sections each with 20px dimensions. Now if you look at the border-image element below, you'll see all the Ts are at the top, all the B's are at the Bottom and so on. And there are no M’s (M stands for middle in this example)
* Prefixes: Although prefixes aren't used too often, there is the occasional use for them. Let’s look at how we can apply the prefixes of Opera, Safari, Chrome and Firefox (Remember, this is not supported in IE).

<style>

.border {

    border-width: 20px;

    -moz-border-image : url(‘BorderImage.png’) 20 20 20 20 repeat;

    -webKit-border-image: url(‘BorderImage.png’) 20 20 20 20 repeat;

    -o-border-imaqe: url(‘BorderImage.png’) 20 20 20 20 repeat;

    border-image: url(‘BorderImage.png’) 20 20 20 20 repeat;

    }

</style>

* With border-image s, we must set a border-width. If you use equally sized slices you should use the slice size for the width
* Then we applied our prefixes of -moz-, -webkit-, and -o- for Mozilla Firefox, Chrome & Sarafi, and Opera. And lastly, as learned in the prefixes class, we apply the actual declaration last, in case it’s supported now.

|  |  |  |
| --- | --- | --- |
|  |  |  |

**4.27 Box Shadow**

:Another popular idea when creating borders was to shadow certain parts, so it looks like the element is rising above another element. This used to be done with images, but thankfully, we can do it with CSS3 now (It’s faster this way and so much more convenient for us, the developers!)

box-shadow

horizontal-shadow vertical-shadow \*blur \*spread \*color \*inset/outset.

The \*'s above represent optional values

horizontal-shadow is where on the x-axis the shadow should be.

vertical-shadow is where on the y-axis the shadow should be.

\*blur is optional. How much the shadow will be blurred.

\*spread is optional. How much the shadow will be spread out.

\*color is optional. The color of your shadow . Default is black

\*inset/outset is optional .

Inset = shadow is inside the element

Outset = shadow is outside the element Default

Note: The box-shadow property **must** have the horizontal and vertical values.

.box {

    box-shadow: 10px 10px;

    background-color: red;

    }

.box {

    box-shadow: 10px 10px 25px 10px blue;

    background-color:red;

    }

.box {

    box-shadow: 10px 10px 50px;

    background-color:red;

    }

.box {

    box-shadow: 10px 10px 25px 10px blue; background-color:red;

    }

.box {

    box-shadow: 10px 10px 25px 10px;

    background-color:red;

    }

.box {

    box-shadow: 2px 2px 10px 8px blue;

    background-color:red;

    }

**4.28 Background Size**

We touched very quickly on how Twitter does their background images, so that the image does not move with the page. This time, we’re going to cover how they make every background-image spread over the entire page no matter how big or small the image actually’ is.

background-size

widthpx by heightpx, width% by height%, contain, cover

widthpx by heightpx are two values together, like so: 10px 10px;

width% by height% are two percent values together: 10% 10%;

contain will scale the image so that the entire background image will fit inside the element

cover will spread the image so it takes up the entire background of the element

Along with setting a background-image and background-position, we can set it's size.

|  |  |
| --- | --- |
| Width \* Height In Pixels  We can set a background-size to be40px wide and 40px tall, even if the background image is different  <style>  .small {  background-image: url('star.png');  ***background-size: 40px 40px;***  background-repeat: no-repeat;  border :1px solid red;  width:50px;  height: 50px;  }  </style>  Our star png image is now 40px by 40px. The original image is . It also does not take up the entire area of our element . We can use a percent to change it’s size as well. | Width By Height In Percent  Instead of using pixels, we can set the background-size using a percent . Smaller, or larger, both work In this example, we will make the star larger  <style>  .big{  background-image: url('star.png');  ***background-size: 200% 200% ;***  background-repeat: no-repeat;  border :1px solid red;  width: 50px;  height: 50px;  }  </style> |
| Instead of using 100% width and height, we can use contain or cover | |
| Contain will always show the background image, no matter what the size of it's parent element is.  <style>  .contain1{  background-image: url('star.png');  ***background-size: contain ;***  background-repeat: no-repeat;  border :1px solid red;  width: 50px;  height: 50px;  }  </style>  <style>  .contain2{  background-image: url('star.png');  ***background-size: contain ;***  background-repeat: no-repeat;  border :1px solid red;  ***width: 25px;***  height: 50px;  }  </style> | Cover will expand or contract the background image so that it covers the entire element, but will not adjust itself to fit in the element like it did with contain.  <style>  .cover1{  background-image: url('star.png');  ***background-size: cover ;***  background-repeat: no-repeat;  border :1px solid red;  width: 50px;  height: 50px;  }  </style>  <style>  .cover2{  background-image: url('star.png');  ***background-size: cover ;***  background-repeat: no-repeat;  border :1px solid red;  ***width: 25px ;***  height: 50px;  }  </style> |

* In the first example, cover takes up the entire element and the browser will adjust the image appropriately so the width and height stay proportional to each other.
* In the second example, the background image had to be taller to cover the entire element, but because it has to grow proportionally the width of the image is larger too. But because our element is not width enough to fit the background image completely, k is cut off.
* These are useful when you want to create a background image that will fit in the element perfectly. With background-size, there is no need to adjust or edit your image to fit the element, and the images will expand or contract in every HTML5/ CSS3 browser so you know the background-image will appear the same in every browser for every user.
* One thing to look out for, however, is expanding an image When we used a background-size, our ‘star.png’ image became more blurry. That's because the original image is 15x15 pixels and the more you blow up an image the more blurry it will become.

.small {

    background-image: url('star.png');

    background-size: 40px 40px;

    background-repeat: no-repeat;

    border :1px solid red;

    width:50px;

    height: 50px;

}

.big{

    background-image: url('star.png');

    background-size: 200% 200% ;

    background-repeat: no-repeat;

    border :1px solid red;

    width: 50px;

    height: 50px;

}

.contain1{

    background-image: url('star.png');

    background-size: contain ;

    background-repeat: no-repeat;

    border :1px solid red;

    width: 50px;

    height: 50px;

}

.contain2{

    background-image: url('star.png');

    background-size: contain ;

    background-repeat: no-repeat;

    border :1px solid red;

    width: 25px;

    height: 50px;

}

.cover1{

    background-image: url('star.png');

    background-size: cover ;

    background-repeat: no-repeat;

    border :1px solid red;

    width: 50px;

    height: 50px;

}

.cover2{

    background-image: url('star.png');

    background-size: cover ;

    background-repeat: no-repeat;

    border :1px solid red;

    width: 25px ;

    height: 50px;

}

**4.29 Text Shadow**

text-shadow is very much the same as box-shadow. Internet Explorer does not support text-shadow right now. Hopefully in the future Microsoft will cooperate with the rest of the virtual world and start supporting this property.

text-shadow

horizontal-shadow vertical-shadow \*blur \*color

\* represents optional values

horizontal-shadow is how far left/right the shadow will be

vertical-shadow is how far up/down the shadow will be.

\*blur is optional. How much blur to apply.

\*color is optional. What color the shadow should be. Black is default.

To create a text-shadow we need the horizontal and vertical shadow values. Blur and color are optional

|  |  |  |
| --- | --- | --- |
| .shadow1 {  text-shadow:2px 2px;  } | .shadow2 {  text-shadow:2px 2px 10px;  } | .shadow3 {  text-shadow:2px 2px 10px blue;  } |

.shadow1 {

    text-shadow:2px 2px;

}

.shadow2 {

    text-shadow:2px 2px 10px;

}

.shadow3 {

    text-shadow:2px 2px 10px blue;

}

**4.30 Custom Fonts**

We used to be stuck with typical fonts that both the user and the server had. Now, with the benefits of CSS3, we can provide a font, access it, and display it to our viewers. The great part of this, is that our viewers no longer need the font file!

This is a new feature that hasn't been used too often, but allows us to customize our pages in ways that weren't that possible just a few years ago.

@Font-Face

@font-face is how we create a new font-family . And of course, we all know how to use font-family, right? This is how you create a new font-family with a custom font

<style>

@font-face {

font-family: myCustomFamilyName; */\*This is where you name your font \*/*

src: url('myFont.ttf'), url('myFont.otf'), url('myFont.eot');

}

</style>

font-family is the name of our font. We can use any name we like here!

src is a list of sources to try.

***.ttf*** and ***.otf*** are font formats for Opera, Chrome, Safari and Firefox. ***.eot*** is the only font format that IE supports.

Using Your Font

Declaring your new font with @font-face will just tell CSS that you're ready to use a new font. To actually use your font, you have to change the font-family of the *class, id element or attribute* selector of your choice.

IN HERE IS WHERE I AM USING MY CUSTOM FONT THIS IS THE 'OCTIN SPRAYPAINT TTF' FONT I FOUND BY DOING A GOOGLE SEARCH FOR CUSTOM FONTS BELOW IS THE CSS I USED FOR THIS FONT AND THIS PARAGRAPH

<style>

@font-face {

font-family: myName;

src: url('octin\_spraypaint.ttf') ;

}

.font {

font-family: myName;

}

</style>

the .font class has a font-family value of ‘myName’, which matches the font-family in the @font-face above it

This is just the base of the CSS To fool-proof this we need to use, if possible, the .otf and .eot font files and list them in the src property separated by a comma. We should also specify other font-families to choose from in case an older browser cannot read this font.

@font-face {

    font-family: myCustomFamilyName;    /\*This is where you name your font \*/

    src: url('myFont.ttf'), url('myFont.otf'), url('myFont.eot');

}

@font-face {

    font-family: myName;

    src: url('octin\_spraypaint.ttf') ;

}

.font {

    font-family: myName;

}

**4.31 Rotate**

A feature a lot of developers have wanted for a long time is the ability to rotate elements without having to create an image, rotate it, and make the image look like our element.



This is one of those times where you need to use the browser prefixes. Here’s the code I used fix the above element

<style>

.rotate15 {

-ms-transform: rotate (15deg);

-o-transform: rotate(15deg);

-moz-transform: rotate(15deg);

-webkit-transfom: rotate(15deg);

transform: rotate(15deg);

background-color:red;

width:10Opx;

}

</style>

We used the transform property with the value of "rotate(15deg>". Because it’s difficult to write the degree symbol on a keyboard, we just write ”deg”. We tried all 4 browser tapes and followed our rule from before, where we use the actual property after the prefixed properties.

-prefix-transform: rotate (-15deg);

Everything beside your rotated element (all the children) will be rotated as well.

.rotate15 {

    -ms-transform: rotate (15deg);

    -o-transform: rotate(15deg);

    -moz-transform: rotate(15deg);

    -webkit-transfom: rotate(15deg);

     transform: rotate(15deg);

     background-color:red;

     width:10Opx;

    }

**4.32 resize text areas**

You can give Webkit and Firefox users the ability to resize any element. resize is not supported in Opera or IE as of this time. Although, I’m sure this will change.

resize

horizontal, vertical, both, none

horizontal allows the user to expand the element along the x-axis

vertical allows the user to expand the element along the y-axis

both gives the user the ability to expand both the x-axis and y-axis

none removes the ability to resize the element. Default except in <textarea>

|  |  |  |
| --- | --- | --- |
| .horizontal{  resize: horizontal;  overflow: auto;  } | .vertical {  resize: vertical;  overflow: auto;  } | .both {  resize: both;  overflow: auto;  } |

.horizontal{

    resize: horizontal;

    overflow: auto;

}

.vertical {

    resize: vertical;

    overflow: auto;

}

.both {

    resize: both;

    overflow: auto;

}