college CD

Web design and development 1 (DW1 / CW1)

Class 3

The ** ** tag is a neutral tag. A simple container. It is used in a HTML document to regroup elements. Before HTML5 (to be covered at a later time), it was used to divide the different sections of the page.

Being a ***block*** type, ** ** automatically jumps a line (like <p>) and uses 100% width by default. This, of course, can be modified.

```
<div>
Header
</div>
<div>
Main content
</div>
<p>
Footer
</div>
```


The tag **** is also a container, but is type is ***inline***, meaning that is doesn't jump a line and its dimension if the one of its content. It is used for styling element on a signle line (like <b>).

```
<p>
Element of a <span>section</span> of a page.
</p>
```

« float » property

Since ** ** is a block, it is impossible to place to of them next to each other whatever their sizes. Do do so, **float=" "** can be use to position them left or right.

If ***float: left*** is applied to a container, it will be positioned on the left and following elements will be placed next to it, at its right, and so on.

Same thing if ***float: left*** is used, the container will be positioned on the right and following elements will be placed next to it, at its left, and so on

```
<div style="float: left;">
DIV content
</div>
<div style="float: right;">
DIV content
</div>
```

« clear » property

To stop the impact of a ***clear*** attribute to be applied from a certain point, the clear attribute must be used alogn with a value *(**left**, **right**, or **both**)*.

```
<div style="clear: both;">
Div content
</div>
```

properties

width:	*[px, %, etc.]*	border-style:	*[solid,dotted, double, etc.]*
height:	*[px, %, etc.]*	border-width:	*[px, %, etc.]*
background-color:	*[color, hex, etc.]*	border-color:	*[color, hex, etc.]*
padding:	*[px, etc.]*		
margin:	*[px, etc.]*		

Positioning using CSS

Every HTML element can basically have four types of positioning (we'll see more types at a later time):

- static (default)
- relative
- absolute
- fixed

Position types

position: static
Static is the default position of all element. It is just like the usual HTML positioning which shows on screen elements in their coding order. What's coded first appears first on the screen.

```
div {
    position: static;
}
```

position: relative
Relative position is a lot like static, although it aloows you to specify positioning using properties (top, left,etc.).

It also allows to position absolute element according to this very container instead of according to the browser's window.

```
div {
position: relative;
top: 25px;
}
```

position: absolute
Absolute positioning allows you to position element using coordinates. The coding order has absolutly no importance here.

For instance, you can tell a container to be positioned 20% from the bottom and 50px from the right.

The positioning of an element is relative to the browser's window unless the container it is nested in has a relative positioning; it is then positioned relatively to its parent container.

```
div {
    position: absolute;
    bottom: 20%;
    right: 50px;
}
```

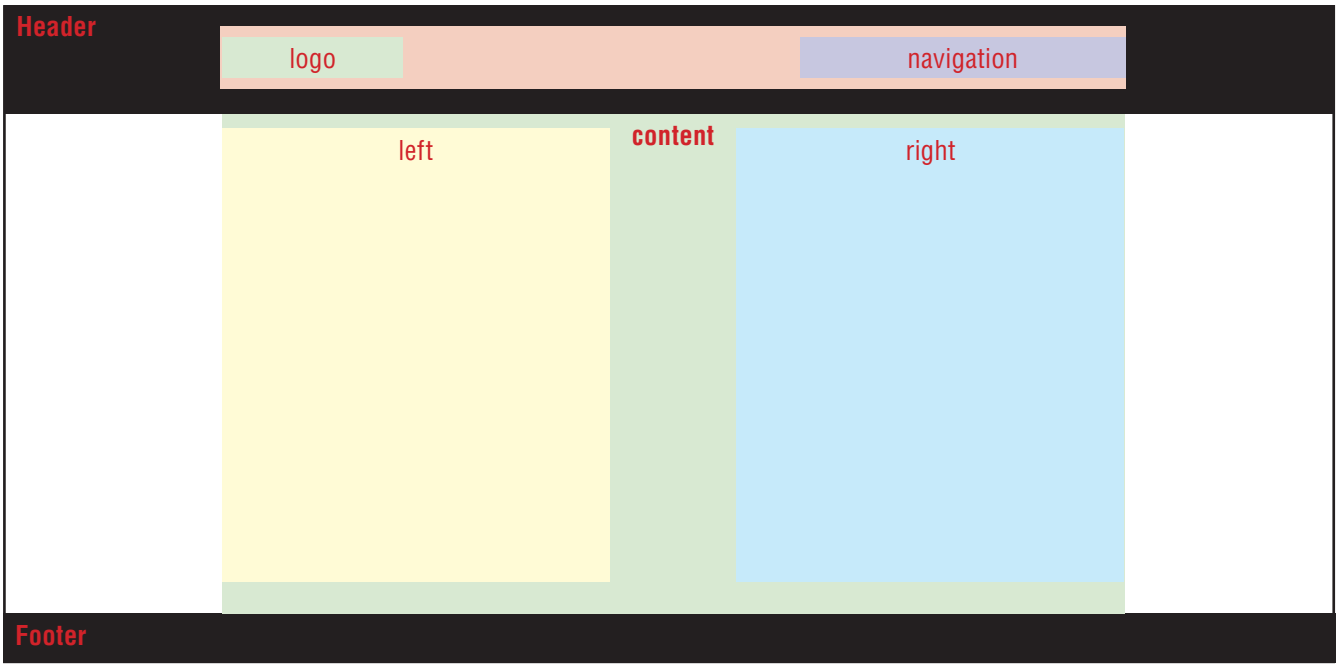
position: fixed
Position fixed is just like absolute positioning, although it remain in place, it doesn't move if the user scrolls down.

```
div {
position: fixed;
top: 0px;
width: 100%;
height: 80px;
}
```

Nesting containers

Tags are considered containers and CSS allows us to style their contents. But CSS also allows us to control the containers position in HTML documents. Containers can then be used to create layout grids by nesting containers and specifying their positions.

Here is an example of a layout based on nested containers :



header
We want to create a black header section which won't move when scrolling with a height of 100px and a full width.

footer
The footer would just be like the header, but 50px high, a fixed position and we would replace *top: 0px* with ***bottom: 0px***.

content
In the section content, we want to limit content to a 750px container centered horixontally. **left / right.**

Also, this section would contain two containers positioned left and right using the ***float*** property.

```
.header {
    position: fixed;
    top: 0px;
    width: 100%;
    height: 100px;
    background-color: black;
}

.content {
    position: relative;
    width: 750px;
    margin: auto;
}

.left {
    position: relative;
    width: 40%;
    float: left;
}
```

RollOvers

It is impossible to create rollover effects in HTML unless you use JavaScript. CSS makes it very easy.

Pseudo-class *:hover*

:**hover** allows you to specify alternate styles when the user's mouse pointer rolls over an element.

Usually applied to hyperlinks, this pseudo-class can be applied to any selector.

First declare perperties to a selector and then create the alternate properties using the pseudo-class. Pour appliquer **hover** à une **class**, procédez de la même manière. Créez la class nécessaire, puis créez-en une version survolée.

Il est possible d'affecter toutes les propriétés à un **hover** : couleur du texte, arrière-plan, dimensions, bordures, police de caractères, marges internes et externes, positionnement, etc.

```
div {
    color: black;
}

div:hover {
    color:red;
}


.element {
    color: black;
}

.element:hover {
    color:red;
}
```

Exclusive class

It is possible to create classes to be used only by a specific selector. This allows you to use the same term for different selectors.

All you need to do is to write the selectorthen the class (using a period, of course).

In the example beside, event if ***.center*** would be applied to different tags, only the contents placed in between ** ** using this class would be centered.

```
p.center {
    text-align: center;
}

<p class="center">
    Content would be centered
</p>

<div class="center">
    Content would not be centered
</p>
```