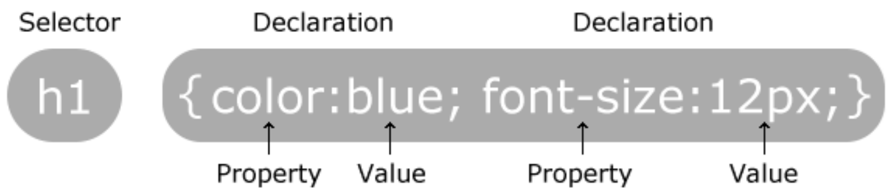
CSS (Cascading Style Sheet)

HTML describes the content of the html document, *what* exactly should be presented. CSS is used to define styles for the web pages, *how* the elements should be presented: the appearance of the elements, design, layout and variations in display for different devices and screen sizes.

CSS SYNTAX

CSS RULE:

A CSS rule set consists of a selector and a declaration block:



- For each selector there are "properties" inside curly brackets
- Semi-colons separate the properties
- A value is given to the property following a colon

CSS COMMENTS

A CSS comment starts with /* and ends with */. Comments are ignored by browsers.
Comments are used to explain your code, or edit the source code at a later date.
Example:

```
p {
   color: red;
   /* This is a single-line comment */
 }
```

THE ELEMENT SELECTORS

The element selectors are based on the element name (you know them as tags in HTML):
Some element selectors: html, body, h1 to h6, p, a, ol, ul, img, hr, strong, em, and more.

Example of p (paragraph) element selector: `p {color: red; text-align: center;}`

To make the CSS code more readable, you should put one declaration on each line:

Paragraph selector:

```
p {
   color: red;
   text-align: center;
   padding:20px;
   margin-left: 5px;
  /*can be more*/
}
```

Body selector:

```
body {
   font-size: 14px;
   color: navy;
   background: white;
   margin:0;
}
```

Hyperlink selector:

```
a:link, a:active, a:visited {
            font-family: Arial,
            font-size: 14px;
            color: #3e9479;
            text-decoration: none;
}
a:hover{
            color: #107d6f;
}
```

Class and Id selectors: If you want to apply different styles to the same types of elements, you have to differentiate those elements by giving them different names. For example, you have several paragraphs, but you want to style some of them differently; for that you have to use class or id selectors with different names.

CLASS SELECTOR

You can use class selector in the same html document many times.
The class selector starts with a dot (class name cannot start with a number).

.name

Example:
In CSS:

.colorRed {color: red;}
and in HTML:
<div class="colorRed">Text in this div is red**</div>**

Example of multiple usages:
In CSS:

.center {margin:auto;}

and in HTML:

<hl class="center">This heading is center-aligned**</hl>**
<p class="center">This paragraph is also center-aligned**</p>**

Example of paragraphs with different text alignments in the same document
In CSS:

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

and in HTML:

<p class="right">The text in this paragraph is right-aligned**</p>**
<p class="center">The text in this paragraph is center-aligned**</p>**

You can apply more than one class per given element:
In CSS:

.center {text-align: center;}
.red {color: right;}

and in HTML: **<p class="center red">**The text in this paragraph is centered and red**</p>**

ID SELECTOR

You should use id selector in an html document just once; usually it is used for any kind of box except for HTML5 semantic elements.
The id selector starts with a pound sign symbol. (Id name cannot start with a number)

#name

Example:
In CSS:

```
#wrapper {
width: 960px;
height:800px;
margin: auto;
}
```

and in HTML:

```
<div id="wrapper">
        Content of this box
</div>
```

This box has 960 px width, 800 px height, and positioned in the middle of the browser horizontally

CLASS AND ID SELECTORS TOGETHER

You can apply the id and the class to the same element, for example:

In CSS:

```
#wrapper {
width: 960px;
height:800px;
margin: auto;
}
.red {color:right;}
```

In HTML:

```
<div id="wrapper" class="red">
Content of this box
</div>
```

This box has 960 px width, 800 px height, positioned in the middle of the browser horizontally, and text inside is red.

MORE USEFUL SELECTORS

* (asterisk) - selects all elements in CSS

Example: ***{box-sizing: border-box;}**

, (comma) serves to list the elements, which will be selected together

Example: body, div, span and iframe will have margin 0, padding 20 pixels and border 0

```
body, div, span, iframe {
        margin: 0;
        padding: 20px;
        border: 0;
        }
```

element:nth-child(2) – selects the 2nd element

Example: text in the second paragraph will be red

```
p:nth-child(2) {
   background: red;
}
```

element:nth-child(2n+1) - selects every second element (starting with the first one)

Example: text in each second paragraph will be red

```
p:nth-child(2n+1) {
   background: red;
}
```

To start with the second element, it has to be **element:nth-child(2n+0)**

Note: you can use any number, for example: text in each second paragraph will be red

```
p:nth-child(3n+1) {
   background: red;
}
```

There are many more advanced selectors (not for this class)

+ (plus) selects adjacent/next-sibling elements,

> (greater-than sign) selects all siblings of a direct parent,

~ (tilde) selects the following-siblings preceded by parent, etc.

DESCENDANT CSS SELECTORS

The first element in the nested elements' structure represents the ancestor element; it is a structurally superior element (a parent element). The second element represents the descendant element in relation to the first element (a child element)**.** The third element is the descendant element in relation to the second element; meanwhile the second element becomes the parent of the third element, etc.

In the following example with lists:
<ul> is a structurally superior element, that is the ancestor element (parent element)
The first-level <li> is a descendant element (child) of the <ul>, but parent to <ol>
<ol> is descendant element (child) of the first-level <li>, but parent to the second-level <li>
The second-level <li> is a descendant element (child) of the <ol>

Example: we want to style only <ol>

CSS

```
ul li ol {
 text-transform: uppercase;
}
```

HTML

```
<ul>
  <li>Birds</li>
  <li>Animals</li>
     <ol>
        <li>Cat</li>
        <li>Dog</li>
     </ol>
  </li>
</ul>
```

RESULT:

- **Birds**
- **Animals**
  1. CAT
  2. DOG

Words CAT and DOG became uppercased (and different from the words Birds and Animals) because we gave them a certain structure here: <ul> is the parent to the first <li> which is the parent to <ol>; and stylized only this <ol> - the last in the row.

Note: to understand the order easier, you can think about the line ul li ol {…} from the end, like this:
ol** of **li** of **ul

Example: two or more descendants of the same tyle
If there are two or more the same descendant elements, like two <li> in this example, then in the following CSS, all <li> will obey the same declaration, because each of them has the <ul> ancestor element.

CSS

```
ul li {
text-transform: uppercase;
}
```

HTML

```
<ul>
  <li>Birds</li>
  <li>Animals</li>
     <ol>
        <li>Cat</li>
        <li>Dog</li>
     </ol>
  </li>
</ul>
```

RESULT:

- **BIRDS**
- **ANIMALS**
  1. CAT
  2. DOG

Words BIRDS, ANIMALS, CAT, DOG will look identical (uppercased in this case).

THREE TYPES OF CSS

There are three ways to insert CSS:

IN-LINE

In-line styles are inserted directly into an HTML tag to style an element that is inside that tag.

Example:

<span style="color: #cccccc;">**Some text**</span>

Note: the element <span> is used to style text.

INTERNAL

Internal styles are used for one page; they are located in the <style> tags inside the head section of this page. These styles are referred and applied to the elements that are inside the body of the page.

For example, if you want a paragraph text to be red and its font size 20 pixels, the code should be:

```
<head>
   <title>Page Name</title>
      <style>
           p {
            color: red;
            font-size:20px;
           }
      </style>
</head>

<body>
<p>This paragraph text will be red with the font size 20 pixels </p>
<p>This paragraph text will also be red with the font size 20 pixels </p>
</body>
```

EXTERNAL

With an external style sheet, you can change the look of an entire website by changing just one file!

External styles are used for the whole, multiple-page website. There is a *separate CSS file*, to which many HTML pages are linked. This file has an extension .css
For example, you created a file named "mystyle.css" in the same directory as an HTML page

In your HTML page, in the <head> section, the link to CSS page should look like this:

```
<head>
   <title>Page Name</title>
   <link rel="stylesheet" href="style.css" > (note: order of parts does not matter here)
</head>
```

Then in this external CSS file "mystyle.css" you put some code there, say for a paragraph:

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

All paragraph texts in all HTML pages connected to "mystyle.css" file will be red.

HTML5

HTML5 is a markup language used for structuring and presenting content on the internet. It is the current version of the HTML standard, with new elements, attributes, and behaviors. HTML5 groups your code, groups your content, create cleaner markup and bring the advanced features you would normally have to write Javascript to do. HTML5 allows you to draw on a canvas, play a video, design better forms, or build web applications that work offline.

For HTML 5 documents doctype and character encoding are changed:
DOCTYPE is <!DOCTYPE html>
Character Encoding is <meta charset="UTF-8"> (for characters in the content to be correctly interpreted)

HTML5 is semantic; it means that it is written in a linguistic way, which clearly describes the meaning of the encoded information to both, the browser and the developer.

NEW HTML5 TAGS

For the new tags, because they are unique, you do not have to define class or id (unless there are several of them for some reason in one document).

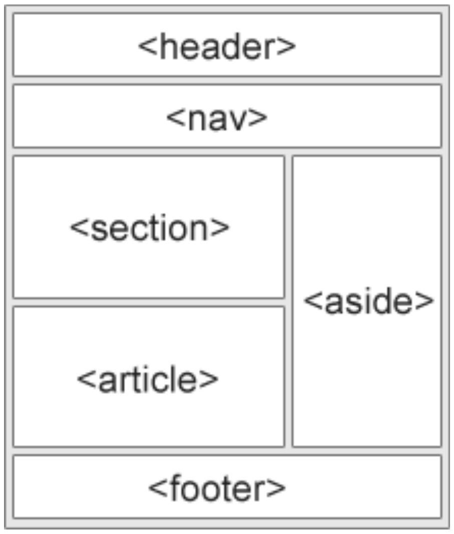
<header> Defines a header for the document or a section;	<nav> Defines navigation links;
<section> Defines a section in the document;	<main> Defines the main content;
<aside> Defines content aside from the page content;	<article> Defines an article or a story;
<figure> Defines content as illustrations, diagrams, photos;	<footer> Defines a footer content;

…. and much more.

HTML5 DOCUMENT STRUCTURAL EXAMPLE:

```
<body>
<header>
<!--header content goes in here -->
</header>
<nav>
<!--navigation menu goes in here -->
</nav>
<section>
    <article>
  <!--some article, story or text goes in here -->
    </article>
</section>
    <aside>
  <!--aside content goes in here -->
    </aside>
<footer>
  <!--footer content goes in here -->
</footer>
</body>
```

STRUCTURAL DIAGRAM EXAMPLE:



Many of the <div> elements have been replaced by new HTML5 layout elements, but you can still use <div> elements when defining specific values for the id or class attributes

STYLING HYPERLINKS
(This example uses internal CSS)

GENERAL LINKS FOR A PAGE:
CSS

```
    a {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 11px;
        color:#333333;
        text-decoration: none;
    }
    a:hover {
        color:#DB3A25;
    }
```

HTML in the body:

```
    <a href="http://www.somesite.com">This is an interesting site</a>
```

Note: In the above case, styling for all three: link, visited link, active link is the same; therefore, **a { }** is written just once.
It can also be written: **a:link, a:visited, a:active { }**

Very important: there must be no space from the left and right of the column

DIFFERENT LINK STYLE:
If you want to create links with different appearance for some different link types (say, for your email), then you have to give them some name, so CSS would not be confused.
For example, the name is anotherStyle

```
    a. anotherStyle:link, a.anotherStyle:visited, a.anotherStyle:active {
        font-family: 'prstregular';
        font-size: 20px;
        color: #999999;
        text-decoration: none;
    }
    a.anotherStyle:hover{
    color:#CCCCCC;
    }
```

HTML in the body:

```
    <a class="anotherStyle"  href="mailto:myemail@gmail.com"> Send me email</a>
```

NAVIGATION: Box method vs. List methods

BOX METHOD:

CSS:

```
#box{
        width: 400px;
        height: 50px;
        background-color: #CCC;
        margin-top: 50px;
}

.text{
        margin-left: 50px;
        margin-top: 10px;
        display: inline-block;
}
```

HTML:

```
<div id="box">

  <span class="text">
    <a href="index.html">HOME</a>
  </span>

  <span class="text">
    <a href="works.html">PORTFOLIO</a>
  </span>

  <span class="text">
    <a href="about.html">ABOUT</a>
  </span>

</div>
```

LIST METHOD:

CSS:

```
*{
/*CSS Reset: to reset it to reduce browser inconsistencies */
        margin: 0;
        padding: 0;
}

ul {
        background: pink;
        height: 50px;
        width: 400px;
        margin-top: 50px;
}

li {
        display: inline-block;
        margin-top: 10px;
        margin-left: 50px;
}
```

HTML:

```
<ul>
<li><a href="index.html">HOME</a><li>
<li><a href="works.html">PORTFOLIO</a></li>
<li><a href="about.html">ABOUT</a></li>
</ul>
```

Results of both methods: HOME   PORTFOLIO   ABOUT

Notes:
1. Why do we need here this CSS rule display: inline-block;?
This rule makes the text to stay in one line – inline part, and in the same time to keep margins (and more box attributes if needed) – block part
2. List method is preferable – more professional and good for SEO (Search Engine Optimization)

GOOGLE FONTS

Now, you have the freedom to use almost any font as a web font, giving you endless typographical creativity and helping improve your site's design. In addition, Google fonts allow using fonts that are not installed on the user's computer.

Google Web Fonts is a completely free and super easy way to implement a variety of fonts on your website in a properly licensed and widely supported fashion.

https://www.google.com/fonts

Thorough instructions of how to use this site: https://tinyurl.com/zthu95b

Thorough video instructions of how to use this site: https://tinyurl.com/j6rx3hf

CSS @FONT-FACE RULE

If you want to use a highly unique font and cannot find it on Google Fonts site, you should use @font-face rule.

Webfont Generator https://www.fontsquirrel.com/tools/webfont-generator

1. Find the font that you like, and upload it to Webfont Generator (which is very easy to use). It will convert your font into the webfont kit, which you then download and unzip.
2. You will find there several font files and a CSS file with the ready-made *@font-face* code inside of it.
3. Put the font files into the folder with your site. Copy the code from the given CSS file and put it into your CSS file
4. When you upload your site to your server, **it is very important not to forget to include and upload all the font files.**

Example: Let’s say, you have (of found) the font 'azbuka04regular'.

STEP 1: Upload the font to Webfont Generator, get your zipped file, unzip it, and put all webfont files into your site folder.

STEP2: In a downloaded CSS file you will find this (or similar) code, copy it and paste into your CSS file

```
@font-face {
font-family: 'azbuka04regular';
src: url('azbuka04-webfont.eot');
src: url('azbuka04-webfont.eot?#iefix') format('embedded-opentype'),
url('azbuka04-webfont.woff') format('woff'),
url('azbuka04-webfont.ttf') format('truetype'),
url('azbuka04-webfont.svg#azbuka04regular') format('svg');
font-weight: bold;
font-style: normal;
}
```

STEP 3: Still in CSS part, under the above code, create the name for this font (e.g. myFont) and write:

```
. myFont{
        font-family: 'azbuka04regular';
}
```

STEP 4: In the HTML document, write the following code:

<span class="myFont"> Here goes any text**</span>**

It will find in CSS file how “myFont” looks, and the phrase "**Here goes any text**" will have font-family: 'azbuka04regular';

CSS DISPLAY - BLOCK and INLINE ELEMENTS

BLOCK elements are elements that have a line break before and after them.
You can assign width, height, margin and more parameters to them.
Examples of block elements: **h1, p, ul, ol, li, div, table**, and more

INLINE elements are positioning next to each other.
They naturally do not have width and height parameters; they only take up as much width as necessary.
An inline element can have only left and right padding and margins.
Examples of inline elements: **span, links**, also **text** and **images**, and more

CONVERTING ELEMENTS

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way

1. Block elements have width and height, but when a block element is declared to be an inline one, it loses its ability to have width and height. Therefore, if width and height is needed, then the code for block elements has to be **"display: inline-block;"**
2. Inline elements do not have width and height. Therefore, if width and height is needed, the code is **"display: block;"**. In the case you want this element still remaining inline, then the code for block elements has to be **"display: inline-block;"**

CONVERTING BLOCK ELEMENT INTO INLINE ELEMENT

Example1. Paragraph

```
<p style="display: inline;">some text</p>
```

The second text part or any inline element in this case will be displayed next to “some text” despite the paragraph tag.

Example2. Displaying a list of links as a horizontal menu

CSS:

```
<style>
li {
   display: inline;
}
</style>
```

HTML:

```
<ul>
<li><a href="http://google.com">Google </a></li>
<li><a href="http://msn.com">MSN </a></li>
<li><a href="http://yahoo.com">Yahoo </a></li>
</ul>
```

This menu will appear as: **Google** **MSN** **Yahoo** despite that lists are block elements

CONVERTING INLINE ELEMENT INTO BLOCK ELEMENT

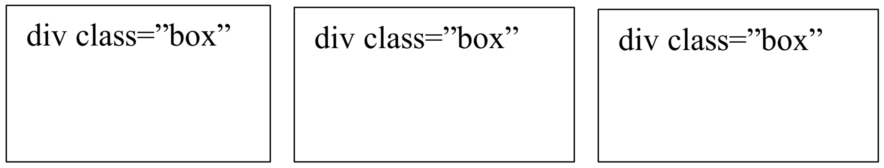
```
.text{
   display: block;
   width: 200px;
   height: 100px;
   padding: 10px;
}
```

It will make the text part to behave as block element, so you can assign to it width, height, margins, etc. like to an any box

DISPLAY INLINE-BLOCK RULE

Displayed inline-block elements behave like inline elements, but they can have width, height and other block's attributes.
Example: these boxes have to be aligned (inline) and at the same time have width, height and margin (block)

```
.box {
  display: inline-block;
  vertical-align: top;
  width: 150px;
  height: 100px;
  margin: 5px;
}
```



DISPLAY: NONE

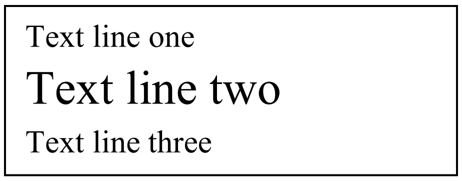
Hiding an element can be done by setting the display property to none. The element will be hidden, and the page will be displayed as if the element is not there:

```
h1{
   display: none;
}
```

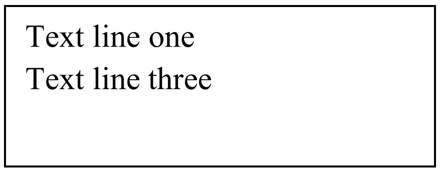
HTML part:

```
Text line one
<h1>Text line two</h1>
Text line three
```

Before applying dispay:none code:


Text line one
Text line two
Text line three

Result after applying dispay:none code:


Text line one
Text line three

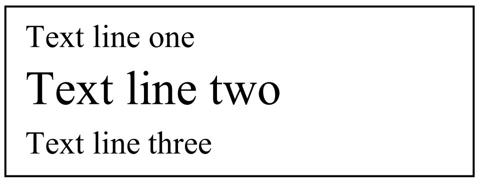
VISIBILITY: HIDDEN

The code visibility:hidden; also hides an element. However, the difference from display:none is: even being invisible, it still takes up the same space as before; therefore, other elements do not take its place.

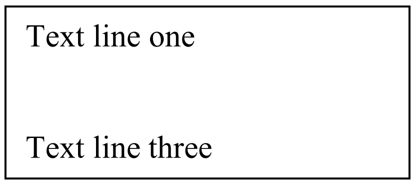
```
h1 {
   visibility: hidden;
}
```

(HTML part is the same as above)

Before applying visibility:hidden code:


Text line one
Text line two
Text line three

Result after applying visibility:hidden code:


Text line one

Text line three

DISPLAY: INLINE-BLOCK - WIDTH AND HEIGHT

1. Block elements have width and height.
When a block element is declared to be an inline one, it loses its ability to have width and height.
Therefore, if width and height is needed, then the code for block elements has to be "display: inline-block;"

Box 1

Box 2

CSS:
```
div{
  width:100px;
  height:100px;
  background-color: beige;
  display:inline-block;
    }
```

HTML:
```
<div>
BOX 1
</div>
<div>
BOX 2
</div>
```

2. Inline elements do not have height. Therefore, if width and height is needed, but you want this element remain inline, then the code for block elements has to be "display: inline-block;"

This is the first sentence in a span with width and height

This is the second sentence in a span, next to the first one

CSS:
```
span{
  display:inline-block;
  width:200px;
  height:40px;
  background-color: beige;
   }
```

HTML:
```
<span> This is the first sentence in a span with width and height </span>
<span> This is the second sentence in a span, next to the first one </span>
```

INLINE ELEMENTS - MARGINS

1. Images, which are inline elements, can have all margins because they have width and height naturally.

CSS:
```
img{
  margin-left:50px;
  margin-bottom:40px;
     }
```

HTML:
```
<img src="image1.com" />
<img src="image2.com" />
```

2. Text, which is inline element, can only have margins left and right, not top and bottom, because it has only width of the line.

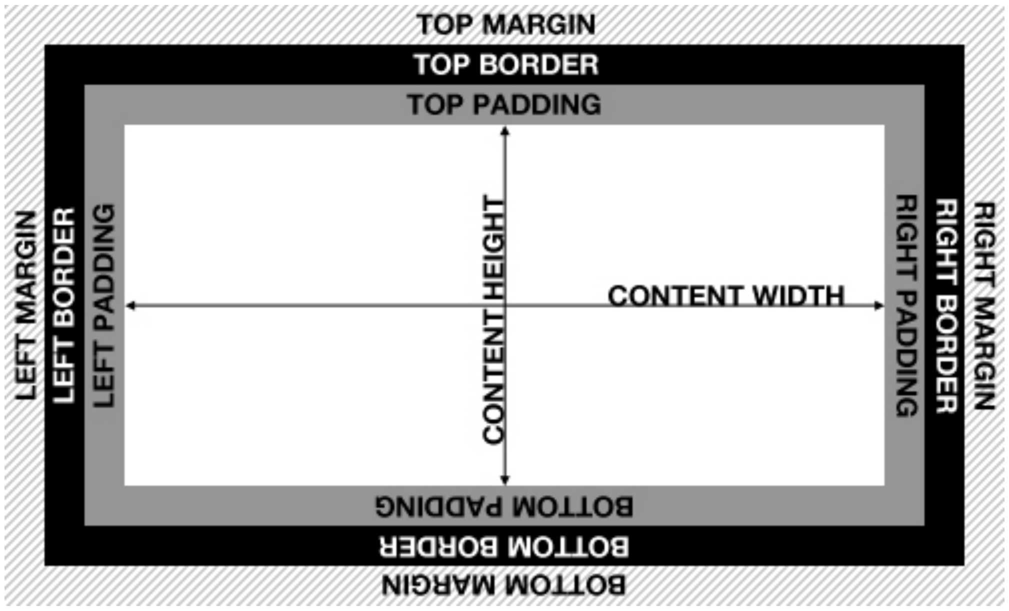
This is the red text with left margin 200px

CSS:
```
span{
margin-left:200px;
color: red;
   }
```

HTML:
```
<span>This is the red text with left margin 200px </span>
```

ELEMENT BOX MODEL

Margins, paddings, borders, content



1. MARGIN

Margins properties clear an area around the border (thus, pushing the element to specified direction). The margin does not have a background color, and it is completely transparent

Example:

margin-top: 100px;
margin-bottom: 200px;
margin-right: 50px;
margin-left: 30px;

Shorthand writing of properties

Note: the order is important: top, right, bottom, left

From above example

margin: l00px 50px 200px 30px;

Values for margins

- Length specifies a margin in units: em*, px, pt, cm, etc. (default value is 0px)
- With the margin: auto of an element, the space around it is calculated, so it will be horizontally positioned in the middle of the browser (or of the outer element in which it is located)
- % specifies a margin in percent of the width of the containing element

Important note: *that elements has to have width for margin code to work*

* *An em unit is not rigid; it is relative to the current font size of the document. For example, if the font size is 16 px (the web usual default), then it is 1 em (or 100%); if in this document you want a font size to be equal 12 px, it will be 0.75 em (or 75%); or if in this document you want a font size to be equal 24 px, it will be 1.5 em (or 150%), etc. (There are calculators on the internet).*

2. BORDER

Borders are lines around the element content

Values for borders: width, size, style and color

Important note*: none of the border properties will have ANY effect unless the border-style property is set!*

Shorthand example for border width, style and color:

```
p {
   border: 5px solid red;
}
```

3. PADDING

Padding properties define the space between the element border and the element content.

Value for paddings and padding sides:

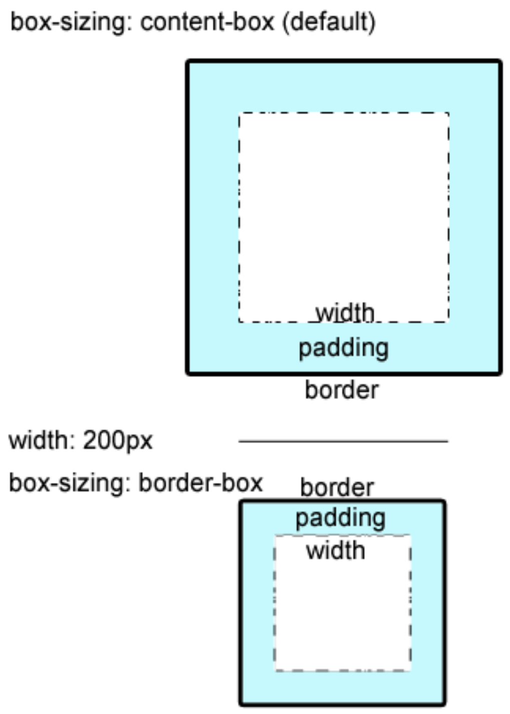
The same as margins

4. CONTENT is the area in the box where text and images are placed

Important about box sizing: When you set the width and height properties of an element with CSS, you just set the width and height of the content area. To calculate the full size of an element (how much space it really takes), you must also add padding and borders to the content area (plus consider margins if needed).

Example

```
div {
    width: 200px;
    padding: 40px;
    border: 5px solid gray;
   }
```



Calculation: total element width is 290 pixels:
200px (content area width)
+ 80px (left + right padding)
+ 10px (left + right border)
= 290px

Solution: The box-sizing property can help this situation by resetting the whole box size, as the content area size (of course the content area becomes smaller).

```
* {
  box-sizing: border-box;
}
```

In this case the width of the above example will be 200 pixels.

Note: Universal Box Sizing

This method selected pseudo elements before and after as well,

```
*, *:before, *:after {
  box-sizing: border-box;
}
```

Note 2: {box-sizing: padding-box;} With this property width and height values apply to the element's content and its padding, but not borders. As of today, only Firefox implemented this value.

EXAMPLES OF BOX MODEL PROPERTIES

PADDING

Without padding

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam non purus sit amet nulla mollis volutpat elementum quis massa. Nunc ut tincidunt leo, nec vehicula lorem. Nullam dolor lacus, sodales ac metus at, imperdiet egestas tortor. Duis dignissim, sem id sodales dignissim, orci nibh sollicitudin eros, a scelerisque felis lorem a tellus.

With 10 pixel top and 20 pixel left padding

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam non purus sit amet nulla mollis volutpat elementum quis massa. Nunc ut tincidunt leo, nec vehicula lorem. Nullam dolor lacus, sodales ac metus at, imperdiet egestas tortor. Duis dignissim, sem id sodales dignissim, orci nibh sollicitudin eros, a scelerisque felis lorem a tellus.

MARGIN

Example 1:

This box has no margins

This box has no margins

Example 2:

This box has bottom margin 10 pixels

This box has no margins

Example 3:

This box has bottom margin 10 pixels

This box has top margin 10 pixels

Note: the same approach with horizontal margins

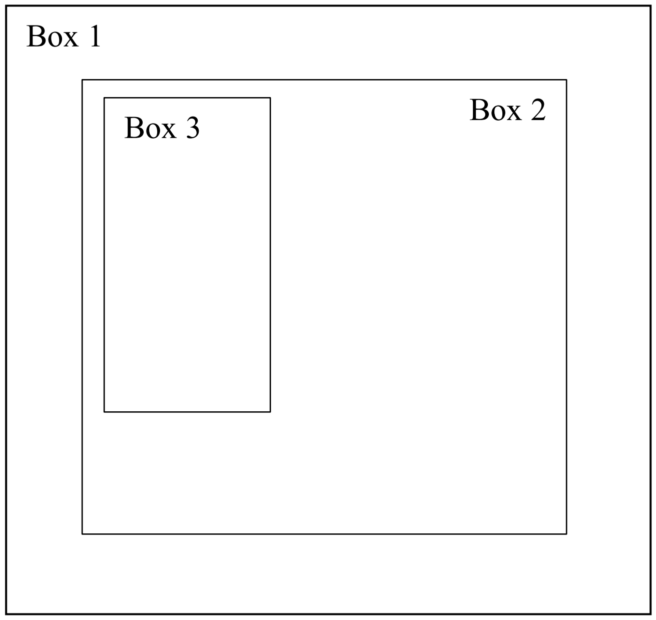
NESTING BOXES

Box 1 has 300 pixel width, 300 pixel height,
Box 2 has 200 pixel width, 200 pixel height,
Box 3 has 80 pixel width, 150 pixel height,

Code variant 1:
Box 1 has 50 pixel padding from all sides.
Box 2 has 10 pixel top and left padding.

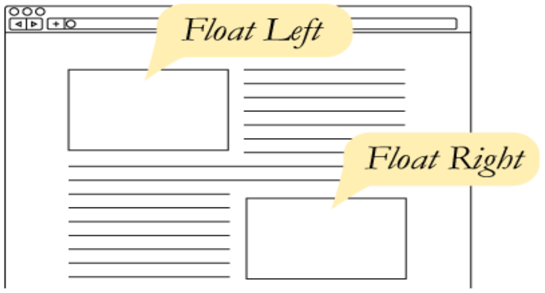
Code variant 2:
Box 2 has 50 pixel margin from all sides.
Box 3 has 10 pixel top and left margin.

There are much more different ways and combinations available



FLOAT

Floating is often used to push an image to one side or another, while having the text of a paragraph wrap around it.

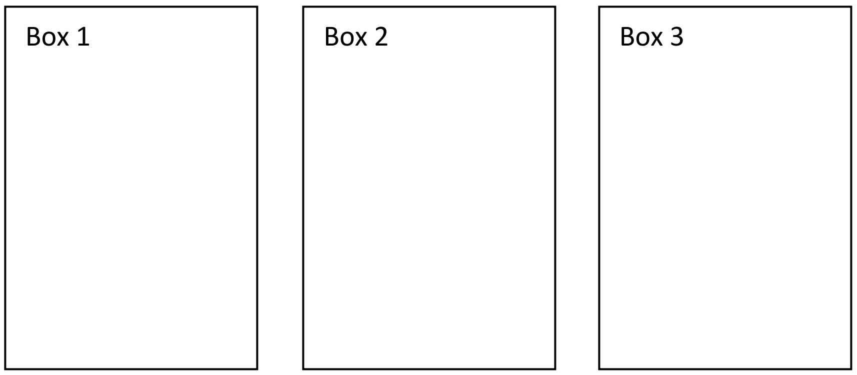


CSS code
```
img.floatLeft {
   float: left;
   margin: 4px;
}
img.floatRight {
   float: right;
   margin: 4px;
}
```

HTML code
```
<img src="pict1.jpg" class="floatLeft">
<p>Text text  text  text … </p>

<img src="pict2.jpg" class="floatRight">
<p>Text text  text  text … </p>
```

FLOAT OPTIONS CAN ALSO BE USED TO POSITION BOXES:
3 boxes floated left:
The second to the first one; the third to the second one

CSS code
```
<style>
 .left {
float:left;
width:100px;
height:200px;
margin-right:20px;
}
</style>
```



HTML code
```
<div class="left">Box 1</div>
<div class="left">Box 2</div>
<div class="left">Box 3</div>
```

TURNING OFF FLOAT - USING CLEAR PROPERTY

Elements after the floating element will flow around it. To avoid this, use the clear property.
The clear property specifies which sides of an element other floating elements are not allowed.

Example 1, inline CSS:
```
<div style="clear: both;"> </div>
```

Example 2, internal CSS:
```
.someText {
   clear: both;
}
```

CLEAR VALUES:

None	Default. Allows floating elements on both sides
Left	No floating elements allowed on the left side
Right	No floating elements allowed on the right side
Both	No floating elements allowed on either side
And more …	

CSS LISTS

CSS code:

```
<style>
ul {
   list-style-type: disk;
}
</style>
```

HTML code:

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
```

Result:

- Coffee
- Tea
- Coca Cola

EXAMPLE OF UNORDERED LISTS OF DIFFERENT STYLES:

CSS code:

```
<style>
ul.a {list-style-type: circle;}
ul.b {list-style-type: square;}
</style>
```

HTML code:

```
<ul class="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

<ul class="b">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
```

Result:

```
○   Coffee
○   Tea
○   Coca Cola

▪   Coffee
▪   Tea
▪   Coca Cola
```

EXAMPLE OF ORDERED LISTS OF DIFFERENT STYLES:

CSS Code

```
<style type>
ol.c {list-style-type: upper-roman;}
ol.d {list-style-type: lower-alpha;}
</style>
```

HTML code:

```
<ol class="c">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ol class="d">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>
```

Result:

```
  I.   Coffee
 II.   Tea
III.   Coca Cola

a.  Coffee
b.  Tea
c.  Coca Cola
```

CODE FOR USING AN IMAGE IN LISTS:

```
ul {list-style-image: url('sqpurple.gif');}
```

Note: if you place different kinds of lists on one page, they have to be defined by different class

POSITION PROPERTY

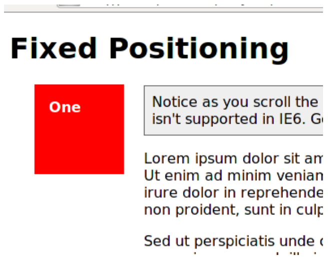
STATICALLY positioned elements
Static is the default position of the elements; in most cases you don't have to declare it

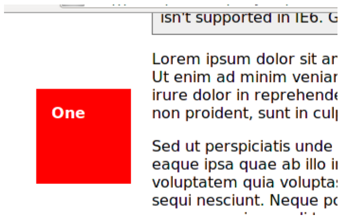
FIXED positioned elements
The element with "position: fixed" is positioned relatively to the viewport; it will not move when the document is scrolled, it will always be in viewpoint. The document and other elements behave like the fixed positioned element does not exist. The top, right, bottom, and left properties are used to position the element.

Before scrolling After scrolling - the square is still on the same position

CSS example:

```
#button {
position:fixed;
top:30px;
left:5px;
}
```

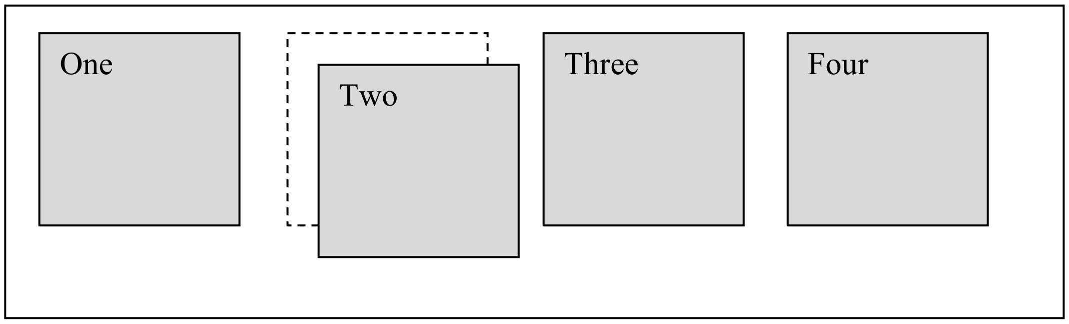




RELATIVELY positioned elements
You can offset the position of the element <u>from its original spot</u> with top, left, bottom, or right. Offsetting does not affect the flow of the elements around it, they respect the space originally occupied by the moved element.

CSS example:

```
#box2 {
   width: 100px;
   height: 100px;
   background: red;
   position: relative;
   top: 20px;
   left: 20px;
}
```



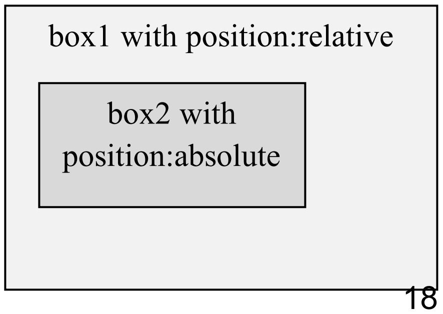
ABSOLUTELY positioned elements; and **RELATIONSHIP** between <u>absolutely and relatively</u> positioned elements
The element with "position:absolute" is not positioned relatively to its original spot.
Absolutely positioned element – child is located inside an outer container – parent, which is called Relatively positioned element. Child becomes absolutely positioned in relation to its parent's top, left, bottom, or right sides. If there is no outer container, then the browser becomes the parent.

CSS .

```
#box1 {                  #box2 {
width: 200px;            width: 100px;
height: 150px;           height: 60px;
position:relative;       position: absolute;
}                        top: 50px;
                         right: 20px;
                         }
```

HTML

```
<div id="box1>

<div id="box2>
</div>

</div>
```

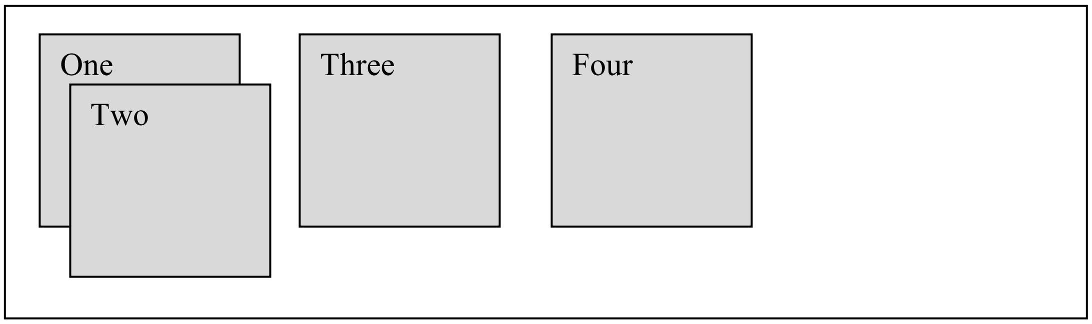


DIFFERENCE REGARDING THE SPACE ORIGINALLY OCCUPIED BY THE MOVED ELEMENT.

Another important difference is the fact that an element set to "position: absolute" is completely removed from the document's normal flow. This means the space originally occupied by the element will not be honored by surrounding elements. Surrounding elements will behave as if the absolutely positioned element is not even there.

CSS example:

box2 {
width: 100px;
height: 100px;
position: absolute;
left: 10px;
top: 20px;
}



Z-INDEX

Overlapping Elements with z-index

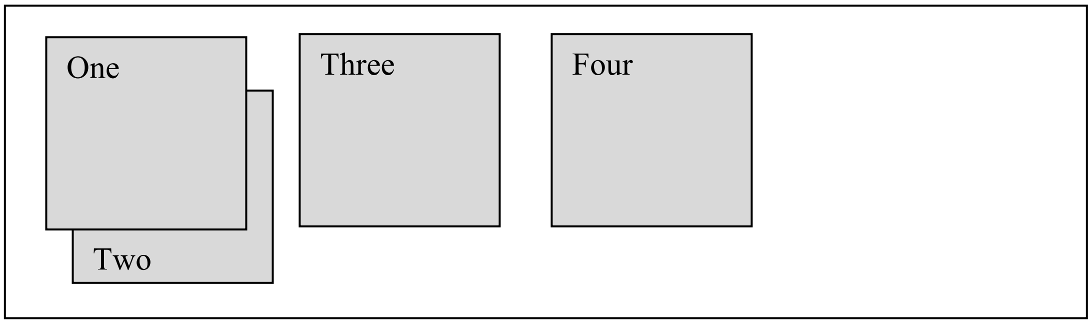
When elements are positioned outside the normal flow, they can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

An element can have a positive or negative stack order:

CSS example from the previous page when the second box goes behind the first box because of z-index:

box2 {
width: 100px;
height: 100px;
position: absolute;
left: 10px;
top: 20px;
z-index: -1;
}

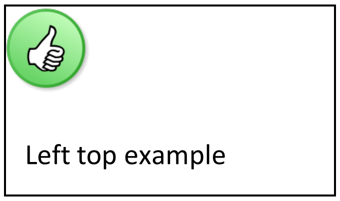


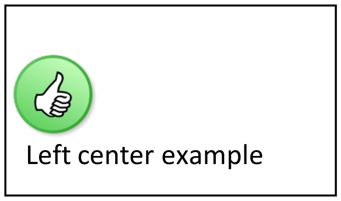
CSS BACKGROUND

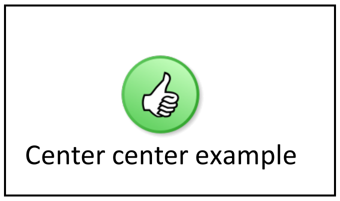
CSS properties used for background effects:

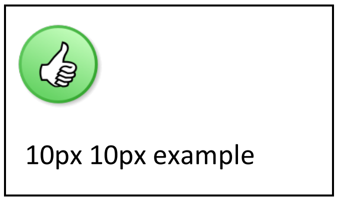
- background-color
- background-image
- background-repeat (no repeat or repeat horizontal or vertical)
- background-attachment (sets whether a background image is fixed or scrolls with the rest of the page.
- background-position

Position examples: left top (initial)
left center
left bottom
right top
right center
right bottom
center top
center center
center bottom
50% 50%
25% 75%
10px 200px
50px 50px









Examples:
<u>BACKGROUND COLOR FOR DIFFERENT ELEMENTS</u>:
body {background-color:#b0c4de;}
h1 {background-color:#6495ed;}
p {background-color:#e0ffff;}
div {background-color:#b0c4de;}

<u>BACKGROUND IMAGE</u>
body {background-image:url('paper.gif');}

1. <u>Background image No Repeat</u>
body {
background-image: url('img_tree.png');
background-repeat: no-repeat;
}

2. <u>Background image Repeat (x or y)</u>
body {
background-image: url('gradient2.png');
background-repeat: repeat-x;
}

3. <u>Background image attachment</u>
body {
background-image: url('image.gif');
background-repeat: no-repeat;
background-attachment: fixed;
}

4. <u>Background image position</u>
body {
background-image: url('img_tree.png');
background-repeat: no-repeat;
background-position: right top;
}

<u>SHORTHAND</u>
body {background: #CCCCCC url('img_tree.png') no-repeat right top;}

STRETCHING BACKGROUND IMAGE

Stretching the background image to completely cover the content area:

CSS code only (no HTML code):

```
html{
   /* This image will be displayed full-screen */
   background: url('name.jpg') no-repeat center center;

   /* Ensure the html element always takes up the full height of the browser window */
   min-height: 100%;

   /* Ensure the element's background is entirely taken up by the image. The image is resized to the smallest size
that allows it to cover the element entirely */
   background-size: cover;
         }

body{
   /* Workaround for some mobile browsers */
   min-height: 100%;
}
```

SHORTHAND:

```
body {
  background: url(name.jpg) center center cover no-repeat fixed;
}
```

JUST IN CASE, FOR DIFFERENT BROWSERS TO WORK WITHOUT PROBLEMS:

```
html {
  background: url(name.jpg) no-repeat center center fixed;
  -webkit-background-size: cover;
  -moz-background-size: cover;
  -o-background-size: cover;
  background-size: cover;
}
```

SIMPLE CONTACT FORM

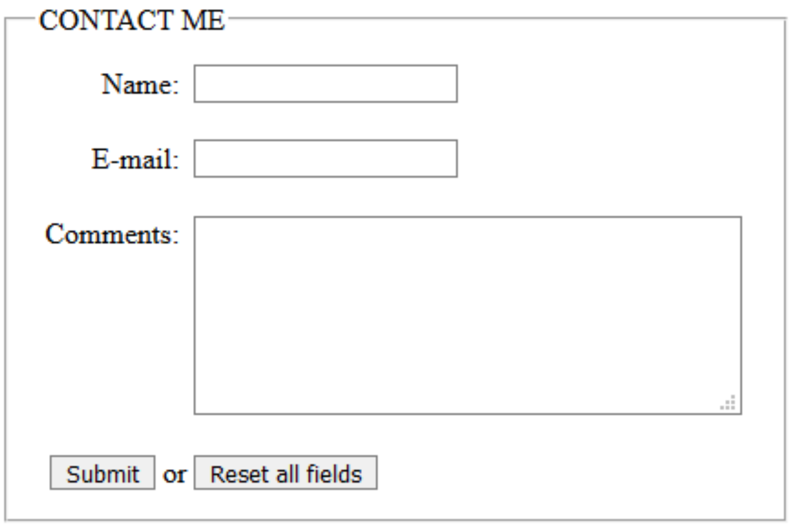
To create a form, usually you have to have 2 files: HTML file, inside which code for form is inserted, and (invisible for the user) PHP file, which processes form code.
There can be many examples of forms layouts; here is only one of them:

Internal CSS part

```
<style>
* {box-sizing: border-box;}

fieldset {
  width: 430px;
  padding: 15px;
     }

input, textarea{
   border:1px solid gray;
    }

label {
  float:left;
  width:20%;
  margin-right:8px;
  text-align:right;
    }
</style>
```

HTML part in <body>

```
<form action="ContactForm.php" method="post">
     <fieldset>
     <legend>CONTACT ME</legend>
       <label for="name">Name:</label>
       <input type="text" name="name" id="name" />
       <br /><br />
       <label for="email">E-mail:</label>
       <input type="text" name="email" id="email" />
       <br /><br />
       <label for="comments">Comments:</label>
       <textarea  name="comments" maxlength="100" cols="35"
rows="6"></textarea>
       <br /><br />
       <input type="submit" value="Submit"> or <input
type="reset" value="Reset all fields">
     </fieldset>
   </form>
```



FEEDBACK
There should be a feedback for the user, something like: *"Thank you for contacting us. We will be in touch with you very soon*
The best way is to insert feedback into PHP file as an HTML part, with an interesting design. Thus, it will be only one PHP page; when it is processed, the user will see only feedback message.

HTML FORM VALIDATION
Whether it's a valid email address or a complete phone number, validation scripts make sure that the data you want arrives in the format you need.
The goal of web form validation is to ensure that the user provided necessary and properly formatted information needed to successfully complete an operation, giving him an error feedback.

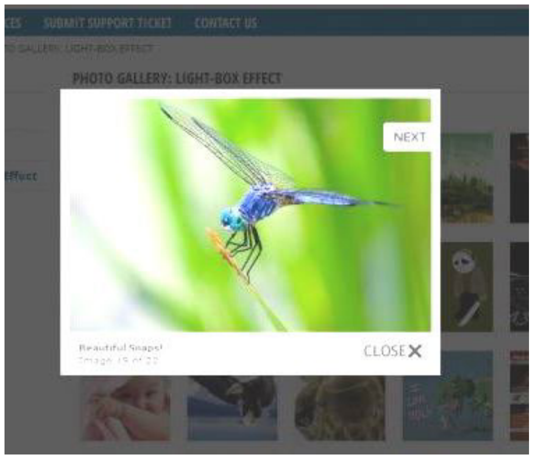
Validation scripts might sound complicated; it can have different techniques, methods and approaches such as Javascript, JQuery, PHP, HTML5 itself.
There are plenty of validation resources, codes and tutorials on the internet

AMAZING CSS THINGS

There are hundreds of cool things that you can do with CSS and sometimes with a little of Javascript or jQuery. You can find and use several codes on your class site, at the end of the left pane

LIGHTWINDOW, LIGHTBOX, LIGHTGALLERY, FANCYBOX, SLIMBOX

All of the above (and more) are overlay boxes (based on JavaScript or jQuery libraries and CSS) with zooming functionality for images and other content, and in the same time making the rest of the web page darker.



THE FEATURES:
Displaying images, HTML elements, videos, Iframes, etc.
Enlarging a selected image and giving the user a closer look at it.
Grouping related items by categories
Having forwards/ backwards navigation for images
Keeping the user on the same page while at the same time focusing on an image (or other content).
Conveying important information without losing the context of the current screen

Warning:
Overlays can be effective in certain contexts, but sometimes they are used without a proper reason, or enough planning, as a convenient way to insert additional content that may or may not be appropriate. Also, at this time overlays do not work consistently across mobile browsers.
Before considering an overlay, carefully investigate your motivations. If you think that overlays are the right design solution for you, proceed with care. Test repeatedly on a variety of browsers and devices to make sure that your design does not break.

ANIMATED FANCY BUTTONS

A REALLY BIG BUTTON
SMALL
A
L
A BUTTON
A BUTTON
A BUTTON

Not only can CSS buttons have hundreds of different appearances, they can also change these appearances to different degrees on mouse over them with CSS 2-D and 3-D animation and transition features
They can change: size, shape (totally or partially), angle, colors, borders, shadows, transparency, etc.
They can move to any direction, curl, rotate, push, pulse, bounce, float, skew, wobble, have bubbles, and much more.

ANCHOR LINKS

Anchor links are used to give a visitor fast access to any area within the page, up or down, to another position or section of the page.
If you have a long page and you want to use anchor links to allow your users to navigate it, you might want to smoothly scroll to the respective position, instead of jumping to it. It takes just some extra steps to enhance that experience by using JavaScript or jQuery.