1.1 Advantages of CSS

Greater control of typography and page layout

Less work

Potentially smaller documents

Potentially more accessible documents

Well-structured and semantically rich documents are more accessible for a wide spectar of devices and users.

Presentational HTML is outdated

Well supported

1.2 Measurement units in CSS

No blanks between the number and the unit. For example 24px, and not 24 px.

No need of unit when the value is 0 (zero).

Values can be decimal, like 14.5cm.

Some values, like object margins, can be negative: margin: -500px

1.2.1.1.1 Units of measurements for style sheet values

Code	Unit	Description
px	Pixel	Pixel units are relative to the monitor resolution.
pt	Point	A traditional publishing unit of measurement for type. In CSS, a point is equal to 1/72 of an inch.
pc	Pica	A traditional publishing unit of measurement equal to 12 points (or 1/6 of an inch).
em	Em	A relative unit of measurement that traditionally equals the width of the capital letter "M" in the current font. In CSS, it is equal to the point size of the font (e.g., an `em` space in `24pt` type is 24 points wide) and is used for both vertical and horizontal measurements.
ex	Ex	A relative unit of measurement that is the height of the lowercase letter "x" for that font (approximately half the length of an em).
in	Inches	Standard unit of measurement in the U.S.
mm	Millimeters	Metric measurement.
cm	Centimeters	Metric measurement.

1.3 Color

Either names or numerical values can be used.

```
h1 {color: olive;}
```

In CSS 2.1 there are 17 names of colors:

aqua	Green	orange	white
black	Lime	purple	yellow
blue	Maroon	red	
fuchsia	Navy	silver	
gray	Olive	teal	

RGB value:

```
{color : #000FF;}
{color: #00F;}
{color: rgb(0,0,255);}
{color: rgb(0%, 0%, 100%);}
```

1.4 Selectors class and id

The selectors class and id can define various styles independent of XHTML elements

Lets assume we need three types of paragraphs:

- Left aligned,
- Right aligned
- Centered

We can do them this way:

```
<style type="text/css">
      p.levo { text-align : left }
      p.desno { text-align : right }
      p.centar { text-align : centar }
</style>

…   …   …

      <p class="desno">
            A right aligned paragraph
      </p>
      <p class="centar">
            A centered paragraph.
```

```
	</p>
	<p class="levo">
		A left aligned paragraph
	</p>
```

Complete code

```
<html>
	<head>
		<title>Стилизирање на страна</title>
		<style type="text/css">
			p.levo { text-align : left }
			p.desno { text-align : right }
			p.centar { text-align : center }
		</style>
	</head>
	<body>
	<p class="desno">
		A right aligned paragraph
	</p>
	<p class="centar">
		A centered paragraph.
	</p>
	<p class="levo">
		A left aligned paragraph
	</p>
	</body>
</html>
```

Results

A right aligned paragraph

A centered paragraph.

A left aligned paragraph.

Similarly with the ID of the tag

```
<style type="text/css">
	#levo { text-align : left }
	#desno { text-align : right }
	#centar { text-align : centar }
</style>
```

In code:

```
<html>
	<head>
		<title>Web page style</title>
			<style type="text/css">
				#levo { text-align : left }
				#desno { text-align : right }
				#centar { text-align : center }
			</style>
	</head>
	<body>
		<p id="desno">
			A right aligned paragraph
		</p>
		<p id="centar">
			A centered paragraph
		</p>
```

```
        <p id="levo">
            A left aligned paragraph
        </p>
    </body>
</html>
```

IMPORTANT: One ID can be used only for one tag, ONCE in a document

It is allowed to use more classes per element

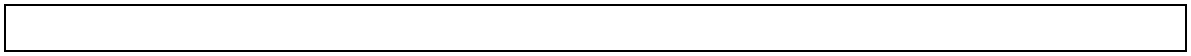
Example:

```
        <style type="text/css">
            p.levo { text-align : left }
            p.desno { text-align : right ; color:blue }
            p.centar { text-align : center }
            p.crveno { color : red }
        </style>
```

Second paragraph

```
        <p class="centar crveno">
            This paragraph will be centered and red.
        </p>
```

The result



A right aligned paragraph

A centered paragraph.

A left aligned paragraph.

If the tag type is not used in the class name:

```
<style type="text/css">
      .levo { text-align : left }
      .desno { text-align : right }
      .centar { text-align : center }
      .crveno { color : red }
</style>
```

The class can be used for all types of tags

```
<h1 class = "centar crveno">
      Centered and red heading
</h1>
<p class = "desno">
      Right aligned paragraph
</p>
```

Centered and red heading

Right aligned paragraph

Example

```
<html>
	<head>
		<title>Стилизирање на страна</title>
			<style type="text/css">
				.polni {font-weight: bold;}
			</style>
	</head>
	<body>
		<p class="polni">
			A paragraph with bold text
		</p>

		<p> In this paragraph, <span class="polni">only
		this part is bold</span>.
			The remainder is written in normal letters.
		</p>
	</body>
</html>
```

1.5 Links and pseudo classes

CSS enables various effects otherwise possible only with JavaScript.

The links have 4 possible states

link – a link that has not been visited
visited – an already visited link
hover – a link being pointed at with the mouse at the moment
active – a selected link in the moment of clicking

.

Pseudo class syntax

selector:pseudo-class {property: value}

Pseudo classes for links

a:{ name of the state } { attribute : value }

Examples

with

```
a {
        color: blue;
```

```
}
```

The link will be blue

```
a:link {
        color: blue;
    }

a:visited {
        color: red;
     }
```

When hovering over a link:

```
a:hover {
        color: orange;
        font-style: italic;
    }
```

Important:

a:hover must be after a:link and a:visited in the CSS definition

A hint to remember: LoVe, HA!

The proper order of the definitions is :link, :visited, :hover, :active.

Example

```
a:hover  {
            letter-spacing : 10px;
            font-weight : bold;
            color : red;
          }
```

Pseudo classes and classes can be combined

Example

```
<html>
    <head>
        <style type="text/css">
            a.one:link {color: red}
            a.one:visited {color: blue}
            a.one:hover {color: green}

            a.two:link {color: red}
            a.two:visited {color: blue}
            a.two:hover {font-size: 150%}

            a.three:link {color: red}
            a.three:visited {color: blue}
            a.three:hover {background: green}

            a.four:link {color: red}
            a.four:visited {color: blue}
            a.four:hover {font-family: monospace}

            a.five:link {color: red; text-decoration:
        none}
            a.five:visited {color: blue; text-decoration:
        none}
```

```
            a.five:hover {text-decoration: underline}
        </style>
    </head>
    <body>
    <p> Changing link appearance </p>
    <p> <b> <a class="one" href=" " target="_blank">
        Change link color</a></b></p>
    <p> <b> <a class="two" href=" " target="_blank">
        Change size</a></b></p>
    <p> <b> <a class="three" href=" " target="_blank">
        Change background</a></b></p>
    <p> <b> <a class="four" href=" " target="_blank">
        Change font family</a></b></p>
    <p> <b> <a class="five" href=" " target="_blank">
        Underline text</a></b></p>
    </body>
</html>
```

1.6 Universal selector

Can influence all tags:

```
* {color: blue; }
```

All tags will have blue text color

Same as if it was written:

```
body, h1, h2, h3, h4, h5, h6, p, table, tr, td, th, pre, ul,
li { color: blue ; }
```

Try

```
<html>
    <head>
        <style>
            div * { color : red }
        </style>
    </head>
    <body>
        <div>
            <table>
                Normal letters
                    <p>
                        Red letters
                    </p>
            </table>
        <div>
    </body>
</html>
```

Normal letters

Red letters

1.6.1.1.1 Summary of selectors		
Selector	**Type of selector**	**Description**
`*`	Universal selector	Matches any element. `* {font-family:serif;}`
`A`	Type selector	Matches the name of an element. `div {font-style: italic;}`
`A B`	Descendant selector	Matches element B only if it is a descendant of element A. `blockquote em {color: red;}`
`A>B`	Child selector	Matches any element B that is a child of any element A. `div.main>p {line-height:1.5;}`

1.6.1.1.1 Summary of selectors		
Selector	**Type of selector**	**Description**
`A+B`	Adjacent sibling selector	Matches any element B that immediately follows any element A. `p+ul {margin-top:0;}`

`.classname`
 	
`A.classname`
	Class selector	Matches the value of the `class` attribute in all elements or a specified element.
		
		`p.credits {font-size: .8em;}`

`#idname`
 	
`A#idname`
	ID selector	Matches the value of the `id` attribute in an element.
		
		`#intro {font-weight: bold;}`

`A[att]`	Simple attribute selector	Matches any element A that has the given attribute defined, whatever its value. `table[border] {background-color: white;}`
`A[att="val"]`	Exact attribute value selector	Matches any element B that has the specified attribute set to the specified value. `table[border="3"] {background-color: yellow;}`
`A[att~="val"]`	Partial attribute value selector	Matches any element B that has the specified value as one of the values in a list given to the specified attribute. `table[class~="example"] {background-color: orange;}`

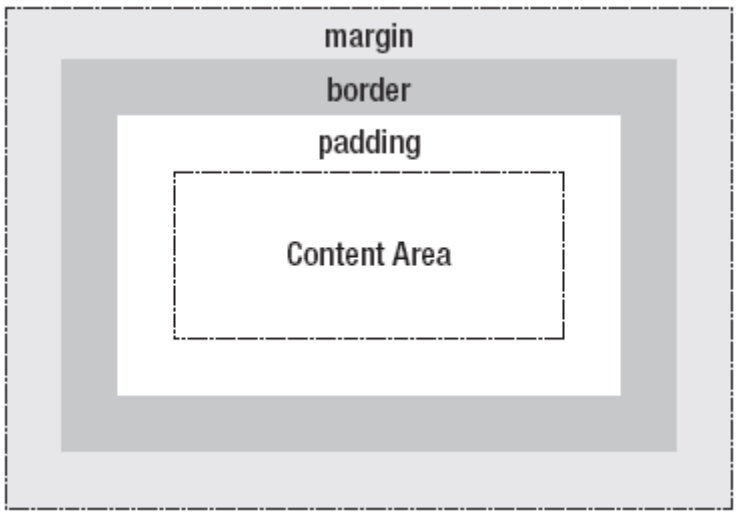
1.6.1.1.1 Summary of selectors		
Selector	**Type of selector**	**Description**
`A[hreflang|="es"]`	Hyphenated prefix attribute selector	Matches any element A that has an attribute `hreflang` with a hyphen-separated list of values beginning (from the left) with "es". `a[hreflang|="es"] {background-image: url(flag-es.png);}`
`a:link`	Pseudoselector	Specifies a style for links that have not yet been visited. `a:link {color: purple;}`
`a:visited`	Pseudoselector	Specifies a style for links that have already been visited. `a:visited {color: gray;}`
`:active`	Pseudoselector	Applies a style to elements (typically links) while in their active state. `a:active {color: red;}`
`:after`	Pseudoselector	Inserts generated text at the end of the specified element and applies a style to it. `p.intro:after {content: "fini"; color: gray;}`
`:before`	Pseudoselector	Inserts generated text at the beginning of the specified element and applies a style to it. `p.intro:before {content: "start here "; color: gray;}`
`:firstchild`	Pseudoselector	Specifies a style for an element that is the first child of its parent element in the flow of the

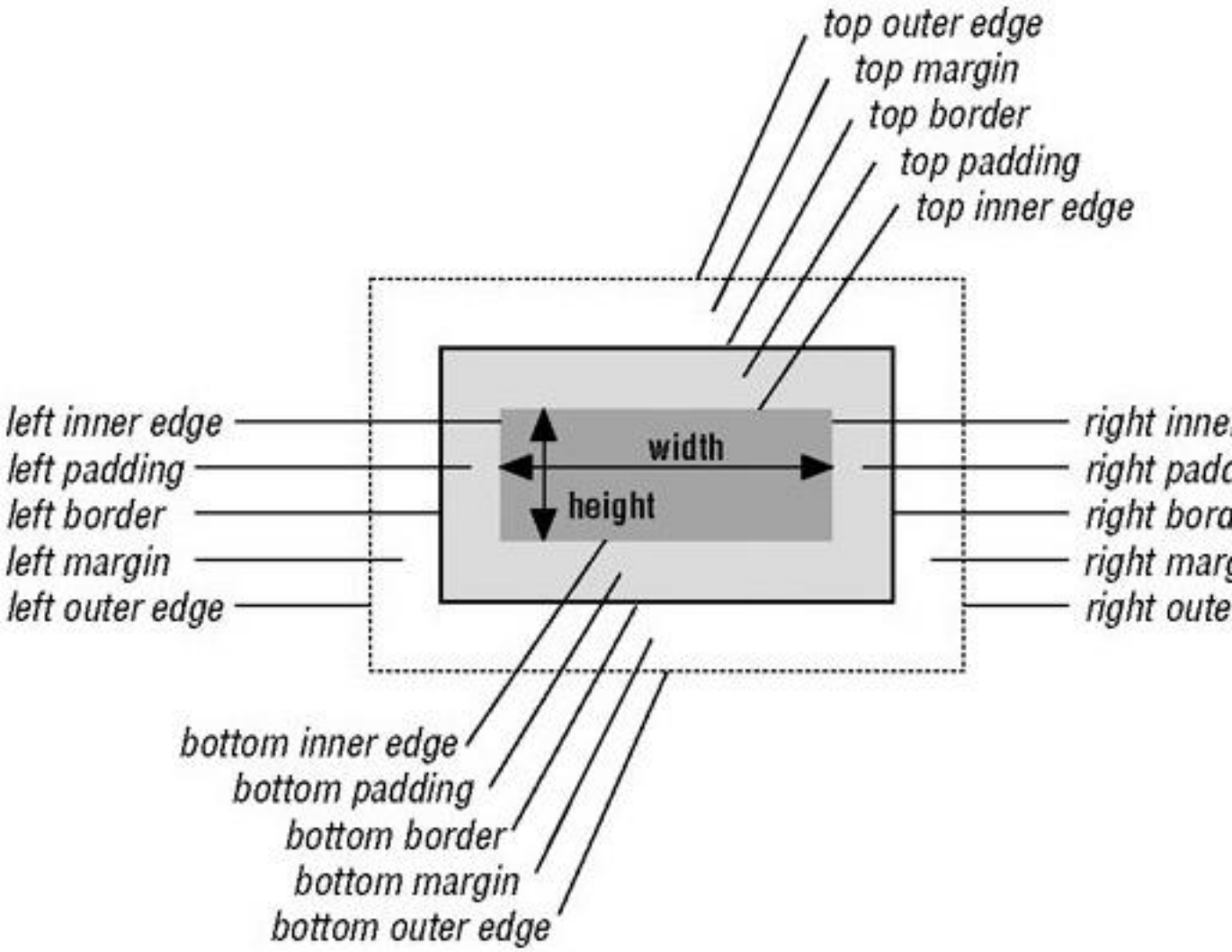
1.6.1.1.1 Summary of selectors		
Selector	**Type of selector**	**Description**
		document source. `p:firstchild {text-style: italic;}`
`:first-letter`	Pseudoselector	Specifies a style for the first letter of the specified element. `p:first-letter {font-size: 60px;}`
`:first-line`	Pseudoselector	Specifies a style for the first line of the specified element. `p:first-line {color: fuchsia;}`
`:focus`	Pseudoselector	Specifies a style for elements (typically form controls) that have focus (selected and ready for user input). `input[type="text"]:focus {background-color: yellow;}`
`:hover`	Pseudoselector	Specifies a style for elements (typically links) that appears while the pointer is over them. `a:hover {text-decoration: underline;}`
`:lang(ab)`	Pseudoselector	Specifies a style for an element for which its language matches the given language code (or language code prefix). `a:lang(de) {color: green;}`

2 Box model

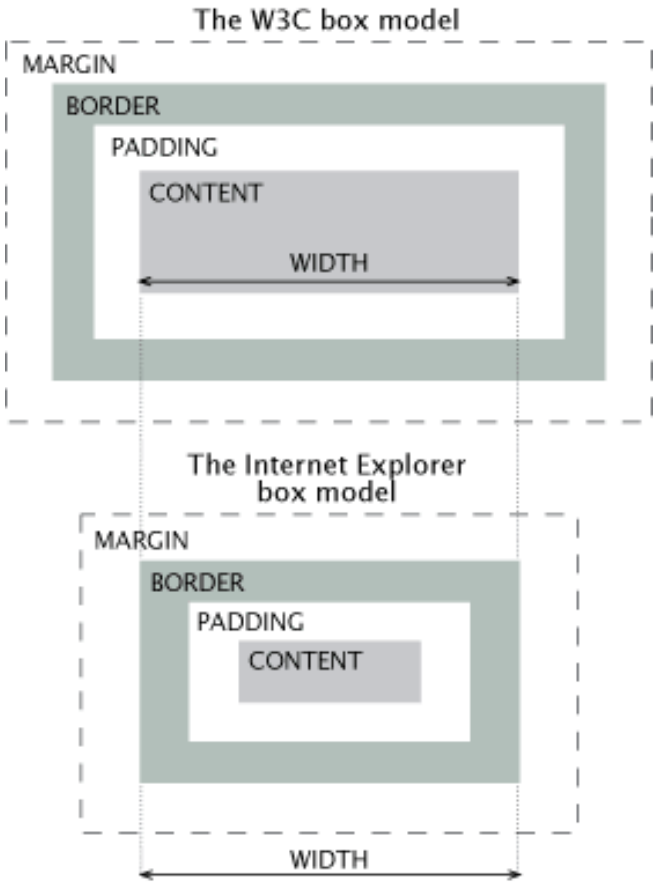
Three important concepts with cascade style sheets are (floating), positioning and box model.

Every element on the page is a box





Be careful for the difference in the W3C box model and the Internet Explorer model



To avoid such issues, some elements can be set to 0 селектор *

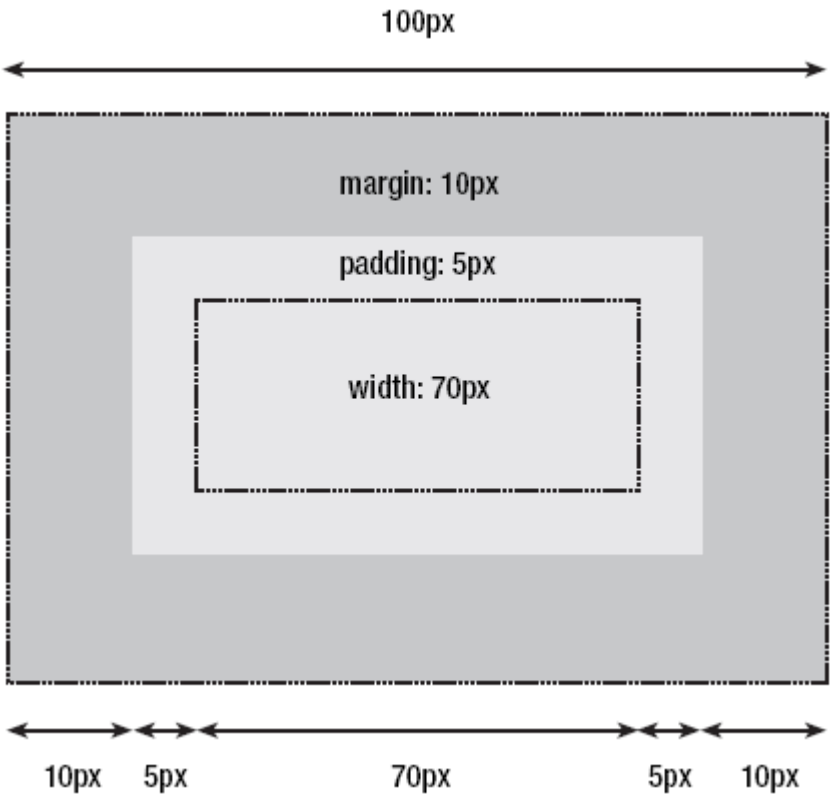
```
* {
    margin: 0;
    padding: 0;
  }
```

With

```
#myBox {
        margin: 10px;
        padding: 5px;
```

```
 width: 70px;
}
```

You get



Background colors and images applied to the element are visible in the padding and go below the border.(If there are transparent parts of the border, the background is visible).

The margin is always transparent.