



Fulvio Corno, Dario Bonino



Basics

CSS: Cascading Style Sheet

CSS 1: W3C recommendation 17 Dec 1996

CSS 2.1: W3C Candidate Recommendation 19 July 2007

CSS 3: “under construction”

Resources:

- CSS2.1 standard

- <http://www.w3.org/TR/CSS21/>

- W3C CSS Tutorial

- <http://www.w3.org/Style/Examples/011/firstcss>



How to write CSS

Using a normal text editor

- (recommended for learning)

Using a dedicated tool

Examples:

- jEdit
- Notepad
- Dreamweaver
- Emacs...



Comments and Units of measure

Comments: similar to C or Java:

- `/* this is a comment */`

Units of measure

- Relative units

- em: the font-size of the relevant font
- ex: the 'x-height' of the relevant font
- px: pixels, relative to the viewing device



Comments and Units of measure

- Absolute units

- in: inches
- cm: centimeters
- mm: millimeters
- pt: points $1/72 \times 1\text{inch}$
- pc: picas, (12 points)



Rules and Style sheets

CSS is based on rules

A *rule* is a statement about one stylistic aspect of one or more elements (XHTML)

A *style sheet* is a set of one or more rules that apply to an XHTML document

Example

```
■ h1 {color: green}
```



Anatomy of a Rule

A rule consists of two parts

- Selector – the part before the left curly brace
- Declaration – the part within the curly braces

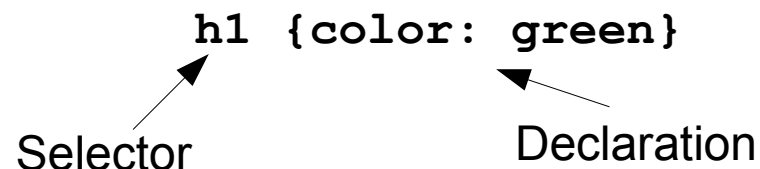
The selector is the link between the XHTML document and the style

The selector specifies what elements are affected by the declaration

The declaration sets what the effect will be

`h1 {color: green}`

Selector Declaration





Anatomy of a Rule (2)

There are several kind of selectors

- The selector h1 in the previous slide is a *type selector* and is based on an XHTML element type
- The *type selector* is the simplest kind of selector

The final effect of the h1 rule is:

- All the h1 elements are written in green



Anatomy of a Declaration

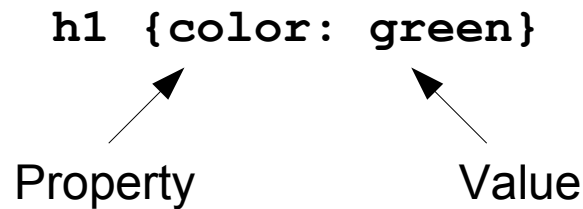
A declaration has two parts separated by a colon:

- Property – the part before the colon
- Value – the part after the colon

The property is a quality or a characteristic that something possesses (the color in the example)

The value is a precise specification of the property (in the example: green)

```
h1 {color: green}
```



Property Value



Grouping Selectors and Rules

Style sheets load faster if they are shorter

There are several mechanisms to shorten a style sheet by grouping selectors and declarations

Examples

```
h1 {font-weight: bold}
h2 {font-weight: bold}
h3 {font-weight: bold}  ≡  h1, h2, h3 {font-weight: bold}
```



Grouping Selectors and Rules (2)

Example2

```
h1 {font-weight: bold}  
h1 {color: green}
```

==

```
h1  
{  
    font-weight: bold;  
    color: green;  
}
```



“Gluing” Style sheets to the document

3 main ways

- Using the `<style>` element
- Using in-line styles (`style` attribute)
- Link an external style sheet using the element `<link>`



Document-wide style sheets

The `<style>` element is used

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title> My first Home Page with Styles </title>
    <style type="text/css">
      <!--
        h1 {color: green}
      -->
    </style>
  </head>
  <body>
    ...
  </body>
</html>
```



In-line style sheets

The style attribute is used

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Untitled Document</title>
  </head>

  <body>
    <h1 style=" color: green;">prova</h1>
  </body>
</html>
```



Separated style sheets

The style sheet is define in a separated file

style.css

```
h1 {color: green}
```

test.html

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Untitled Document</title>
    <link rel="stylesheet" type="text/css" href="style.css"/>
  </head>

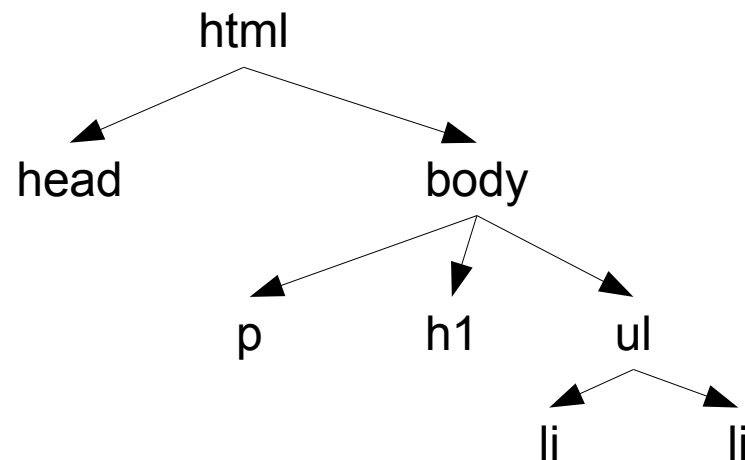
  <body>
    <h1>prova</h1>
  </body>
</html>
```



Tree structure and inheritance

XHTML documents are trees

Styles are inherited along trees





Tree structure and inheritance

(2)

So if we want to set the color:green for all elements we can:

- Set the color for each XHTML element (not recommended)
 - `h1, h2, h3 {color: green}`
- Set the color for the common ancestor of the elements (recommended)
 - `body { color: green }`



Overriding Inheritance

It is possible to override styles assigned to the parent elements of a given tag

Example

```
body {color: green}
```

```
h1 {color: navy}
```

The two rules are in conflict!! who wins?

- The most specific one!!



Inheritance Exceptions

Usually properties in CSS inherit from parent to child as in the previous slide

However some properties do not!

Example: background!!

Why?

- You may end up with a non smooth background surface (if each element has its own background that is equal to all the others)
- The loading time can sensibly increase



Common tasks: fonts

CSS allow to define several font properties for tuning the text presentation of a XHTML document

font-size: fixes the size of a text element (in pixel, points or ems)

font-family: establishes the font with wich the text is written (not all fonts are supported by all browsers)

font-style: the style of the font **normal** | **italic** | **oblique**



Common tasks: fonts(2)

font-weight: the “boldness” of the font, can be
bold | bolder | lighter | normal | 100 | 200 | 300 |
400 | 500 | 600 | 700 | 800 | 900

font-variant: used for formatting text with all
uppercase characters (bigger for initials), can be
normal | small-caps

line-height: fixes the height of the line in which
the text is written values can be in pixels, points or
ems (px, pt, em), in percentage with respect to the
font-size, etc.



Example

Want to have a heading1 in Times font, with 24pt height, bold, italic, navy color and a line height that is 150% of the font size

```
h1
{
    font-size: 24pt;
    font-family: Times, Times New Roman;
    font-style: italic;
    font-weight: bold;
    color: navy;
    line-height: 150%;
}
```



Common Tasks: margins

The margins define the space that the browser shall leave around a given text element

The possible margins are:

- `margin-top`
- `margin-right`
- `margin-left`
- `margin-bottom`

The margin size is usually given in ems

`1em` = font-size



Example

A blockquote margins'

```
blockquote
{
    margin-top: 1em;
    margin-right: 0em;
    margin-left: 0em;
    margin-bottom: 1em;
    font-style: italic;
}
```

There is also a shorthand that allows to specify margins in a single value (top, right, bottom, left):

```
margin: 1em 0em 1em 0em;
```




Common tasks: background

Each XHTML element can have a background

```
blockquote
{
    margin-top: 1em;
    margin-right: 0em;
    margin-left: 0em;
    margin-bottom: 1em;
    font-style: italic;
    background: #EDB;
}
```

Background is a shortcut for 5 different attributes (whose values can be written in a single line, space separated)



Common tasks: background (2)

The 5 background attributes are

- **background-attachment**

- Fixes the attachment to the document, can be

- `fixed` | `scroll`

- Scroll: follows the document scrolling

- **background-color**

- Fixes the background color, can be a named color or a RGB value

- **background-image**

- An image to be used as background

- Syntax: `url(imageURL)`



Common tasks: background (2)

- **background-position**

- Establishes the left and top edges of the background image
- Can be
 - A percentage (horizontal% height%) (if only 1 value is specified it is interpreted as an horizontal% and height% is fixed at 50%)
 - A length (same as for percentage)
 - A mix of
 - `top | center | bottom`
 - `left | center | right`



Common tasks: background (3)

- **background-repeat**

- Set if a background image shall be repeated and along which axis
- Allowed values
 - `no-repeat` | `repeat` | `repeat-x` | `repeat-y`



Common tasks: links

Browsers usually represent links as underlined text

CSS offer a special support for styling anchors

4 different pseudo classes

- `a:link { color: red } /* unvisited links */`
- `a:visited { color: blue } /* visited links */`
- `a:hover { color: yellow } /* user hovers */`
- `a:active { color: lime } /* active links */`



Example

A link should be visualized blue on white without being underlined when it is not selected

White on blue when the mouse is hover the link

And magenta on white when the link has been already selected



Example (2)

```
/* not-visited link: blue on white */  
a:link {color: navy; background-color: white;}  
  
/* mouse over the link: white on blue */  
a:hover{color: white; background-color: navy;}  
  
/* already visited link: magenta on white */  
a:visited {color: magenta; background-color: white;}
```



Advanced Selectors: Classes

CSS **Classes** can be used to style XHTML in different ways

Like “styles” in wordprocessors

In XHTML

```
<p class="author">Dario Bonino</p>
```

In the CSS

```
p.author {font-style: italic; font-color:red;}
```




Advanced Selectors: Before / After

It is also possible to insert content before and after a given XHTML element identified by a selector

Before (CSS 2)

```
table:before {  
    content: "Table<br/>";  
}
```



Advanced Selectors: Before / After (2)

After (CSS 2)

```
img:after {  
    content: "Fig." ;  
}
```



Advanced Selectors: Multiple selectors, Child selectors,...

Selectors can be multiple

- `h1, h2, h3 { font-weight: bold}`

It is possible to select children of a given node

- `div.section > div.note {}`

It is possible to select descendants of a given node

- `p quote { font-style: italic}`

Etc.



Examples

e /*matches any e element*/

e f /*matches any f element descendant of an e element*/

e > f /*matches any f element that is child of an e element*/

e + f /*matches any f element immediately preceded by a sibling element e*/

e [foo="warning"] /*matches any element e having a foo attribute with value "warning"*/

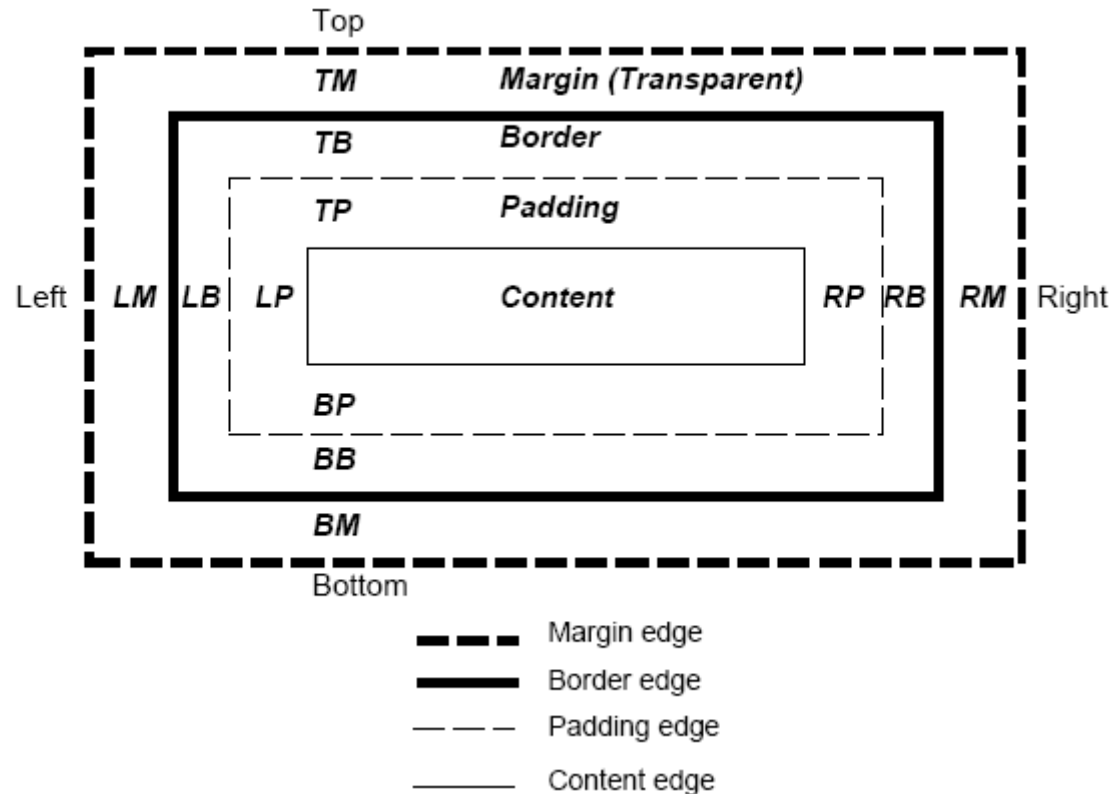
#d /* matches any element with an attribute id='d' */

.c /* matches any element with an attribute class='c' */

CSS Selector	Meaning
*	Matches any element.
E	Matches any E element (i.e., an element of type E).
E F	Matches any F element that is a descendant of an E element.
E > F	Matches any F element that is a child of an element E.
E:first-child	Matches element E when E is the first child of its parent.
E:link, E:visited	Matches element E if E is the source anchor of a hyperlink of which the target is not yet visited (:link) or already visited (:visited).
E:active, E:hover, E:focus	Matches E during certain user actions.
E:lang(c)	Matches element of type E if it is in (human) language c (the document language specifies how language is determined).
E + F	Matches any F element immediately preceded by an element E.
E[foo]	Matches any E element with the "foo" attribute set (whatever the value).
E[foo="warning"]	Matches any E element whose "foo" attribute value is exactly equal to "warning".
E[foo~="warning"]	Matches any E element whose "foo" attribute value is a list of space-separated values, one of which is exactly equal to "warning".
E[lang ="en"]	Matches any E element whose "lang" attribute has a hyphen-separated list of values beginning (from the left) with "en".
DIV.warning	The same as DIV[class~="warning"].
E#myid	Matches any E element ID equal to "myid".

Page formatting: the Box Model

Each XHTML element has a box model





Page formatting: the Box Model

content edge or inner edge

- The content edge surrounds the rectangle given by the width and height of the box, which often depend on the element's rendered content.

padding edge

- The padding edge surrounds the box padding. If the padding has 0 width, the padding edge is the same as the content edge.



Page formatting: the Box Model

border edge

- The border edge surrounds the box's border. If the border has 0 width, the border edge is the same as the padding edge.

margin edge or outer edge

- The margin edge surrounds the box margin. If the margin has 0 width, the margin edge is the same as the border edge.

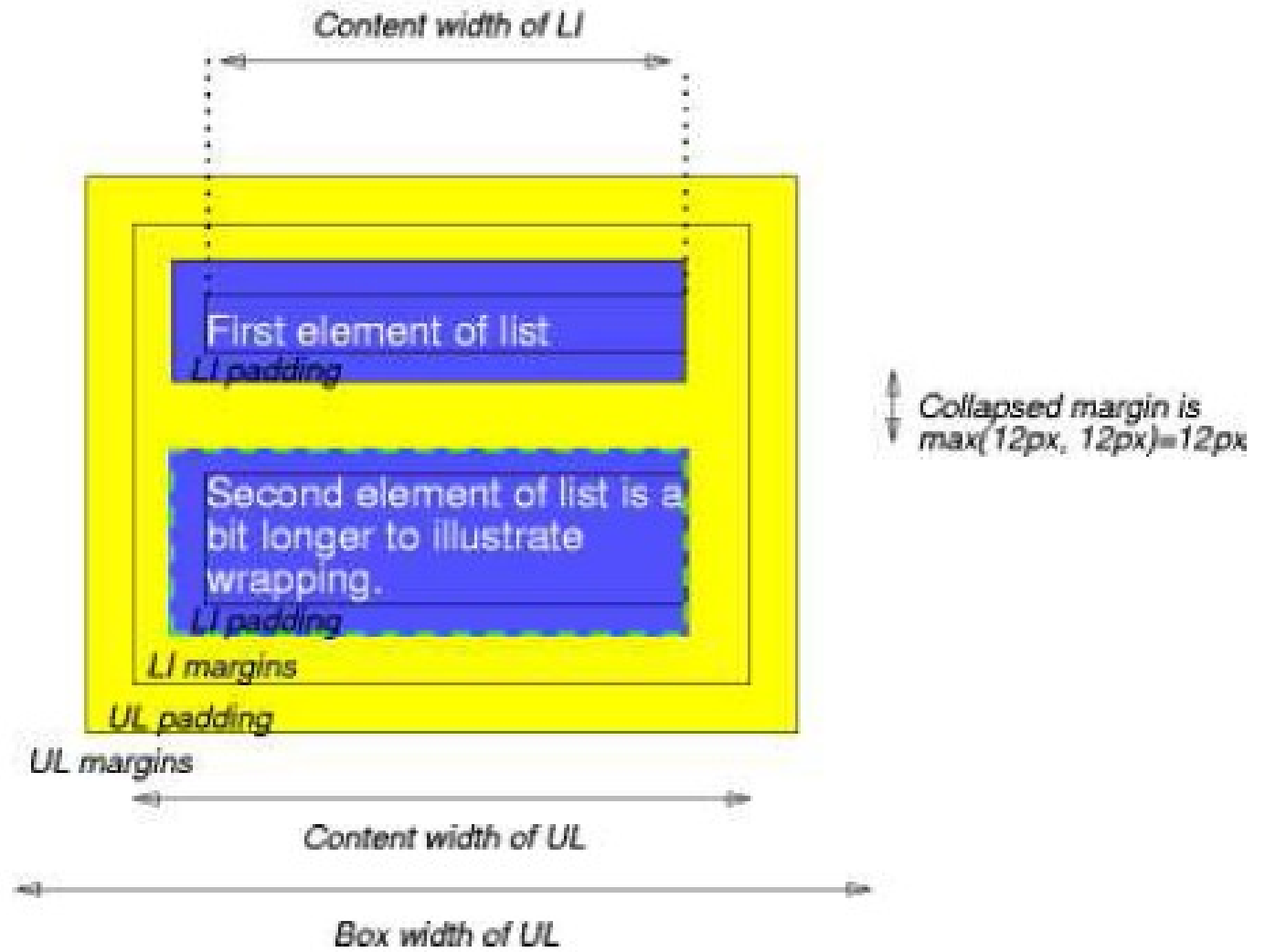


Examples (css code)

```
ul {
background: yellow;
margin: 12px 12px 12px 12px;
padding: 3px 3px 3px 3px;
/* No borders set */
}
li {
color: white; /* text color is white */
background: blue; /* Content, padding
will be blue */
margin: 12px 12px 12px 12px;
padding: 12px 0px 12px 12px; /* Note 0px
padding right */
list-style: none /* no glyphs before a
list item */
/* No borders set */
}
li.withborder {
border-style: dashed;
border-width: medium; /* sets border
width on all sides */
border-color: lime;
}
```

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html
xmlns="http://www.w3.org/1999/xhtml">
  <head></head>
  <body>
    <ul>
      <li>First element of list</li>
      <li class="withborder">
        Second element of list is
        a bit longer to illustrate
        wrapping.
      </li>
    </ul>
  </body>
</html>
```

Examples (results)





Run-in boxes

They are useful for example to make “run-in” headers

Example

```
h3 { display: run-in }
```

The result is:

A run-in heading.

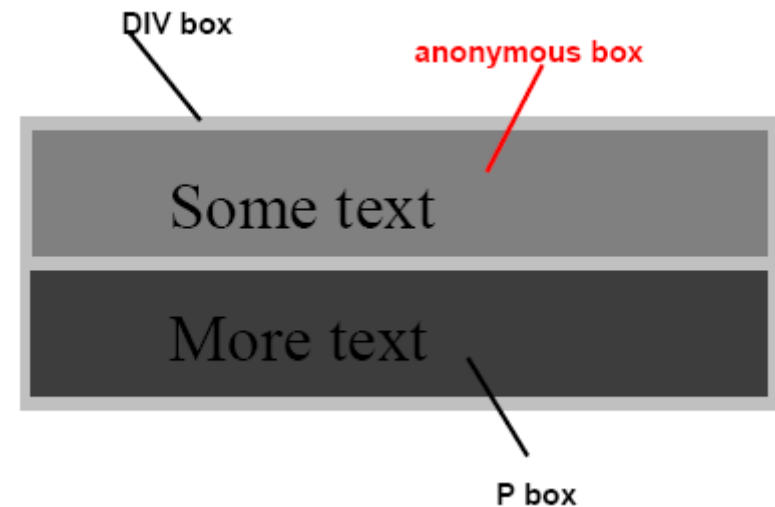
And a paragraph of text that follows it.

Block elements

The XHTML block element are the `<div></div>` element and the `` element

Example:

```
<div>  
  Some text  
  <p>More text</p>  
</div>
```





Box Positioning

A block can be positioned in different ways to which correspond different positioning schemes

Position: `static` | `relative` | `absolute` | `fixed` | `inherit`

static: normal block

relative: the offset values are relative to the block position in the normal flow. If a relative block B follows a relative block A, the offset is respect to the position of A without the offset



Box Positioning

absolute: the box position is determined by the top, left, right, bottom properties and is relative to the containing block

fixed: the box is fixed with respect to some reference (the viewport as an example)

Examples

```
@media screen {  
  div.header { position: fixed }  
}  
  
@media print {  
  div.header { position: static }  
}
```



Box offset

For absolute and relative position, an offset can be specified

- `top : length | percentage`
- `left: length | percentage`
- `right : length | percentage`
- `bottom : length | percentage`



Floats

A float is a box that is shifted to the left or right on the current line.

Content may flow along its side (or be prohibited from doing so by the 'clear' property)

A floated box is shifted to the left or right until its outer edge touches the containing block edge or the outer edge of another float.



Float (2)

If there's a line box, the top of the floated box is aligned with the top of the current line box.

If there isn't enough horizontal room for the float, it is shifted downward until either it fits or there are no more floats present.

Since a float is not in the flow, non-positioned block boxes created before and after the float box flow vertically as if the float didn't exist.

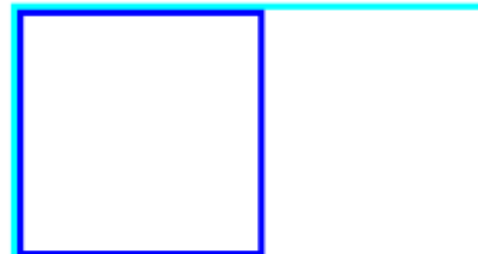


Example

the containing block is too narrow to contain the content next to the float, so the content is moved below the float

```
p { width: 10em; border: solid aqua; }  
span { float: left; width: 5em; height: 5em;  
      border: solid blue; }
```

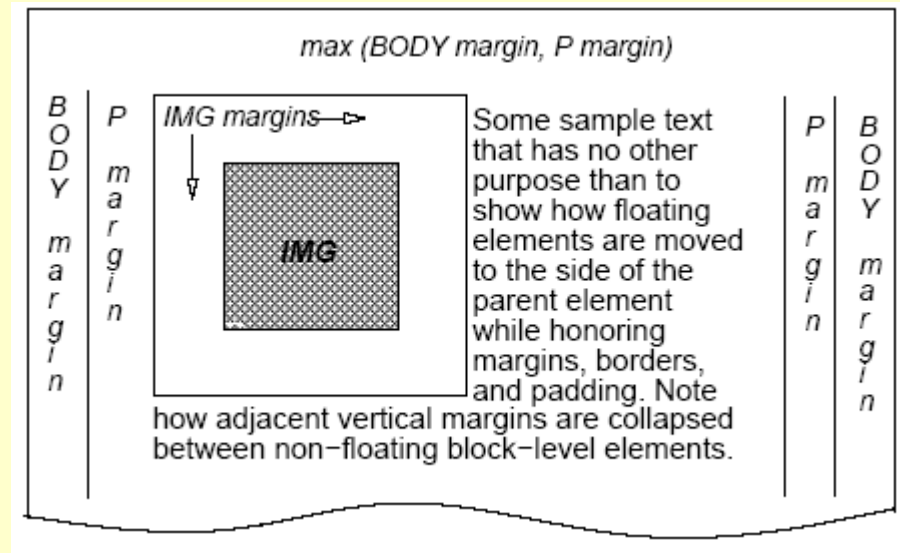
```
<p>  
<span> </span>  
Supercalifragilisticexpialidocious  
</p>
```



Supercalifragilisticexpialidocious

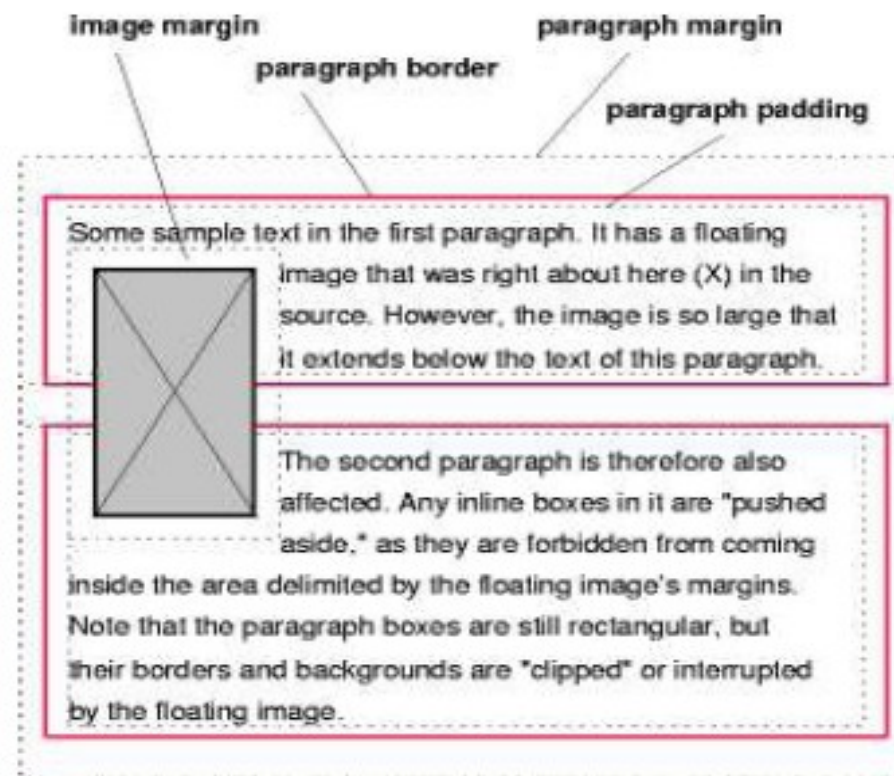
Example 2

```
<html>
<head>
<title>float example</title>
<style type="text/css">
img { float: left }
body, p, img { margin: 2em }
</style>
</head>
<body>
<p><img src=img.png alt="this image will illustrate floats">
some sample text that has no other...
</body>
</html>
```



Example 3

Floating along two paragraphs

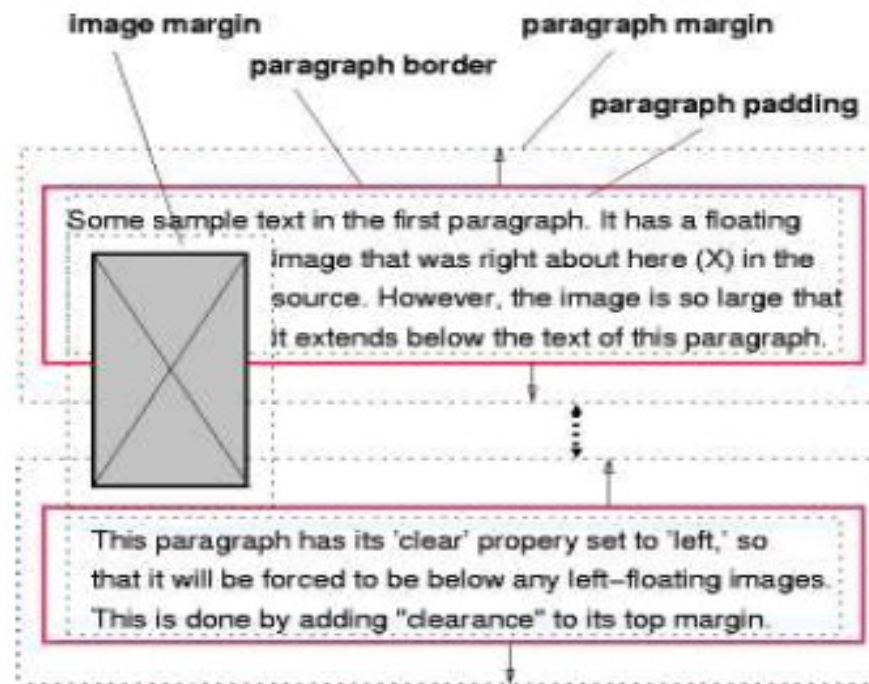


Example 4

The `clear` property

If the `p` style is

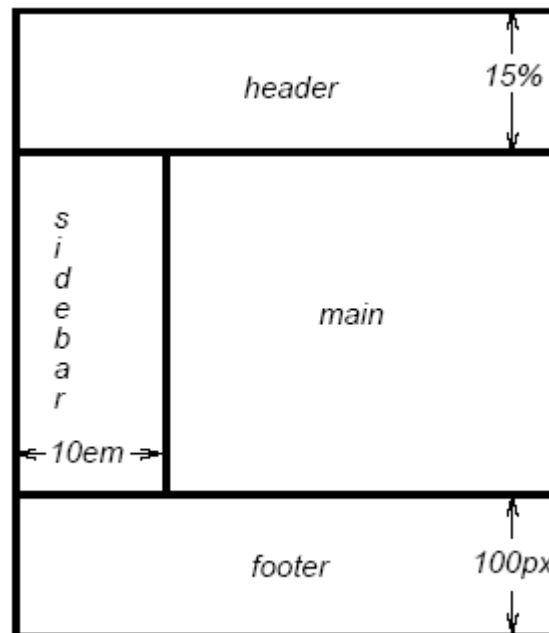
- `p { clear: left }`



Page arrangement

Fixed positioning can be used to create some complex frame-like presentations

For example, the following:





Page arrangement (2)

```
#header { position: fixed; width: 100%;  
height: 15%; top: 0; right: 0; bottom:  
auto; left: 0; }
```

```
#sidebar { position: fixed; width: 10em;  
height: auto; top: 15%; right: auto;  
bottom: 100px; left: 0; }
```

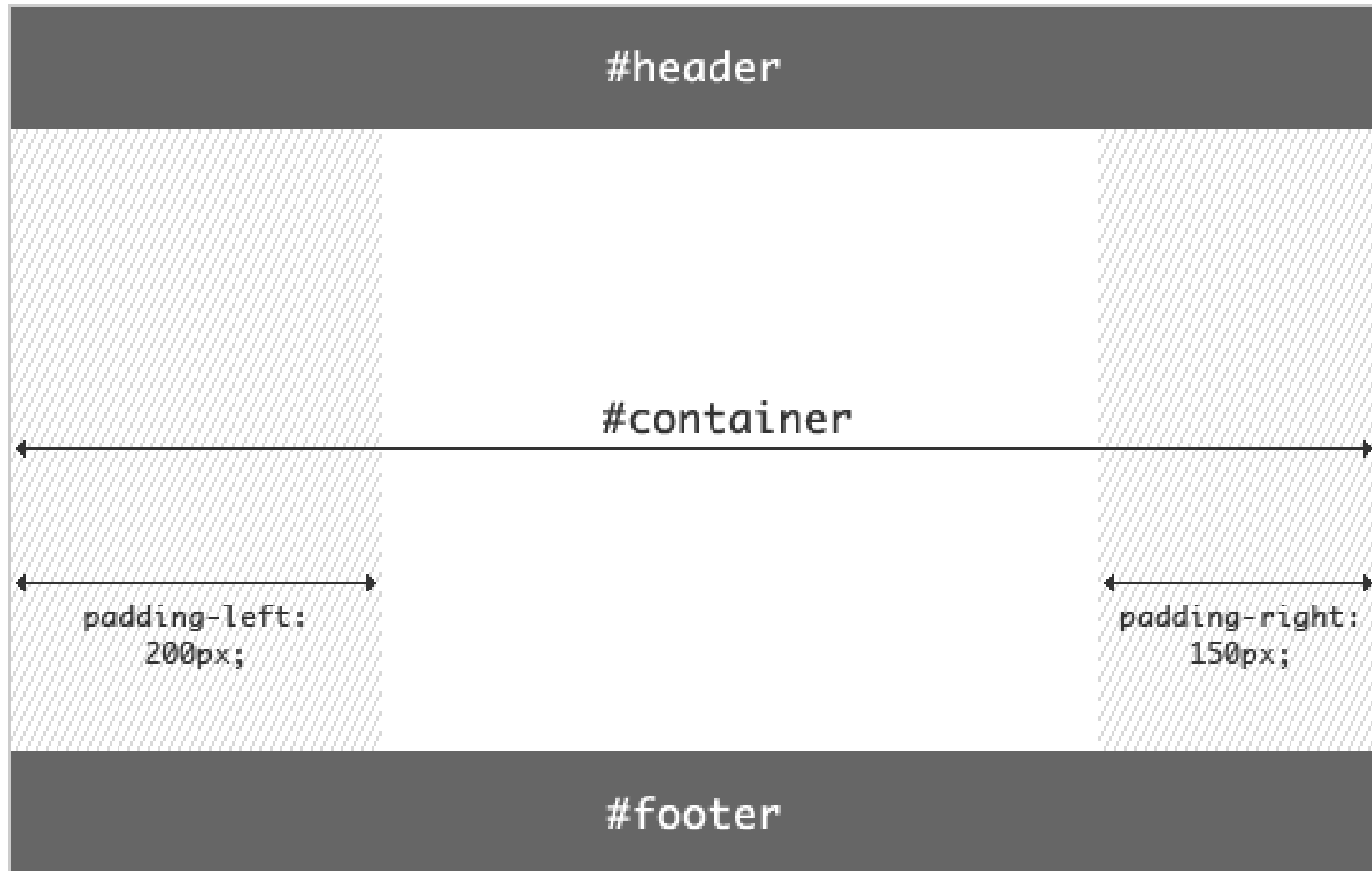
```
#main {position: fixed; width: auto;  
height: auto; top: 15%; right: 0; bottom:  
100px; left: 10em; }
```

```
#footer {position: fixed; width: 100%;  
height: 100px; top: auto; right: 0;  
bottom: 0; left: 0; }
```

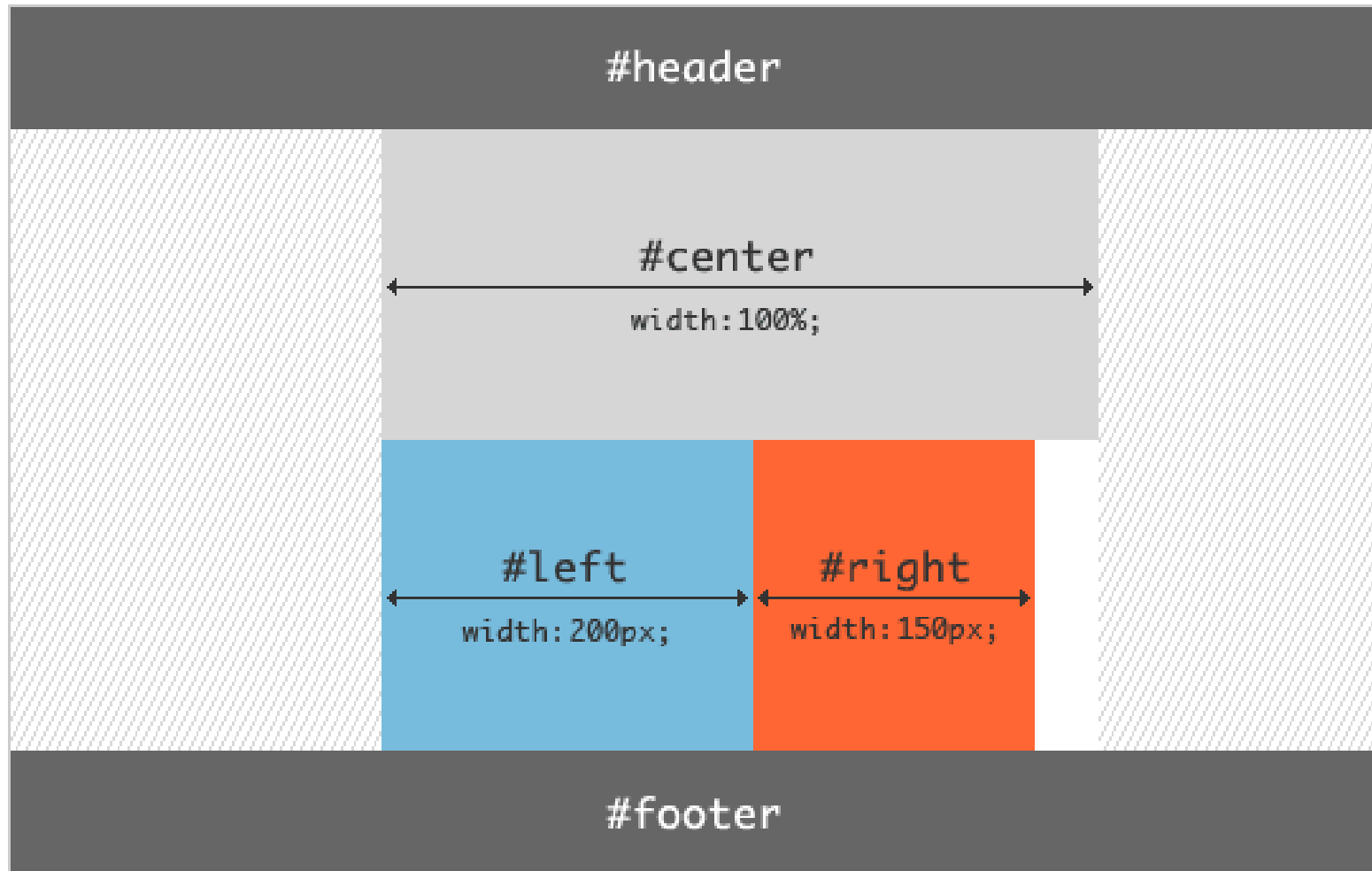
3 Columns Liquid Layout (1/7)



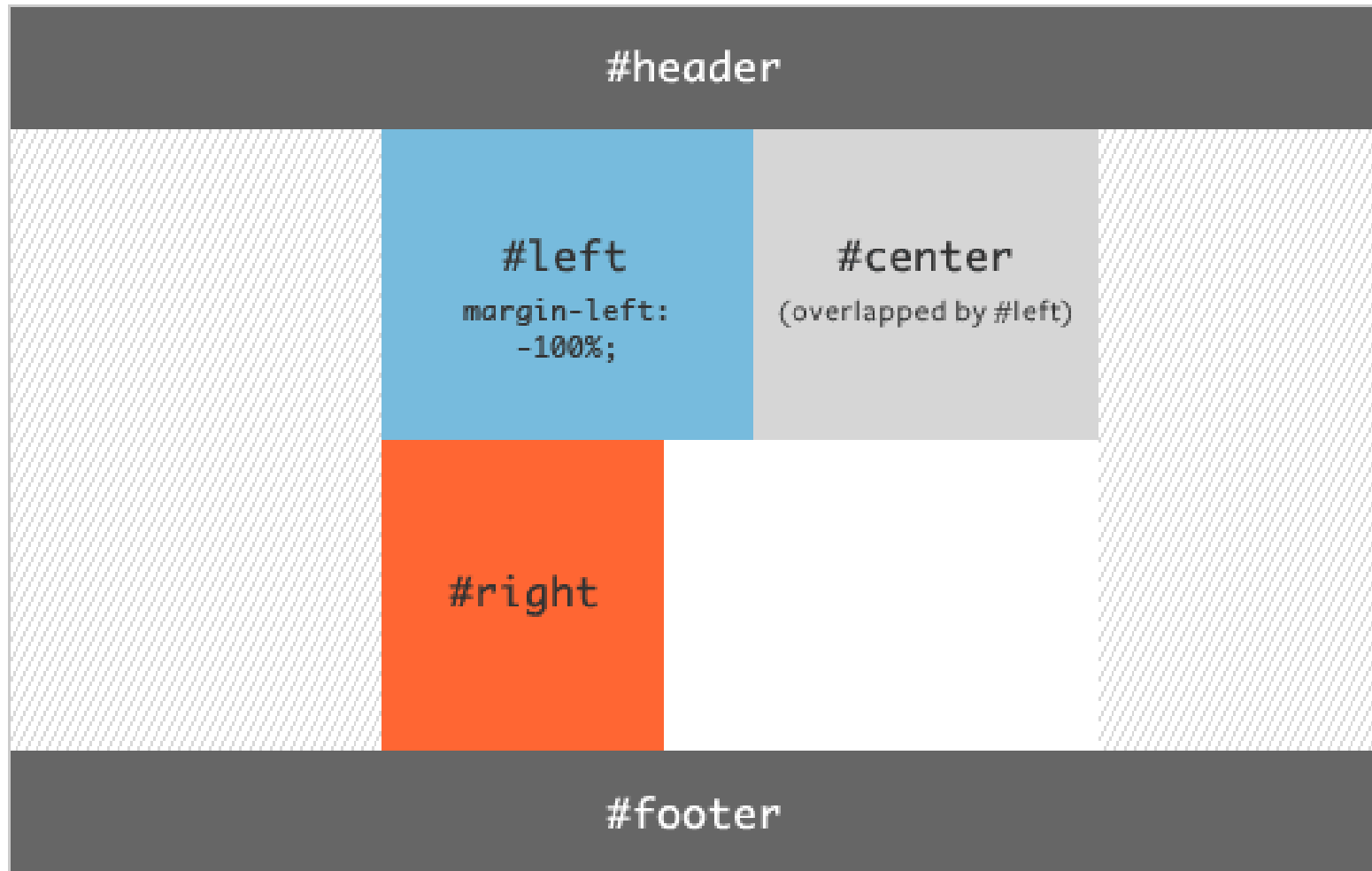
3 Columns Liquid Layout (2/7)



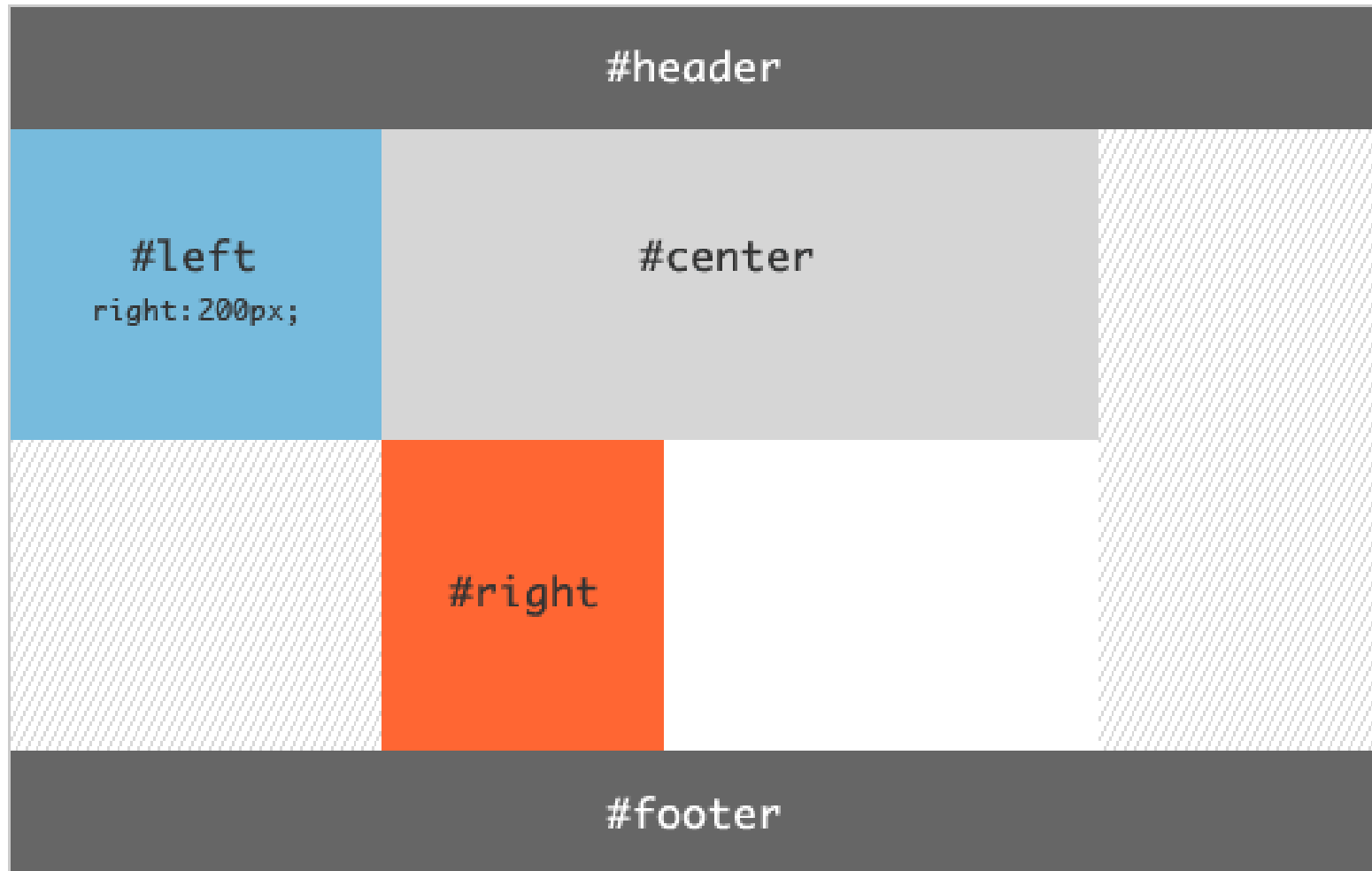
3 Columns Liquid Layout (3/7)



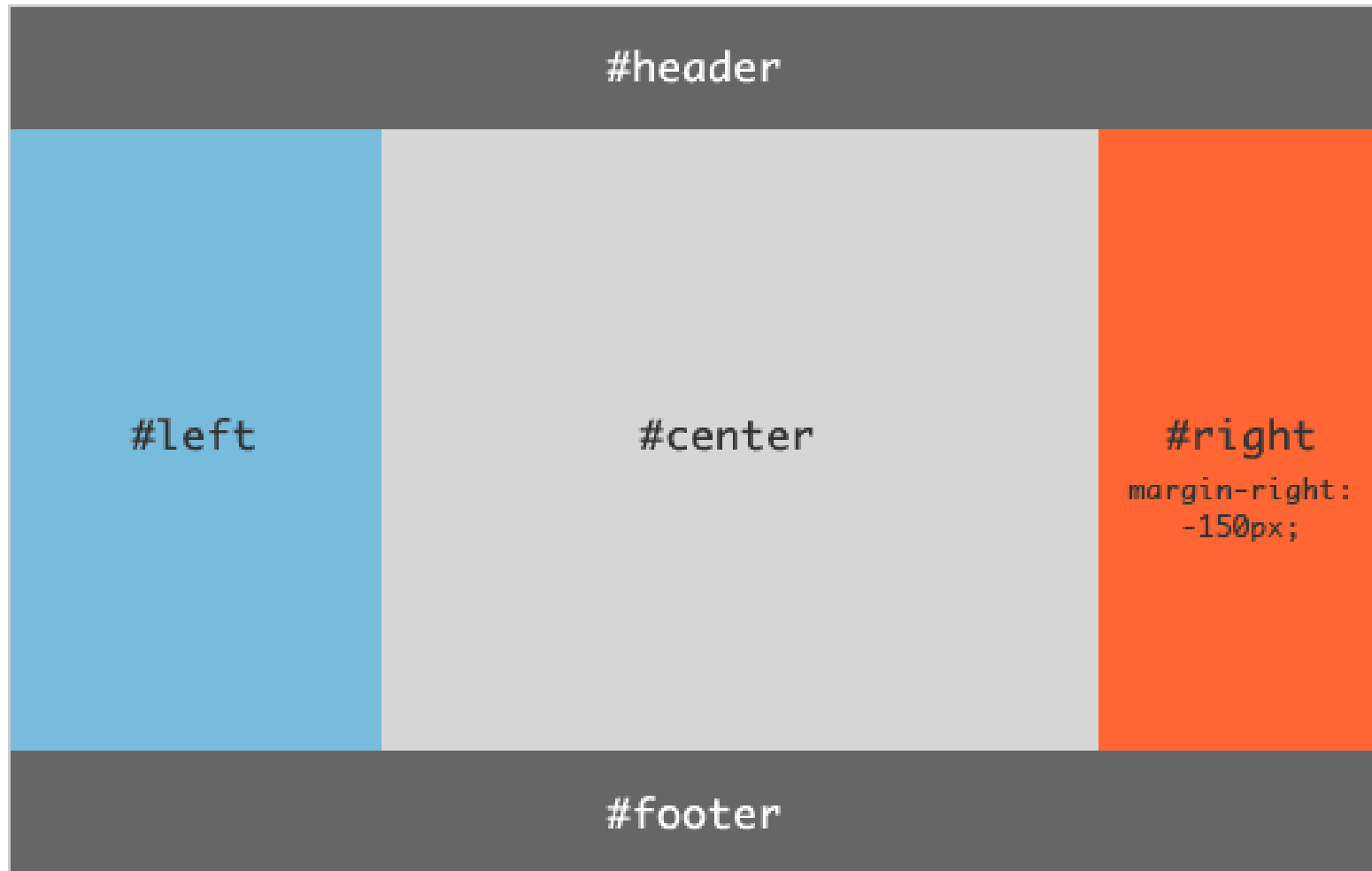
3 Columns Liquid Layout (4/7)



3 Columns Liquid Layout (5/7)



3 Columns Liquid Layout (6/7)





3 Columns Liquid Layout (7/7)

```
body {
    min-width: 630px;
    /* 2x (LC fullwidth +CC padding) +
    RC fullwidth */
}
#container {
    padding-left: 200px; /* LC
fullwidth */
    padding-right: 190px; /* RC
fullwidth + CC padding */
}
#container .column {
    position: relative;
    float: left;
}
#center {
    padding: 10px 20px; /* CC padding
*/
    width: 100%;
}

#left {
    width: 180px; /* LC width
*/
    padding: 0 10px; /* LC
padding */
    right: 240px; /* LC
fullwidth + CC padding */
    margin-left: -100%;
}
#right {
    width: 130px; /* RC width
*/
    padding: 0 10px; /* RC
padding */
    margin-right: -190px; /* RC
fullwidth + CC padding */
}
#footer {
    clear: both;
}

/**** IE Fix ****/
* html #left {
    left: 150px; /* RC
fullwidth */
}
```