

CSS: Cascading Style Sheets

**BASICS, SELECTORS AND
PAGE LAYOUT**



**POLITECNICO
DI TORINO**

Laura Farinetti - DAUIN



Summary

- Introduction
- CSS syntax
- CSS selectors
- CSS box model
- Page layout

Cascading Style Sheets

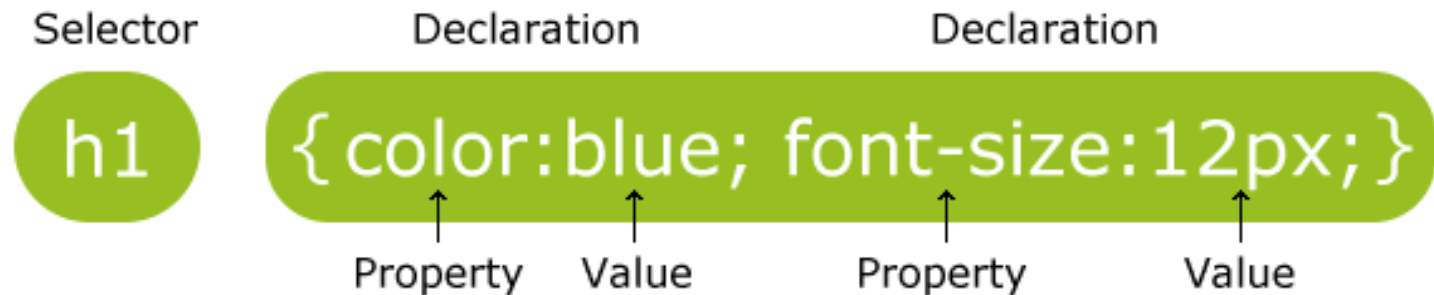
- CSS: Cascading Style Sheet
- CSS 1: W3C recommendation (17 Dec 1996)
- CSS 2.1: W3C Recommendation (7 June 2011)
- CSS 3: different stages (REC, PR, CR, WD)
 - see <http://www.w3.org/Style/CSS/current-work>
- Resources:
 - CSS 2.1 standard, <http://www.w3.org/TR/CSS21/>
 - W3C CSS Tutorial, <http://www.w3.org/Style/Examples/011/firstcss>

CSS level 3 (CSS3)

- Major change: introduction of modules
- Advantage of modules: they (supposedly) allow the specification to be completed and approved more quickly, because segments are completed and approved in chunks
 - This also allows browser and user-agent manufacturers to support sections of the specification but keep their code bloat to a minimum by only supporting those modules that make sense

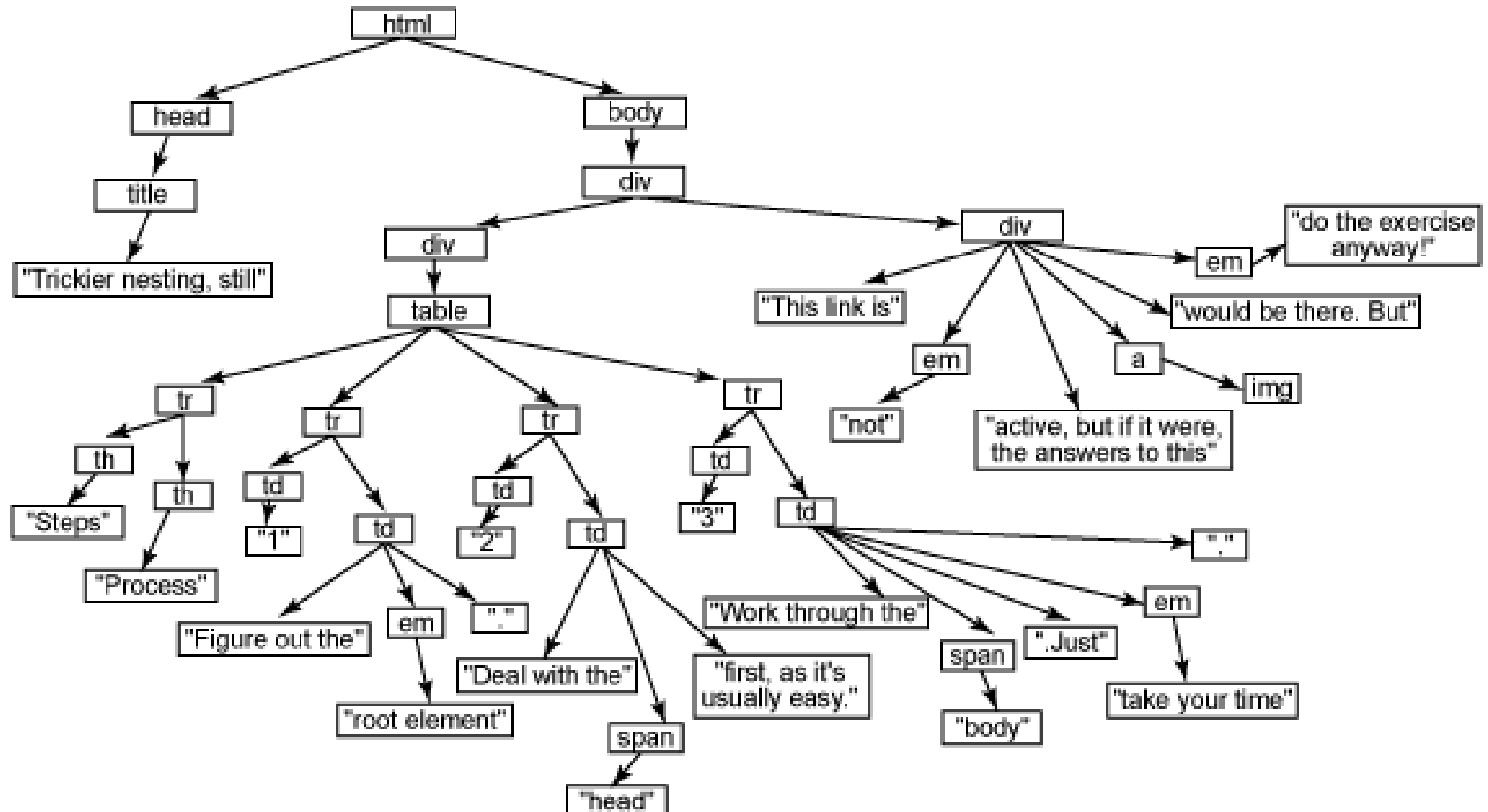
CSS Syntax

- CSS is based on rules
- A rule is a statement about one stylistic aspect of one or more HTML element
- A style sheet is a set of one or more rules that apply to an HTML document



Tree structure

- HTML documents are trees



Tree structure

```
<html lang="en">
<head>
  <title>Trickier nesting, still</title>
</head>
<body>
  <div>
    <div>
      <table>
        <tr><th>Steps</th><th>Processes</th></tr>
        <tr><td>1</td><td>Figure out the <em>root element</em>.</td></tr>
        <tr><td>2</td><td>Deal with the <span>head</span> first as it's
          usually easy.</td></tr>
        <tr><td>3</td><td>Work through the <span>body</span>. Just
          <em>take your time</em>.</td></tr>
      </table>
    </div>
    <div>
      This link is <em>not</em> active, but if it were, the answer to this
      <a></a> would be there. But <em>do the
        exercise anyway!</em>
    </div>
  </div>
</body>
</html>
```

Tree structure

```
<html lang="en">
<head>
  <title>Trickier nesting, still</title>
</head>
<body>
  <div>
    <div>
      <table>
        <tr>
          <th>Steps</th>
          <th>Processes</th>
        </tr>
        <tr>
          <td>1</td>
          <td>Figure out the root element.</td>
        </tr>
        <tr>
          <td>2</td>
          <td>Deal with the head first as it's usually easy.</td>
        </tr>
        <tr>
          <td>3</td>
          <td>Work through the body. Just take your time.</td>
        </tr>
        <tr>
          <td colspan="2">
            This link is not active, but if it were, the answer to this
            <a href="#"><img alt="exercise.jpg" data-bbox="695 515 725 580"/></a> would be there. But do the
            exercise anyway!</td>
        </tr>
      </table>
    </div>
    <div>
      This link is not active, but if it were, the answer to this
      <a href="#"><img alt="exercise.jpg" data-bbox="695 515 725 580"/></a> would be there. But do the
      exercise anyway!</div>
    </div>
  </div>
</body>
</html>
```

Steps

Processes

- 1 Figure out the *root element*.
- 2 Deal with the head first as it's usually easy.
- 3 Work through the body. Just *take your time*.



This link is *not* active, but if it were, the answer to this [exercise.jpg](#) would be there. But *do the exercise anyway!*

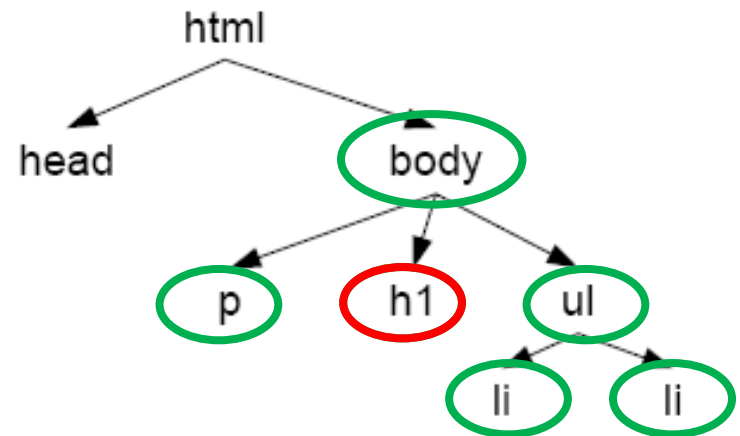
Tree structure and inheritance

- XHTML documents are trees
- Styles are inherited along trees

- When two rules are in conflict the most specific wins
- Example

– body {color: green}

– h1 {color: red}




Example

```
<style type="text/css">
  th { color:green; font-size: 24pt;}
</style>
```

Steps Processes

- 1 Figure out the *root element*.
- 2 Deal with the head first as it's usually easy.
- 3 Work through the body. Just *take your time*.

This link is *not* active, but if it were, the answer to this  would be there. But *do the exercise anyway!*

```
This link is <em style="color:red; ">not</em> active, ...
```


Example

Steps

- 1 Figure out the *root element*.
- 2 Deal with the **head** first as it's usually easy.
- 3 Work through the **body**. Just *take your time*.

Processes



This link is *not* active, but if it were, the answer to this  would be there. But *do the exercise anyway!*


```
<style type="text/css">
  th { color:green; font-size: 24pt; }
  td { text-align: center; }
  span { font-weight: bold; }
</style>
```

Example

Steps

- 1 Figure out the *root element*.
- 2 Deal with the **head** first as it's usually easy.
- 3 Work through the **body**. Just *take your time*.

Processes

This link is *not* active, but if it were, the answer to this  would be there. But *do the exercise anyway!*



does not change

```
<style type="text/css">
  th { color:green; font-size: 24pt; }
  td { text-align: center; }
  span { font-weight: bold; }
  em { color:brown; }
</style>
```

CSS properties

- <http://www.w3schools.com/cssref/>

CSS Property Groups

- [Color](#)
- [Background and Borders](#)
- [Basic Box](#)
- [Flexible Box](#)
- [Text](#)
- [Text Decoration](#)
- [Fonts](#)
- [Writing Modes](#)
- [Table](#)
- [Lists and Cour](#)
- [Animation](#)
- [Transform](#)
- [Transition](#)
- [Basic User Int](#)
- [Multi-column](#)

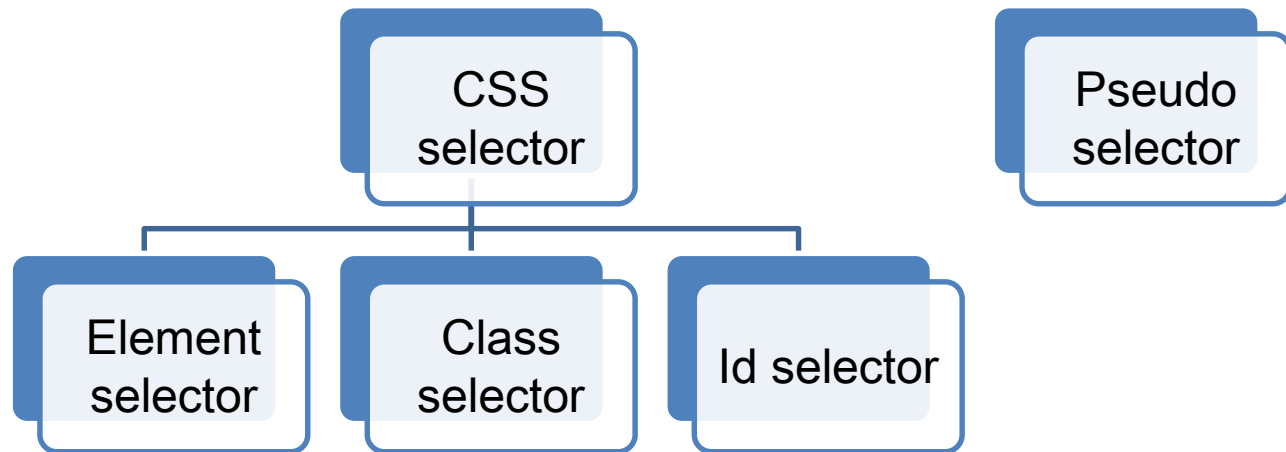
Color Properties

Property	Description	CSS
color	Sets the color of text	1
opacity	Sets the opacity level for an element	3

Default value:	<i>not specified</i>
Inherited:	yes
Animatable:	yes. Read about animatable
Version:	CSS1
JavaScript syntax:	<code>object.style.color="#0000FF"</code>

CSS selectors

- Three types of selectors plus “pseudo-selectors”



Element selector

- Used to apply the same style to all instance of a specific element in a document
- In an element selector you choose an element with its tag name and apply the style on that
- Example: apply the color red to all h1 elements that appear in the document

```
h1
{
    color : red;
}
```

style.css

Class selector

- Used to apply the same style to all elements belonging to a specific (defined) class
- Usually you use the class selector when you want to apply a specific style to a different set of elements
- Example: apply the color blue style to various elements on the document
 - Declare a class name for all elements that must be blue and then select them on the basis of that class name

doc.html

```
<h1 id="titlemessage">Education is must</h1>
  <p class="bluetext">Education in its general sense is a
    form of learning in which [...]
  </p>
<h3 class="bluetext">Try to teach</h3>
```

```
.bluetext
{
    color:blue;
}
```

stile.css

Id selector

- Used to apply a style to a specific element in a document
- You can select an element by its (declared) id and apply a style to that
- Example: apply the color grey to the “titlemessage” h1 element

doc.html

```
<h1 id="titlemessage">Education is must</h1>
  <p class="bluetext">Education in its general sense is a
    form of learning in which [...]
  </p>
<h3 class="bluetext">Try to teach</h3>
```

```
#titlemessage
{
    color : gray
}
```

stile.css

Pseudo class selector

- Used to style an element based on something other than the structure of the document
 - E.g., the status of a form element or link

```
/* makes all unvisited links blue */  
a:link {color:blue;}  
/* makes all visited links green */  
a:visited {color:green;}  
/* makes links red when hovered or activated */  
a:hover, a:active {color:red;}  
/* makes table rows red when hovered over */  
tr:hover {background-color: red;}  
/* makes input elements yellow when focus is applied */  
input:focus {background-color:yellow;}
```

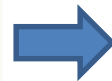
Meaningful HTML

- Meaningful elements
 - h1, h2, ...
 - ul, ol, and dl
 - strong and em
 - blockquote and cite
 - abbr, acronym, and code
 - fieldset, legend, and label
 - caption, thead, tbody, and tfoot
- Id and class names
 - Allow to give extra meaning
- Div and span
 - Add structure to document

Div element

- Stands for “division”
- Used to group block-level elements
 - Provides a way of dividing a document into meaningful areas
- Use only if necessary and not redundant

```
<div id="mainNav">  
  <ul>  
    <li>Home</li>  
    <li>About Us</li>  
    <li>Contact</li>  
  </ul>  
</div>
```



```
<ul id="mainNav">  
  <li>Home</li>  
  <li>About Us</li>  
  <li>Contact</li>  
</ul>
```

Span element

- Used to group or identify inline elements

```
<h2>Where's Durstan?</h2>
<p>Published on
    <span class="date">March 22nd, 2005</span>
by <span class="author">Andy Budd</span></p>
```

CSS selectors

Selector	Example	Example description	CSS
<i>.class</i>	.intro	Selects all elements with class="intro"	1
<i>#id</i>	#firstname	Selects the element with id="firstname"	1
<i>*</i>	*	Selects all elements	2
<i>element</i>	p	Selects all <p> elements	1
<i>element,element</i>	div, p	Selects all <div> elements and all <p> elements	1
<i>element element</i>	div p	Selects all <p> elements inside <div> elements	1
<i>element>element</i>	div > p	Selects all <p> elements where the parent is a <div> element	2
<i>element+element</i>	div + p	Selects all <p> elements that are placed immediately after <div> elements	2
<i>element1~element2</i>	p ~ ul	Selects every element that are preceded by a <p> element	3

http://www.w3schools.com/cssref/css_selectors.asp

CSS selectors

Selector	Example	Example description	CSS
<code>[attribute]</code>	<code>[target]</code>	Selects all elements with a target attribute	2
<code>[attribute=value]</code>	<code>[target=_blank]</code>	Selects all elements with target="_blank"	2
<code>[attribute~value]</code>	<code>[title~flower]</code>	Selects all elements with a title attribute containing the word "flower"	2
<code>[attribute =value]</code>	<code>[lang =en]</code>	Selects all elements with a lang attribute value starting with "en"	2
<code>[attribute^=value]</code>	<code>a[href^="https"]</code>	Selects every <a> element whose href attribute value begins with "https"	3
<code>[attribute\$=value]</code>	<code>a[href\$=".pdf"]</code>	Selects every <a> element whose href attribute value ends with ".pdf"	3
<code>[attribute*=value]</code>	<code>a[href*="w3schools"]</code>	Selects every <a> element whose href attribute value contains the substring "w3schools"	3

CSS selectors

Selector	Example	Example description	CSS
:active	a:active	Selects the active link	1
::after	p::after	Insert something after the content of each <p> element	2
::before	p::before	Insert something before the content of each <p> element	2
:checked	input:checked	Selects every checked <input> element	3
:disabled	input:disabled	Selects every disabled <input> element	3
:empty	p:empty	Selects every <p> element that has no children (including text nodes)	3
:enabled	input:enabled	Selects every enabled <input> element	3
:first-child	p:first-child	Selects every <p> element that is the first child of its parent	2
::first-letter	p::first-letter	Selects the first letter of every <p> element	1
::first-line	p::first-line	Selects the first line of every <p> element	1

CSS selectors

Selector	Example	Example description	CSS
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent	3
:focus	input:focus	Selects the input element which has focus	2
:hover	a:hover	Selects links on mouse over	1
:in-range	input:in-range	Selects input elements with a value within a specified range	3
:invalid	input:invalid	Selects all input elements with an invalid value	3
:lang(<i>language</i>)	p:lang(it)	Selects every <p> element with a lang attribute equal to "it" (Italian)	2
:last-child	p:last-child	Selects every <p> element that is the last child of its parent	3
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent	3
:link	a:link	Selects all unvisited links	1

CSS selectors

Selector	Example	Example description	CSS
<code>:not(selector)</code>	<code>:not(p)</code>	Selects every element that is not a <code><p></code> element	3
<code>:nth-child(n)</code>	<code>p:nth-child(2)</code>	Selects every <code><p></code> element that is the second child of its parent	3
<code>:nth-last-child(n)</code>	<code>p:nth-last-child(2)</code>	Selects every <code><p></code> element that is the second child of its parent, counting from the last child	3
<code>:nth-last-of-type(n)</code>	<code>p:nth-last-of-type(2)</code>	Selects every <code><p></code> element that is the second <code><p></code> element of its parent, counting from the last child	3
<code>:nth-of-type(n)</code>	<code>p:nth-of-type(2)</code>	Selects every <code><p></code> element that is the second <code><p></code> element of its parent	3
<code>:only-of-type</code>	<code>p:only-of-type</code>	Selects every <code><p></code> element that is the only <code><p></code> element of its parent	3
<code>:only-child</code>	<code>p:only-child</code>	Selects every <code><p></code> element that is the only child of its parent	3
<code>:optional</code>	<code>input:optional</code>	Selects input elements with no "required" attribute	3
<code>:out-of-range</code>	<code>input:out-of-range</code>	Selects input elements with a value outside a specified range	3

CSS selectors

Selector	Example	Example description	CSS
:read-only	input:read-only	Selects input elements with the "readonly" attribute specified	3
:read-write	input:read-write	Selects input elements with the "readonly" attribute NOT specified	3
:required	input:required	Selects input elements with the "required" attribute specified	3
:root	:root	Selects the document's root element	3
::selection	::selection	Selects the portion of an element that is selected by a user	
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)	3
:valid	input:valid	Selects all input elements with a valid value	3
:visited	a:visited	Selects all visited links	1

Selectors

- Examples of new CSS3 pseudo-classes

```
:nth-child(3n)
```



```
:nth-child(3n+2)
```



```
:nth-child(-n+4)
```



Selectors

- N = pseudoclass expressions

n	$2n+1$	$4n+1$	$4n+4$	$4n$	$5n-2$	$-n+3$
0	1	1	4	-	-	3
1	3	5	8	4	3	2
2	5	9	12	8	8	1
3	7	13	16	12	13	-
4	9	17	20	16	18	-
5	11	21	24	20	23	-

Selectors

- Examples of new CSS3 pseudo-classes

odd = $2n+1$
even = $2n$

★★★★★	Akiko's Restaurant
★★★★★	Warakubune Sushi Restaurant
★★★★★	Sushi Zone
★★★★☆	Kabuto Sushi
★★★★☆	Sushi Bistro
★★★★☆	Okina Sushi
★★★★☆	Sushi Raw
★★★★☆	Okoze Sushi
★★★★★	Red Box Sushi
★★★☆☆	Sushi Time

```
tr:nth-child(odd) td  
{ background: #ecffd9; }
```

★★★★★	Akiko's Restaurant
★★★★★	Warakubune Sushi Restaurant
★★★★★	Sushi Zone
★★★★☆	Kabuto Sushi
★★★★☆	Sushi Bistro
★★★★☆	Okina Sushi
★★★★☆	Sushi Raw
★★★★☆	Okoze Sushi
★★★★★	Red Box Sushi
★★★★★	Sushi Time

```
tr:nth-child(-n+3) td  
{ background: #ecffd9; }
```

Selectors

- Examples of new CSS3 pseudo-classes

WHAT IS SUSHI?

Sushi, from [Wikipedia](#), is a food made of vinegared rice with various fillings including fish (cooked or uncooked) and vegetables. The word sushi can also come to refer to a complete dish with rice and fish.

The original word Japanese: sushi, written in kanji, means “sushi”. Outside of Japan, sushi is sometimes misunderstood to mean sashimi. In Japan, sliced raw fish alone is called sashimi and is distinct from sushi.

There are various types of sushi: sushi served rolled up in a nori seaweed sheet called makizushi or rolls; sushi made with toppings laid over a bed of rice called nigirizushi; and a small pouch of fried tofu called inarizushi; and to

```
:not (:first-child)
{ font-size: 75%; }
```

```
p:first-of-type
{ background: #fafcf5;
  font-size: 1.3em;
  color: #030;
}
```

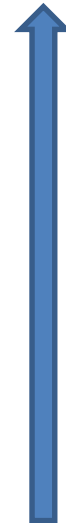
Selectors

- Pseudo-elements
 - `::selection`
Targets elements that have been highlighted by the user
- Example: match any user-selected text within a `textarea` element

```
textarea::selection
```


Cascading Style Sheets

- The term “cascading” means that a document can include more than one style sheet
- In this case, visualization follows priority rules
 - User Style
 - Inline Style (inside HTML tag)
 - Internal Style (usually in the HTML head section)
 - External Style
 - Browser Default Style



External style

- Link to an external style sheet using the `<link>` element

```
h1 { font-size:17px;
     font-family:verdana; color:green; }
h2 { font-size:18px;
     font-family:arial; color:red; }
```

stile.css

```
<head>
<link rel=stylesheet type="text/css"
      href="stile.css">
</head>
<body>
<h1>Questo testo e' di colore verde, e utilizza il
font verdana a 17 pixel</h1>
<h2>Questo testo e' di colore rosso, e utilizza il
font arial a 18 pixel</h2>
</body>
```

External style

- Alternative method
- `@import` directive in the `<style>` element

```
<head>
  <style>
    @import url(stile.css);
  </style>
</head>
<body>
  ...
</body>
```

Internal style

- `<style>` element inside the document header

```
<head>
<style type="text/css">
h1 { font-size:17px; font-family:verdana;
color:green; }
h2 { font-size:18px; font-family:arial;
color:red; }
</style>
</head>
```

Inline style

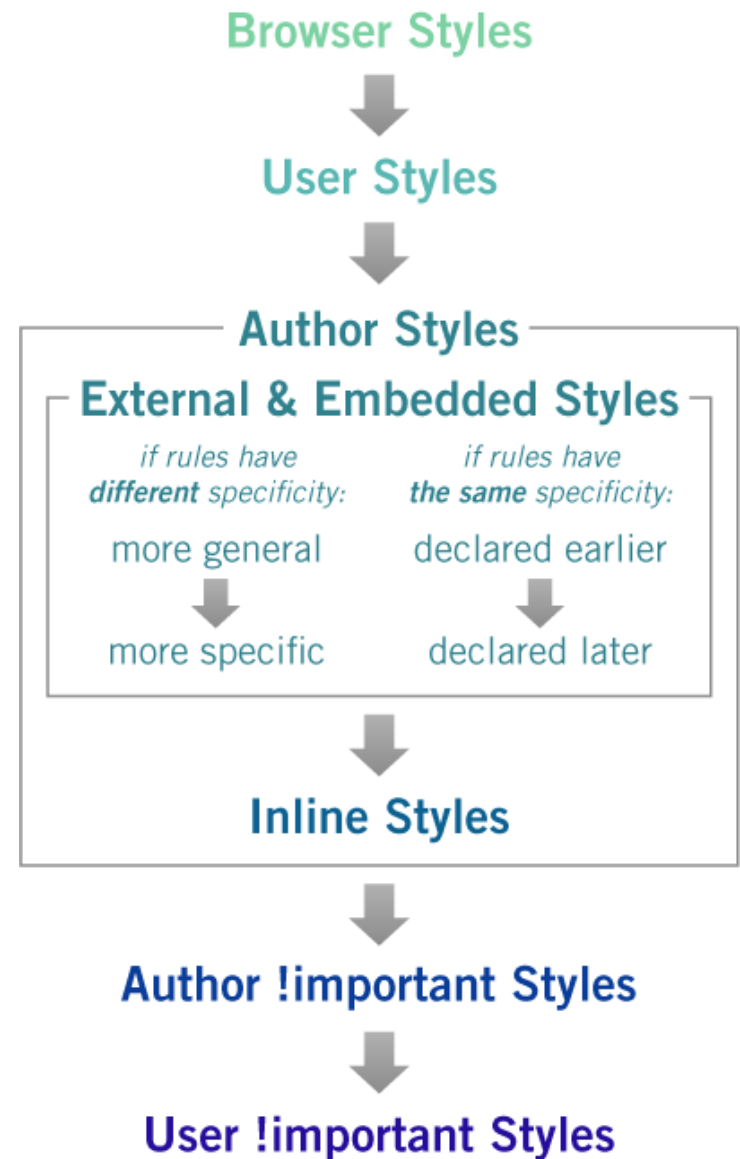
- `<style>` attribute within an HTML element

```
<h1 style="font-size:17px;
font-family:verdana; color:green; "> Questo
testo e' di colore verde, e utilizza il
font verdana a 17 pixel </h1>
```

Priority rules

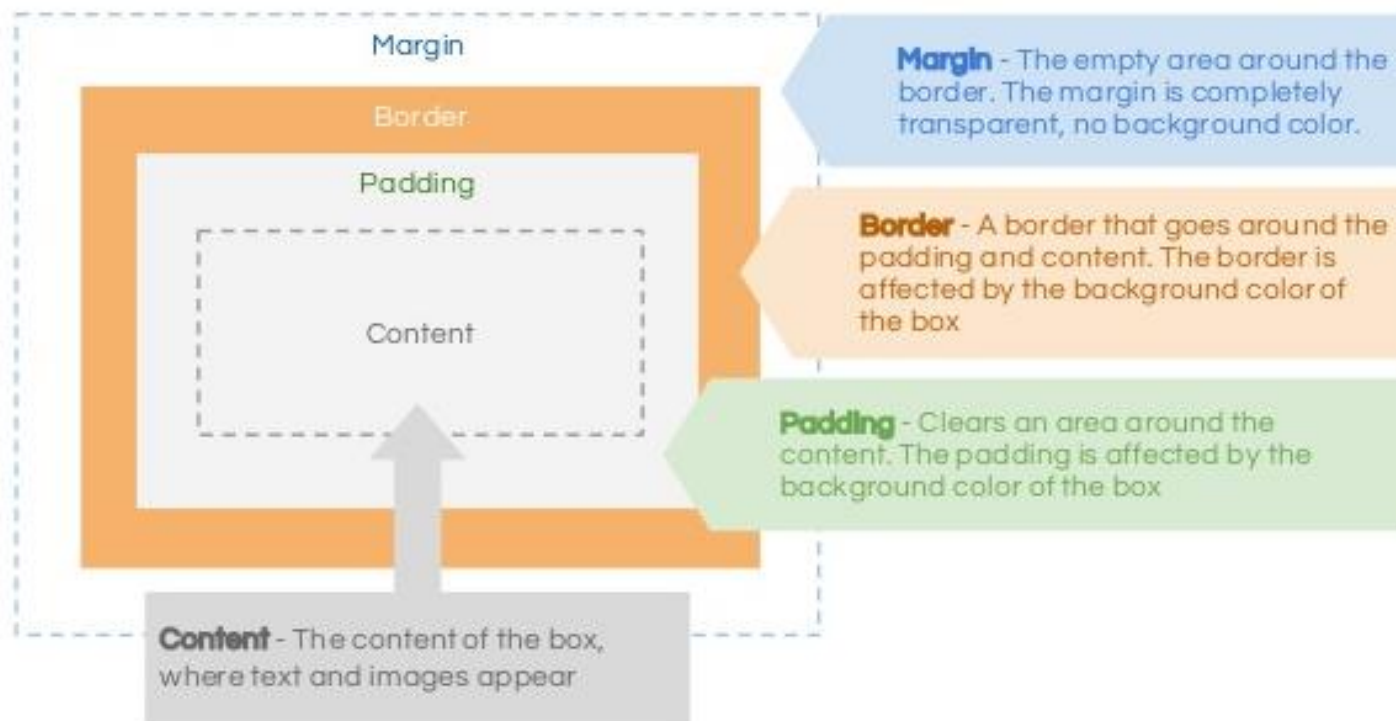
- Rules can be marked as “important”

```
h1 {  
  color:red !important  
}
```



The box model

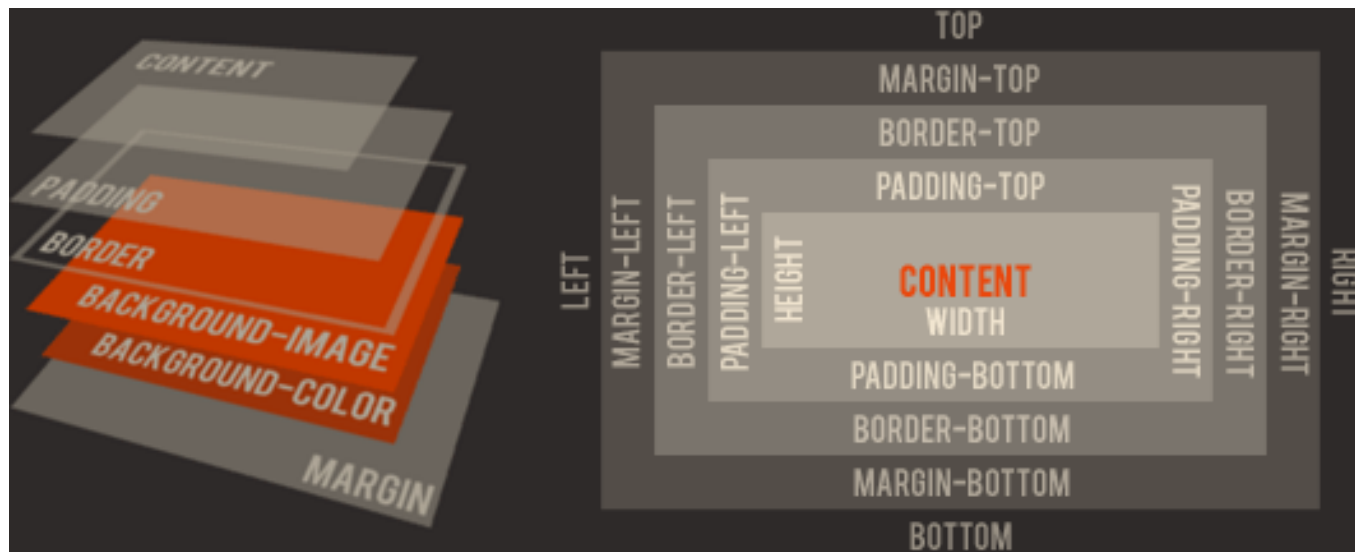
- One of the cornerstones of CSS
- Every element on the page is considered to be a rectangular box



The box model

http://www.w3schools.com/Css/css_boxmodel.asp

- Total element width = width + left padding + right padding + left border + right border + left margin + right margin
- Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

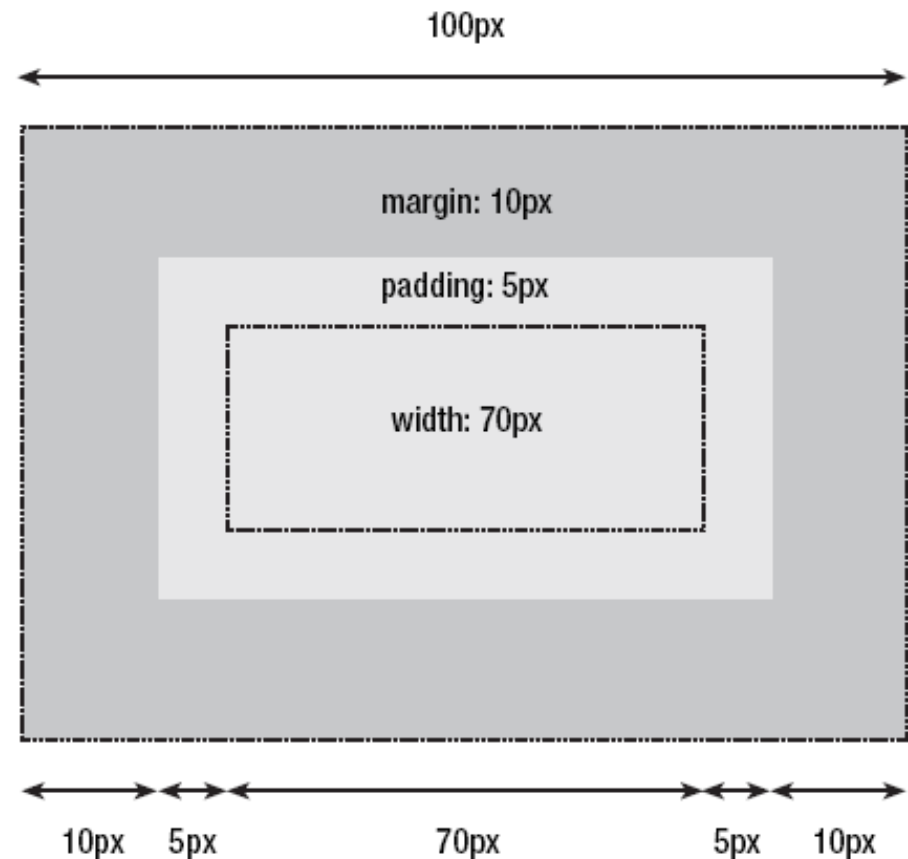


- You can set any of these properties, independently

Example

- Padding, borders, and margins are optional and default to zero

```
#myBox {  
  margin: 10px;  
  padding: 5px;  
  width: 70px;  
}
```



Positioning schemes

- In CSS there are three basic positioning schemes
 - Normal flow
 - Floats
 - Absolute positioning
- Unless specified, all boxes start life being positioned in the normal flow
 - The position of an element's box in the normal flow is dictated by that element's position in the HTML code

Normal flow

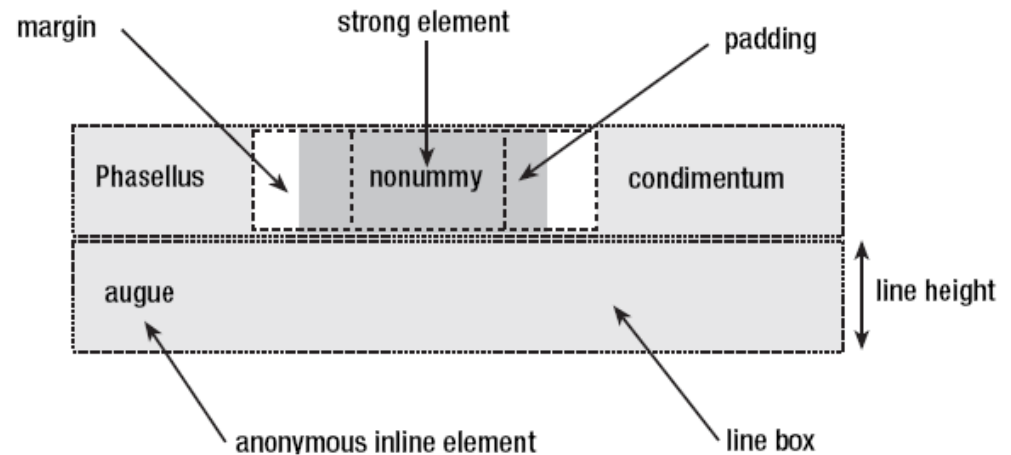
- Block-level boxes will appear vertically one after the other
 - The vertical distance between boxes is calculated by the boxes' vertical margins

```
<div> ... </div>
```

- Inline boxes are laid out in a line horizontally

```
<span> ... </span>
```

- Their horizontal spacing can be adjusted using horizontal padding, borders, and margins
- Vertical padding, borders, and margins will have no effect on the height of an inline box



Display property

- Allows to control element visualization (block or inline)
- Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way

```
li {display:inline;}
```

```
span {display:block;}
```

- Example

http://www.w3schools.com/Css/css_display_visibility.asp

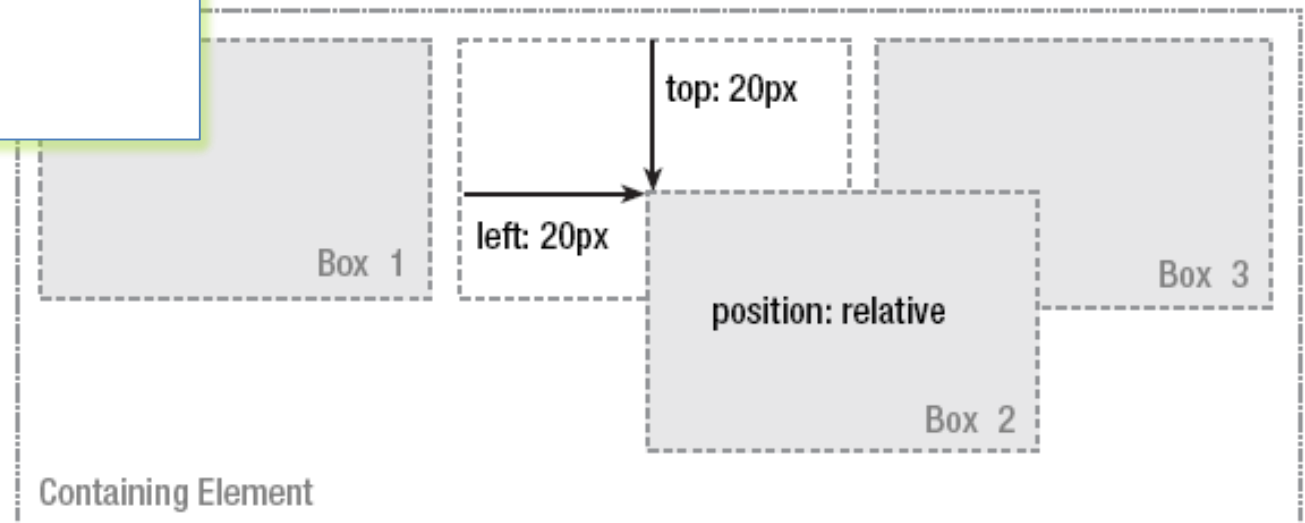
Box positioning

- A block can be positioned in different ways to which correspond different positioning schemes
 - Static: normal flow
 - Relative: the offset values are relative to the block position in the normal flow
 - 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 point (the viewport as an example)

Relative positioning

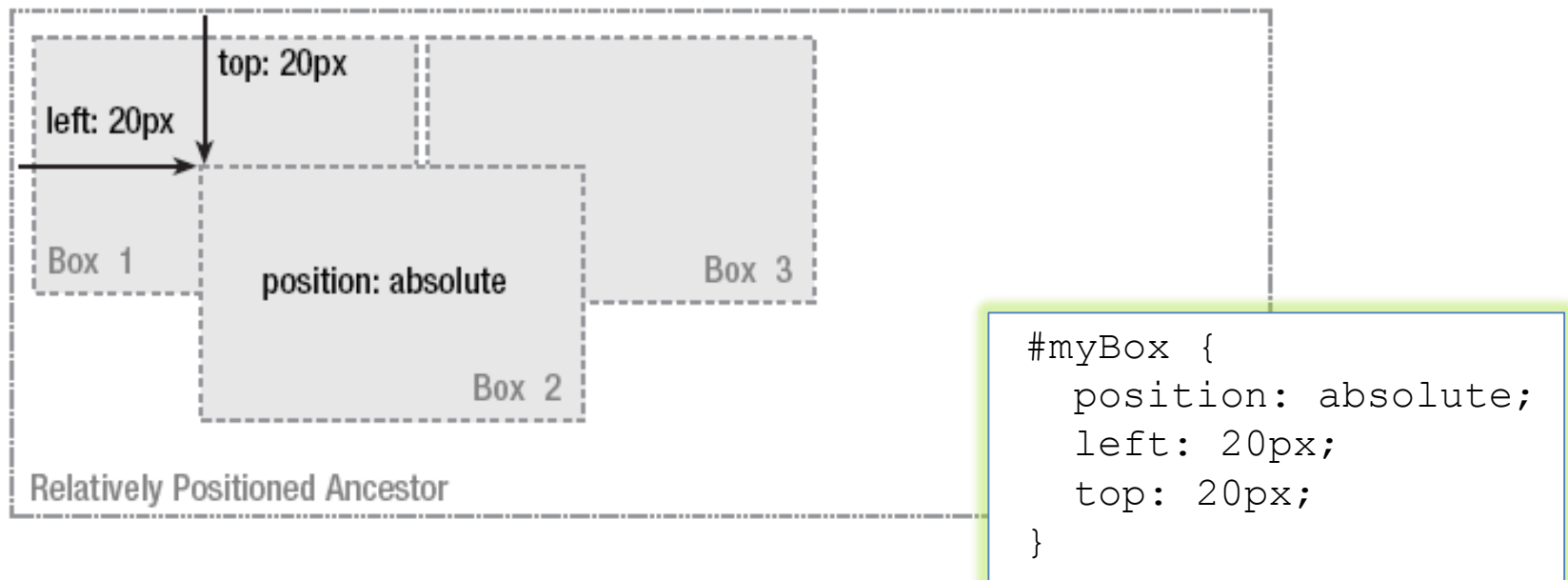
- It is possible to shift one element “relative” to its starting point by setting a vertical or horizontal offset

```
#myBox {  
  position: relative;  
  left: 20px;  
  top: 20px;  
}
```



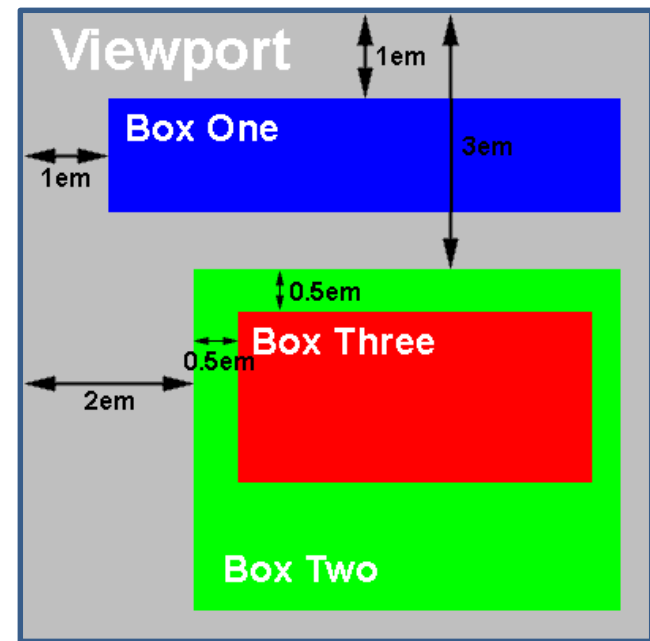
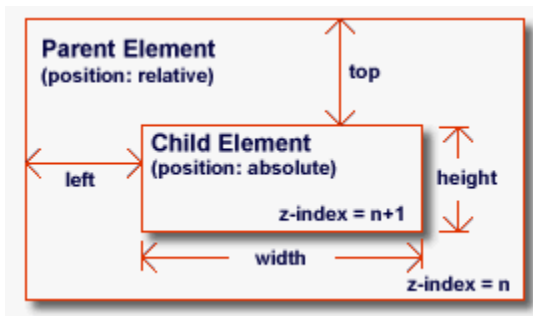
Absolute positioning

- Takes the element out of the flow of the document, thus taking up no space
- Other elements in the normal flow of the document will act as though the absolutely positioned element was never there



Fixed positioning

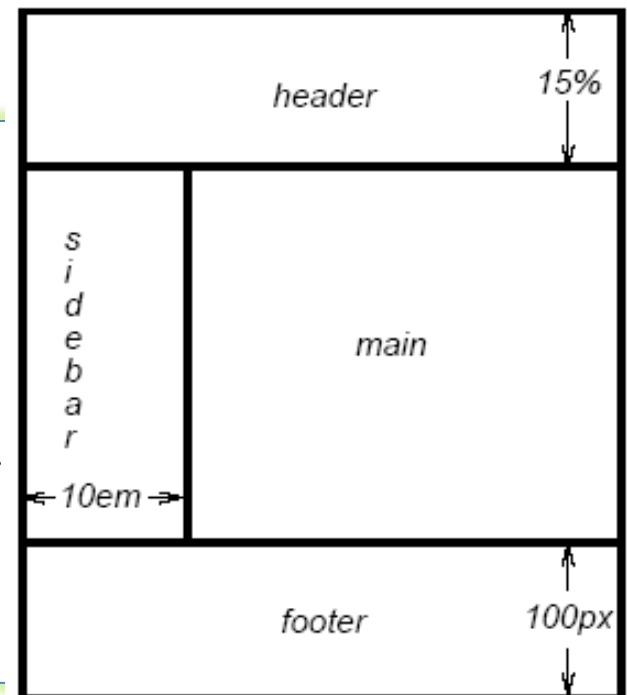
- A subcategory of absolute positioning
 - The container block is the viewport
- Allows to create elements that always stay at the same position in the window
- In case of overlaps the z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others)



Fixed positioning

- Can be used to create complex frame-like presentations

```
#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; }
```

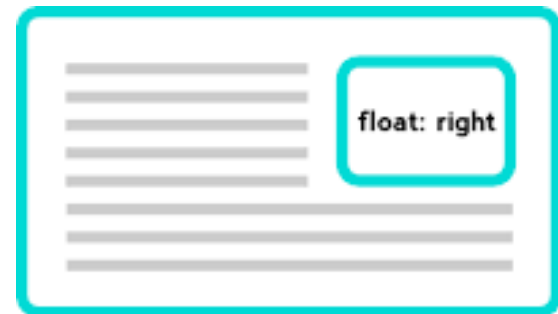


http://www.w3schools.com/Css/css_positioning.asp

Floating

- A floated box can either be shifted to the left or the right until its outer edge touches the edge of its containing box, or another floated box
- Often used for images and when working with layouts

```
img
{
float:right;
}
```

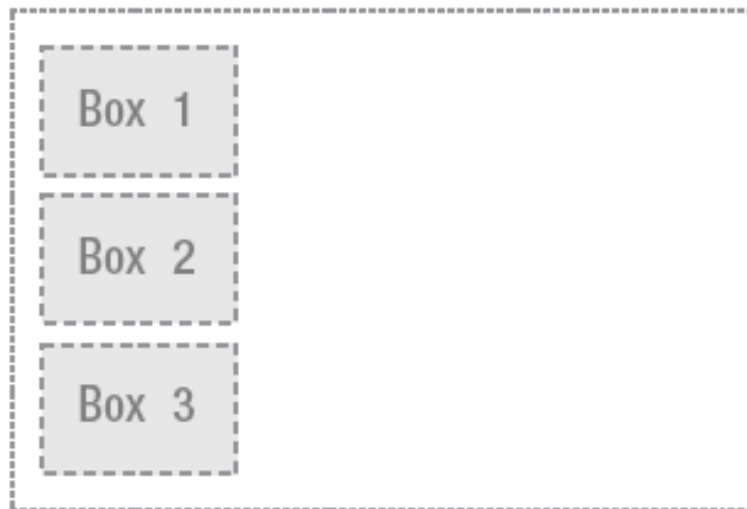


http://www.w3schools.com/Css/css_float.asp

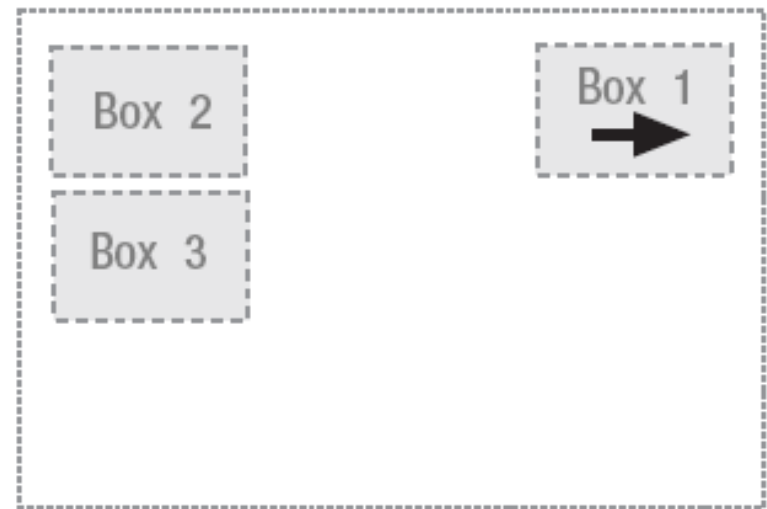
Floating

- Floated boxes aren't in the normal flow of the document, so block boxes in the regular flow of the document behave as if the floated box wasn't there

No boxes floated



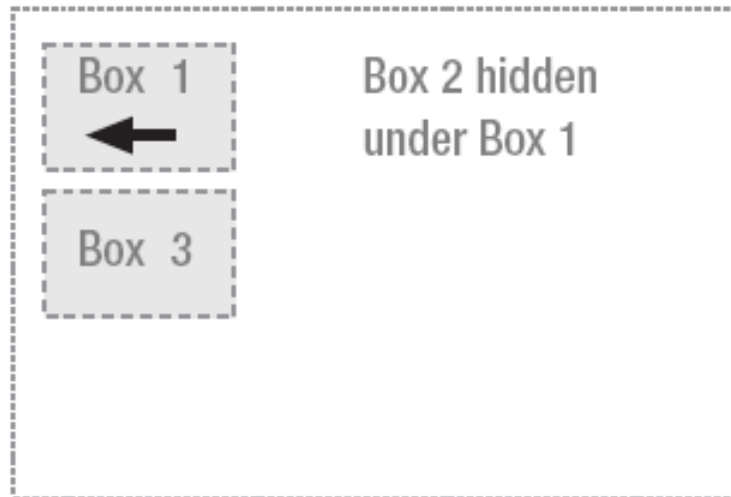
Box 1 floated right



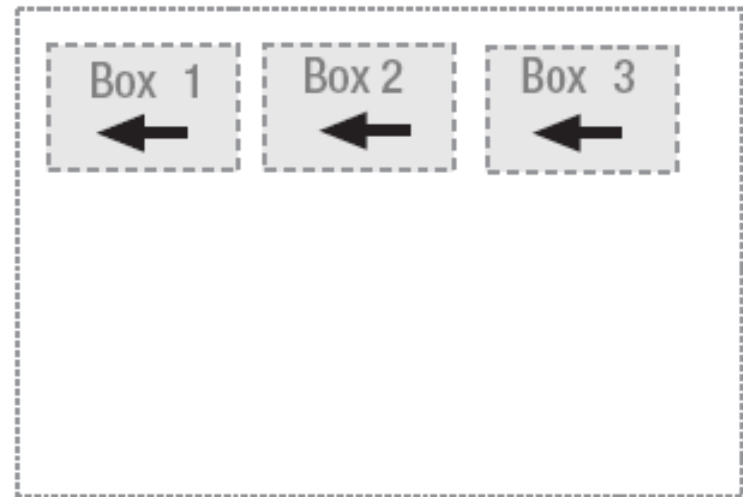
Floating

- If all three boxes are floated left
 - Box 1 is shifted left until it touches its containing box
 - Other two boxes are shifted left until they touch the preceding floated box

Box 1 floated left



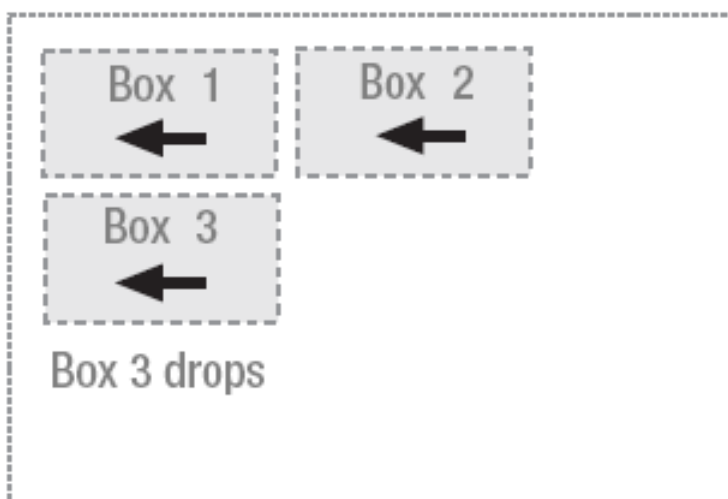
All three boxes floated left



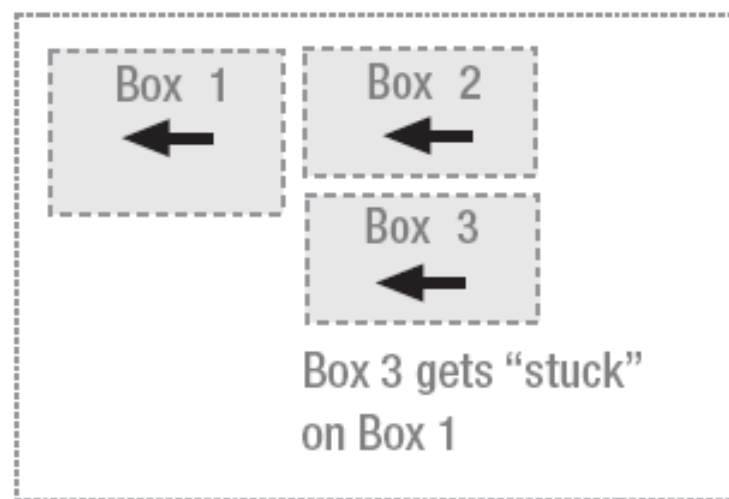
Floating

- If the containing block is too narrow for all of the floated elements to fit horizontally
 - The remaining floats will drop down until there is sufficient space
 - If the floated elements have different heights, it is possible for floats to get “stuck” on other

Not enough horizontal space



Different height boxes



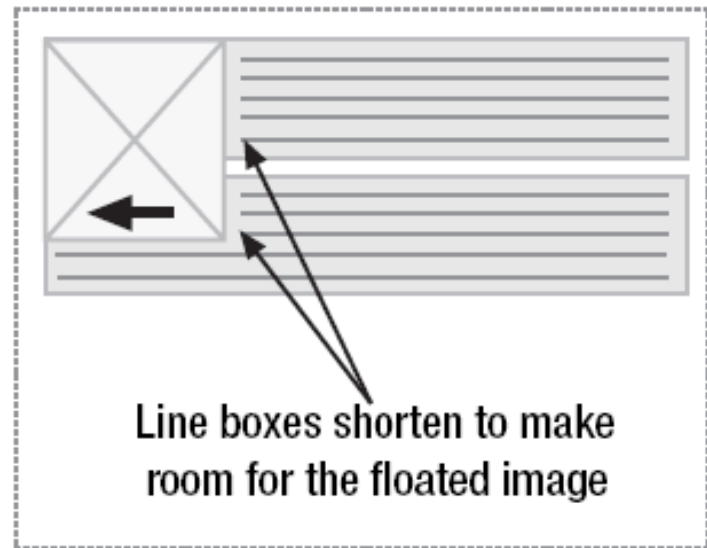
Line boxes and clearing

- Line boxes next to a floated box are shortened to make room for the floated box, and flow around the float
 - Floats were created to allow text to flow around images

No boxes floated



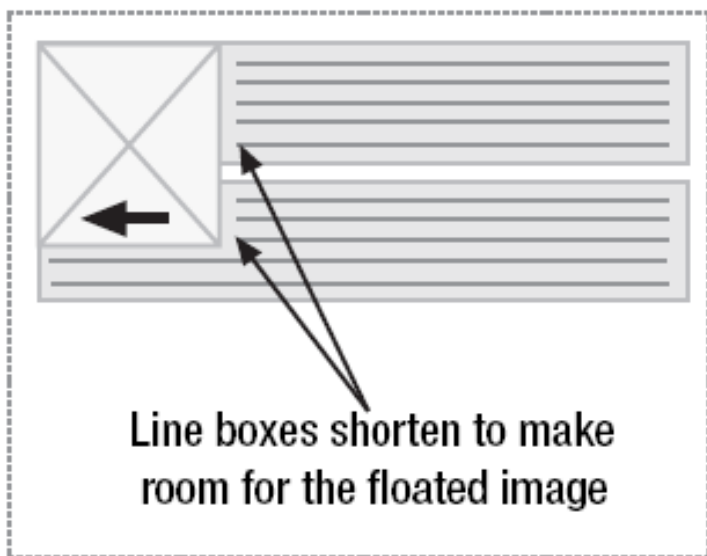
Image floated left



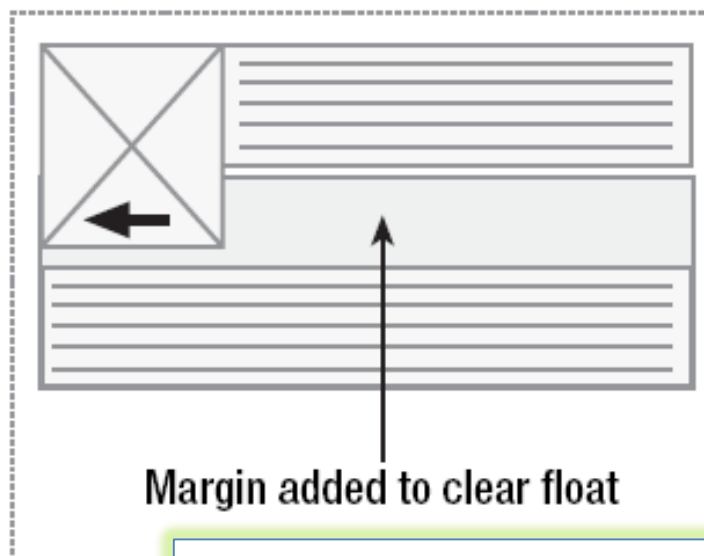
Line boxes and clearing

- To stop line boxes flowing around the outside of a floated box, you need to apply a clear to that box
 - The clear property can be left, right, both, or none, and indicates which side of the box should not be next to a floated box

Image floated left



Second paragraph cleared

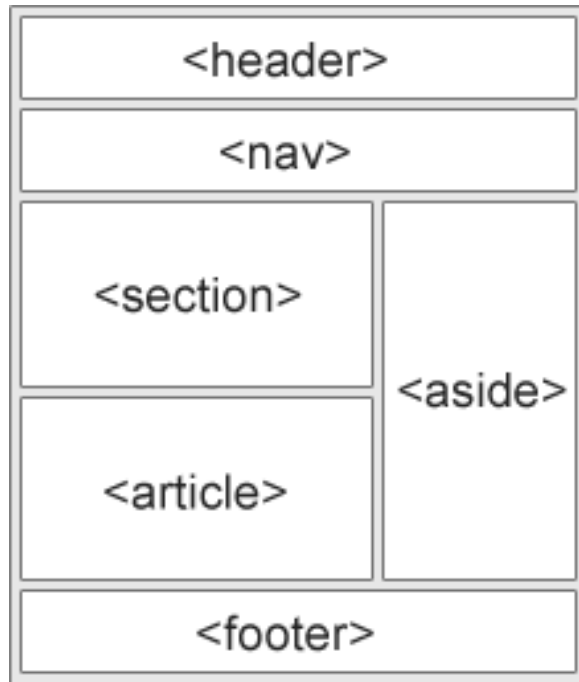


Page layout

- Possibility to control page layout without the need to use presentation markup
- Example



HTML5 semantic tags



- <header>: defines a header for a document or a section
- <nav>: defines a container for navigation links
- <section>: defines a section in a document
- <article>: defines an independent self-contained article
- <aside>: defines content aside from the content (like a sidebar)
- <footer>: defines a footer for a document or a section
- <details>: defines additional details
- <summary>: defines a heading for the <details> element

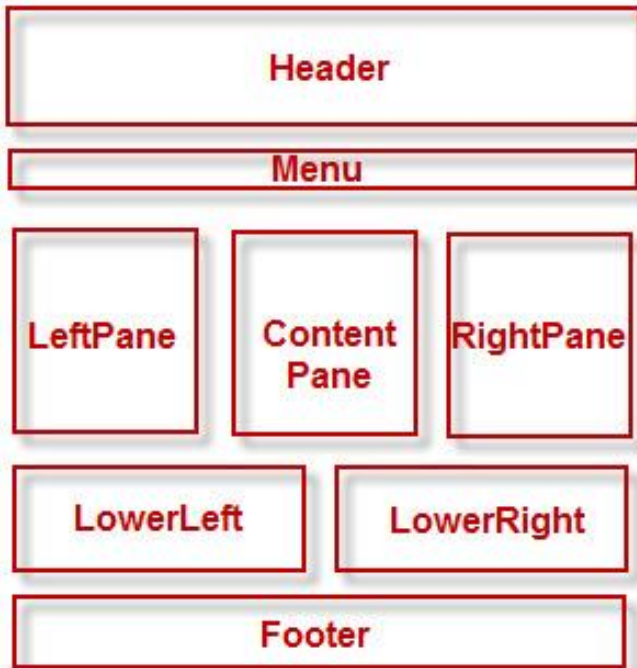
Example

```
header {
  background-color:black;
  color:white;
  text-align:center;
  padding:5px; }
nav {
  line-height:30px;
  background-color:#eeeeee;
  height:300px;
  width:100px;
  float:left;
  padding:5px; }
```

```
section {
  width:350px;
  float:left;
  padding:10px; }
footer {
  background-color:black;
  color:white;
  clear:both;
  text-align:center;
  padding:5px; }
```



Example



```
<div id="headerWrap">Header</div>
<div id="MenuWrap">Menu</div>
<div id="LeftPane">LeftPane</div>
<div id="ContentPane">ContentPane</div>
<div id="RightPane">RightPane</div>
<div id="LowerLeftPane">LowerLeft</div>
<div id="LowerRightPane">LowerRight</div>
<div id="footerWrap">Footer</div>
```

```
#headerWrap { width:100%;}
#MenuWrap { width:100%;}
#LeftPane { float:left; width:33%;}
#ContentPane { float:left; width:34%;}
#RightPane { float:left; width:33%;}
#LowerLeftPane { clear:both; float:left;
                 width:50%;}
#LowerRightPane { float:left; width:50%;}
#footerWrap { clear:both; width:100%; }
```

Multi-column layout

- How to create an horizontally centered page design
- How to create two- and three-column float-based layouts
- How to create fixed-width, liquid, and elastic layouts



Centering a design

- Long lines of text can be difficult and unpleasant to read
- Rather than spanning the full width of the screen, centered designs span only a portion of the screen, creating shorter and easier-to-read line lengths
- Two basic methods
 - Auto margins
 - Positioning and negative margins



Auto margins

- Define the width of the wrapper div
- Set the horizontal margin to auto

```
<body>  
<div id="wrapper">  
</div>  
</body>
```

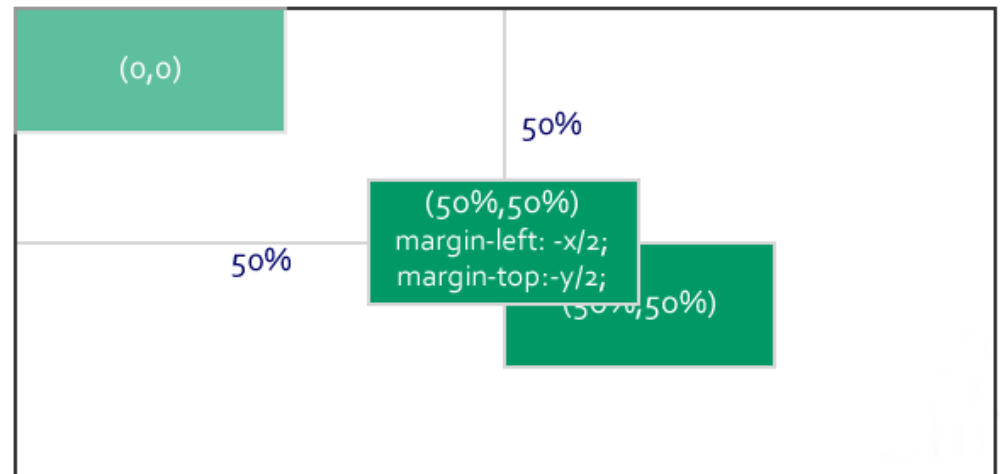
```
#wrapper {  
width: 200px;  
margin: 20px auto;  
}
```



Positioning and negative margins

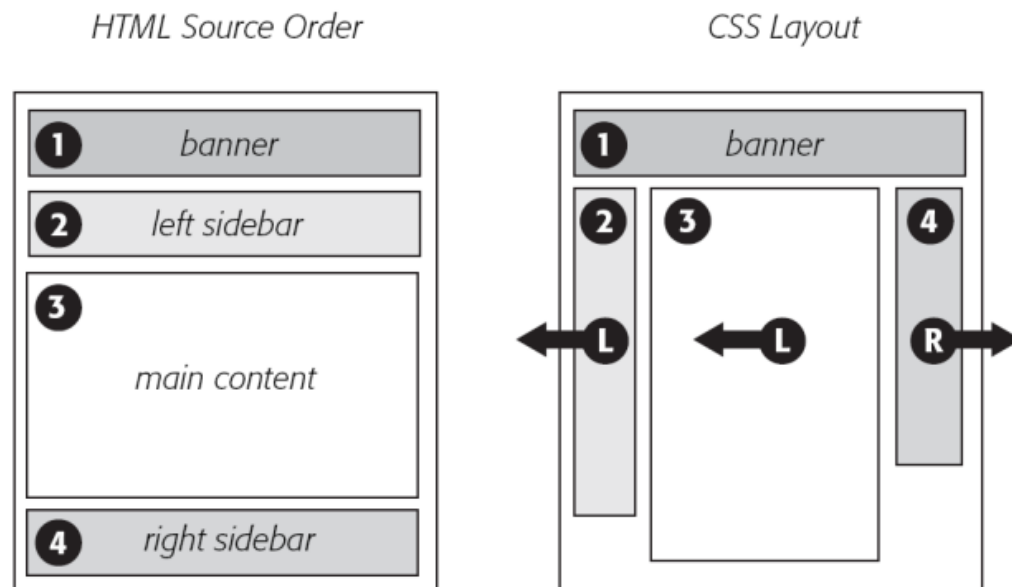
- Define the width of the wrapper div
- Set the position property of the wrapper to relative
- Set the left property to 50%
- Apply a negative margin to the left side of the wrapper, equal to half the width of the wrapper

```
#wrapper {  
width: 720px;  
position: relative;  
left: 50%;  
margin-left: -360px;  
}
```



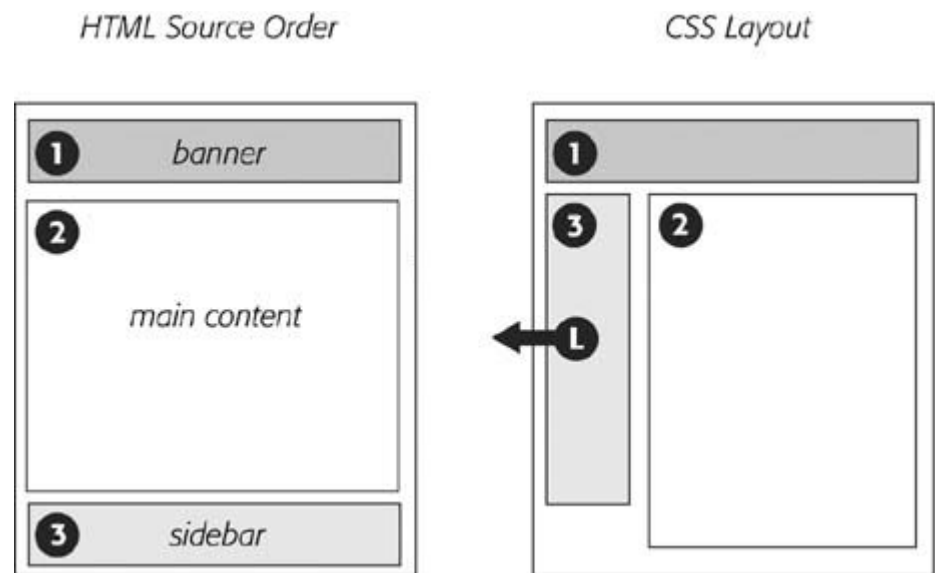
Float-based layouts

- Set the width of the elements you want to position, and then float them left or right
 - Two-column floated layout
 - Three-column floated layout



Two-column floated layout

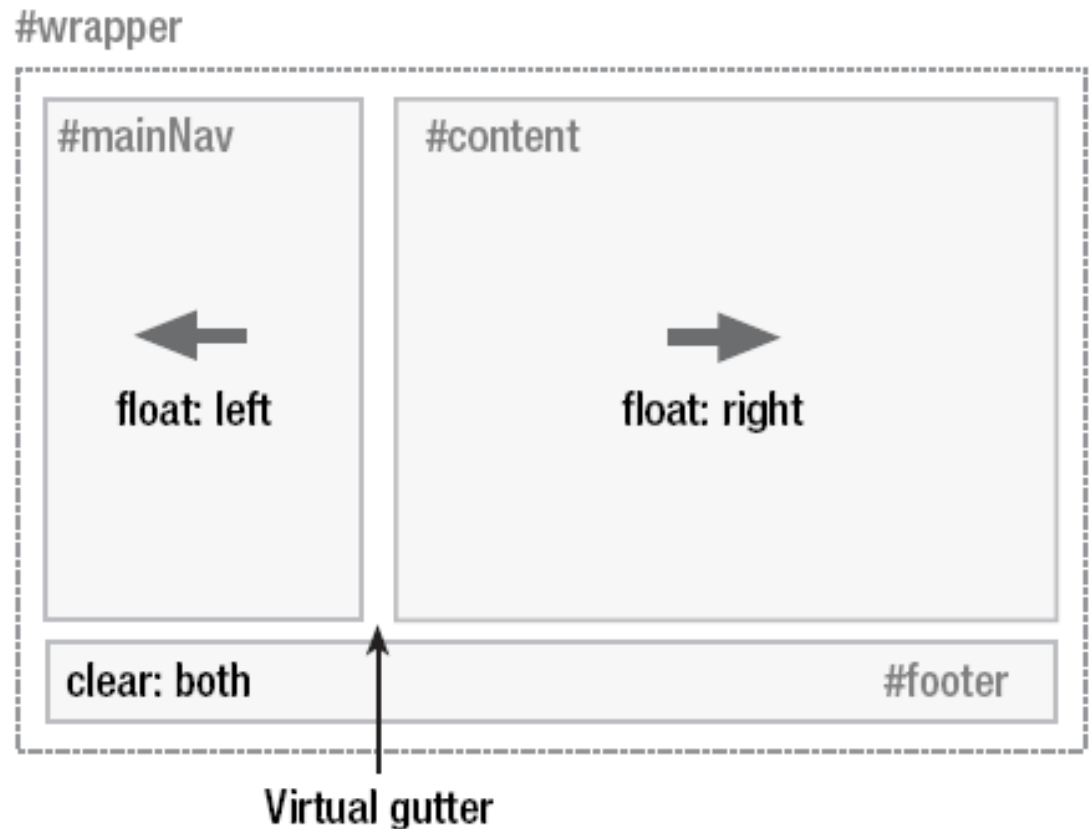
- (X)HTML framework
 - Main navigation on the left side of the page
 - Content on the right
- For accessibility reasons the content area is above the navigation in the source
 - The main content is the most important element in the page and it so should come first in the document
 - There is no point forcing screen-reader users to read through a potentially long list of links before they get to the content



Two-column floated layout

- Create a virtual gutter by floating one element left and one element right

```
<div id="wrapper">  
<div id="branding">  
...  
</div>  
<div id="content">  
...  
</div>  
<div id="mainNav">  
...  
</div>  
<div id="footer">  
...  
</div>  
</div>
```



Two-column floated layout

- Better: add horizontal padding

```
#content {  
width: 520px;  
float: right;  
}  
#mainNav {  
width: 180px;  
float: left;  
}  
#footer {  
clear: both;  
}
```

```
#mainNav {  
padding-top: 20px;  
padding-bottom: 20px;  
}  
#mainNav li {  
padding-left: 20px;  
padding-right: 20px;  
}  
#content h1, #content h2,  
    #content p {  
padding-right: 20px;  
}
```

<http://blog.html.it/layoutgala/index.html>

Three-column floated layout

- HTML framework
 - Two new divs inside the content div

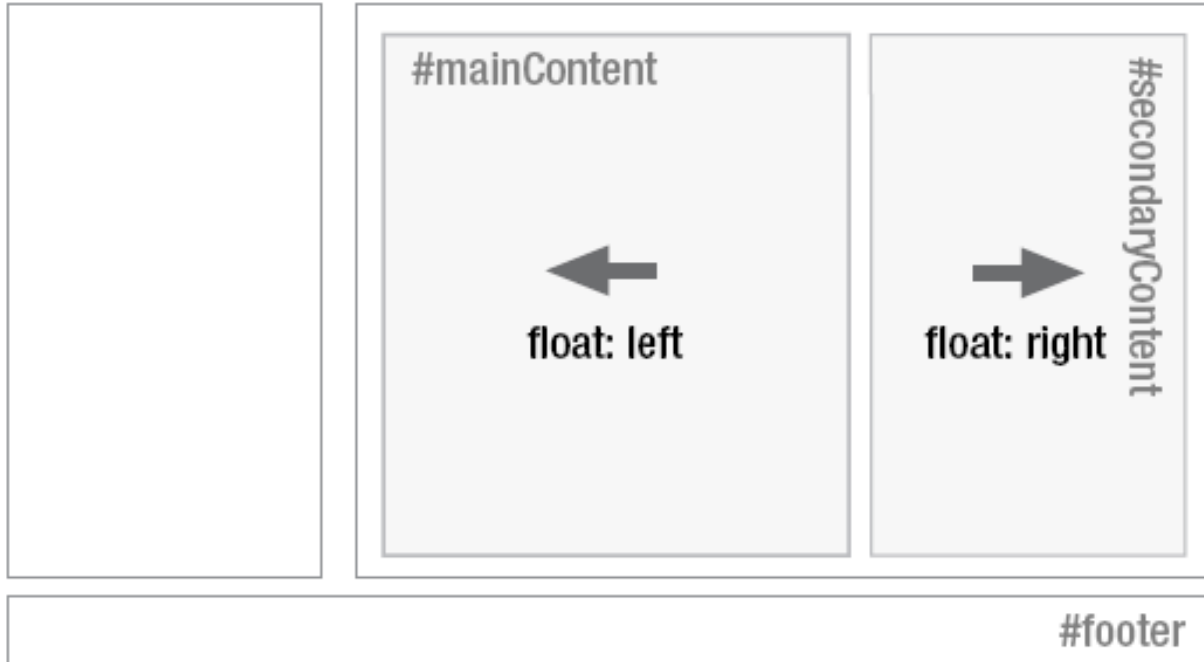
```
<div id="content">
  <div id="mainContent">
    ...
  </div>
  <div id="secondaryContent">
    ...
  </div>
</div>
```

- Float the main content left and the secondary content right, inside the already floated content div
 - Divides the second content column in two, creating a three-column effect

Three-column floated layout

#mainNav

#content

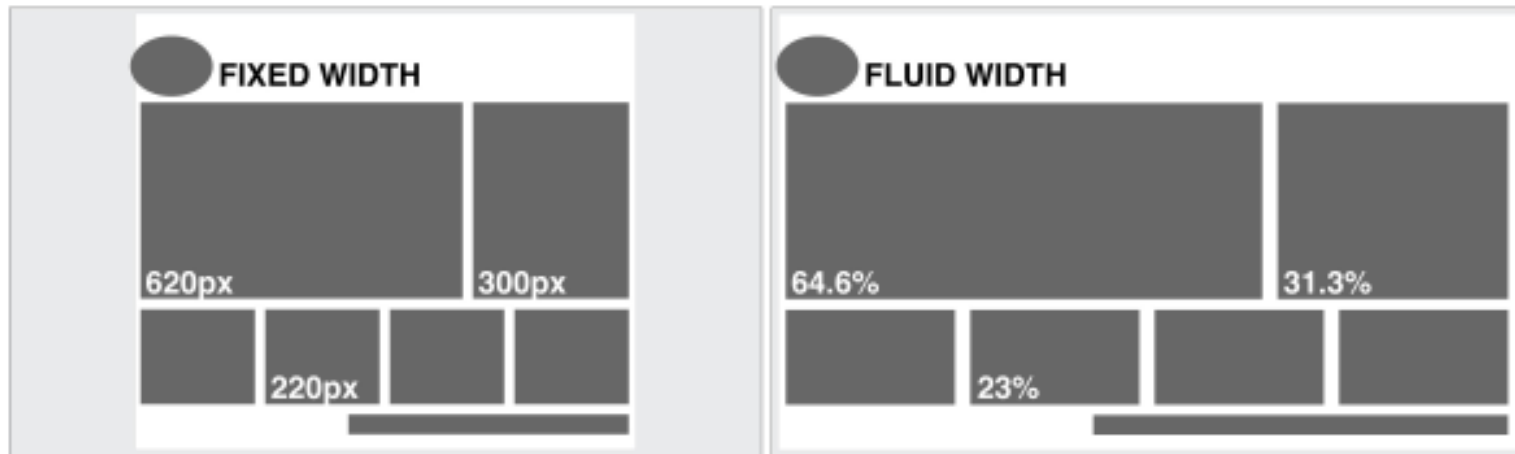


```
#mainContent {  
width: 320px;  
float: left; }  
#secondaryContent {  
width: 180px;  
float: right; }
```

```
#secondaryContent h1, #secondaryContent h2,  
#secondaryContent p {  
padding-left: 20px;  
padding-right: 20px; }
```

Fixed-width, liquid, and elastic layout

- Different ways of defining column widths
- Different behaviour: pros and cons



Fixed-width layout

- Column widths defined in pixels
- Very common: they give the developer more control over layout and positioning
- Downsides
 - Do not make good use of the available space: columns are always the same size no matter the window size
 - Usually work well with the browser default text size, but if you increase the text size a couple of steps, sidebars start running out of space and the line lengths get too short to comfortably read

Liquid layout

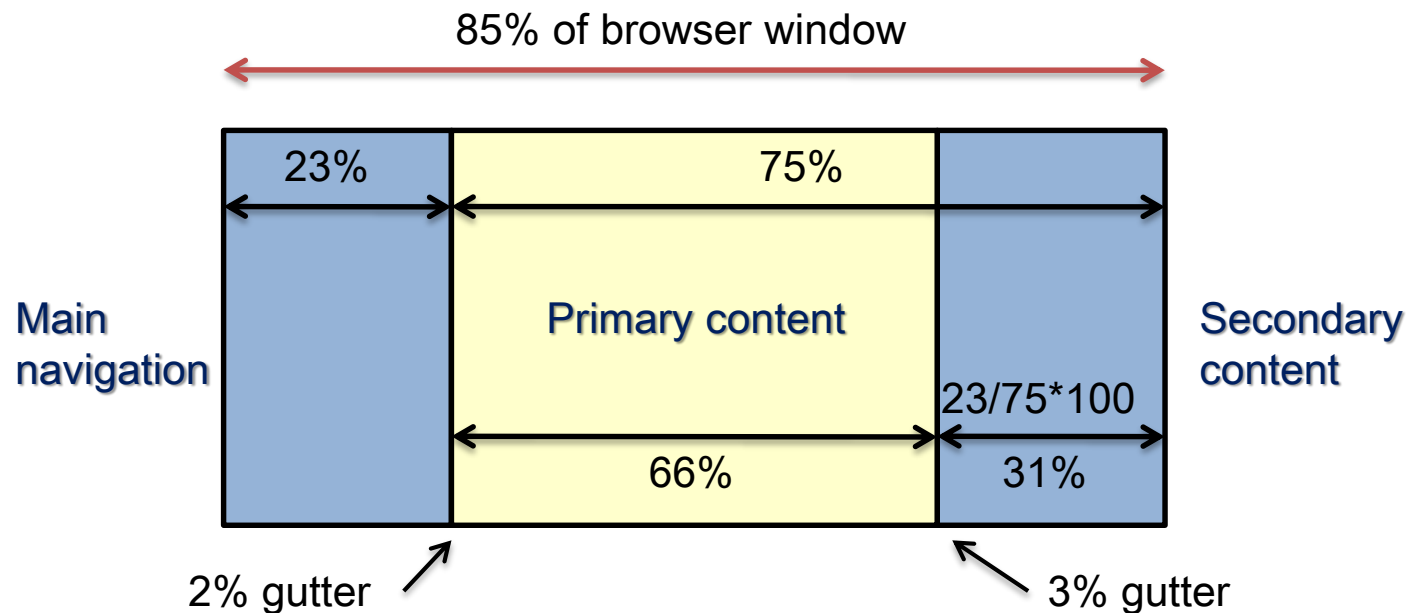
```
#wrapper {  
width: 85%;  
}
```

- Dimensions are set using percentages instead of pixels
 - Very efficient use of space
- If the design spans the entire width of the browser window, line lengths can become long and difficult to read
 - Make the wrapper span just a percentage, e.g. 85 percent
- Set the width of the navigation and content areas as a percentage of the wrapper width
 - 2-percent virtual gutter between the navigation and the wrapper to deal with any rounding errors and width irregularities that may occur

```
#wrapper {  
width: 85%;  
}  
#mainNav {  
width: 23%;  
float: left;  
}  
#content {  
width: 75%;  
float: right;  
}
```

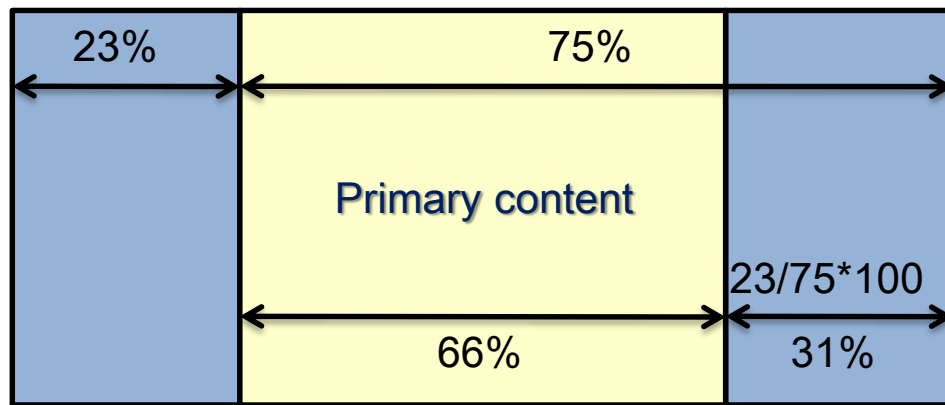

Liquid layout

- The widths of the content divs are based on the width of the content element and not the overall wrapper
 - Width of secondary content area = width of the main navigation area?



Liquid layout

- 3 columns liquid layout



```
#wrapper {  
  width: 85%;  
}  
#mainNav {  
  width: 23%;  
  float: left;  
}  
#content {  
  width: 75%;  
  float: right;  
}
```

```
#mainContent {  
  width: 66%;  
  float: left;  
}  
#secondaryContent {  
  width: 31%;  
  float: right;  
}
```

Elastic layout

- With liquid layouts
 - Line lengths can get uncomfortably long on large resolution monitors
 - Lines can become very short and fragmented in narrow windows or when the text size is increased a couple of steps
- In elastic layouts the width of elements is relative to the font size (ems) instead of the browser width
 - When the font size is increased the whole layout scales
- Allows to keep line lengths to a readable size
 - Particularly useful for people with reduced vision

Elastic layout

```
body {  
  font-size: 62.5%; }
```

- Trick to simplify design: set the base font size so that 1em roughly equals 10 pixels
 - The default font size on most browsers is 16 pixels
 - Ten pixels are 62.5 percent of 16 pixels
- Set the font size on the body to 62.5%
 - 1em now equals 10 pixels at the default font size
- Convert the fixed-width layout into an elastic layout by converting all the pixel widths to em widths

```
#wrapper {  
  width: 72em;  
  margin: 0 auto;  
  text-align: left; }  
#mainNav {  
  width: 18em;  
  float: left; }
```

```
#content {  
  width: 52em;  
  float: right; }  
#mainContent {  
  width: 32em;  
  float: left; }  
#secondaryContent {  
  width: 18em;  
  float: right; }
```

Elastic-liquid hybrid

- Combines both elastic and liquid techniques
- Works by setting the widths in ems, then setting the maximum widths as percentages
- This layout will scale relative to the font size but will never get any larger than the width of the window

```
#wrapper {  
width: 72em;  
max-width: 100%;  
margin: 0 auto;  
text-align: left;  
}  
#mainNav {  
width: 18em;  
max-width: 23%;  
float: left;  
}
```

```
#content {  
width: 52em;  
max-width: 75%;  
float: right;  
}  
#mainContent {  
width: 32em;  
max-width: 66%;  
float: left;  
}  
#secondaryContent {  
width: 18em;  
max-width: 31%;  
float: right;  
}
```

Multi-column layout

- Novelty from CSS3
- Allows to get multi-column layouts without having to use multiple divs

```
.entry-content {  
  column-count: 2;  
  column-gap: 30px; }
```

```
.entry-content {  
  column-width: 270px;  
  column-gap: 30px; }
```

WHAT IS SUSHI?

Sushi, from [Wikipedia](#), is a food made of vinegared rice, usually topped with other ingredients including fish (cooked or uncooked) and vegetables. Sushi as an English word has come to refer to a complete dish with rice and toppings; this is the sense used in this article.

The original word Japanese: sushi, written in kanji, means “snack” and

refers to the rice, but not fish or other toppings. Outside of Japan, sushi is sometimes misunderstood to mean the raw fish by itself, or even any fresh raw-seafood dishes. In Japan, sliced raw fish alone is called sashimi and is distinct from sushi.

There are various types of sushi: sushi served rolled inside nori (dried and

pressed layer sheets of seaweed or alga) called makizushi or rolls; sushi made with toppings laid with hand-formed clumps of rice called nigiri-zushi; toppings stuffed into a small pouch of fried tofu called inarizushi; and toppings served scattered over a bowl of sushi rice called chirashi-zushi.

Multi-column layout


```
div
{ grid-columns:50% * * 200px; }
```

- Add one grid line in the middle of the div element, another one 200 pixels from the right, and another one in the middle of remaining space

```
div
{ grid-rows: 100px (30px 60px); }
```

- Define a header row of 100 pixels, and add as many additional rows as necessary, alternating heights of 30 and 60 pixels

Example

0	1	2	3	4	5	
0	<h2>Grid (page layout)</h2> <p>From Wikipedia, the free encyclopedia</p> <p>A typographic grid composed of a series of intersecting vertical and horizontal axis.</p> <p>The grid in use, typography is arranged flush left, ragged right on the grid.</p> <p>A typographic grid is a two-dimensional structure made up of a series of intersecting vertical and horizontal axis used to structure content. The grid serves as an armature on which a designer can organize text and images in a rational, easy to absorb manner. The less common printing term "reference grid," is an unrelated system with roots in the early days of printing.</p> <p>Contents</p> <ul style="list-style-type: none">1 History1.1 Antecedents1.2 Evolution of the modern grid1.3 Reaction and reassessment2 References					
1	<h2>History</h2> <h3>Antecedents</h3> <p>Before the invention of movable type and printing, simple grids based on optimal proportions had been used to arrange handwritten text on pages. One such system, known as the</p>	<p>"Villard's diagram," was in use at least since medieval times.</p> <h3>Evolution of the modern grid</h3> <p>After World War I, a number of graphic designers, including Max Bill, Gerrit Klauder, and Josef Müller-Brockmann, influenced by the modernist ideas of Jan Tschichold (<i>Die neue Typographie</i>), began to question the relevance of the conventional page layout of the time. They began to devise a flexible system able to help designers achieve coherence in organizing the page. The result was the</p>	<p>modern typographic grid that became associated with the International Typographic Style. The seminal work on the subject, <i>DMG systems in graphic design</i> by Müller-Brockmann, helped propagate the use of the grid, first in Europe, and later in North America.</p> <h3>Reaction and reassessment</h3> <p>By the mid 1970s instruction of the typographic grid as a part of graphic design curricula had become standard in Europe, North America and much of Latin America. The graphic style of the grid was adopted as a look for corporate</p>			

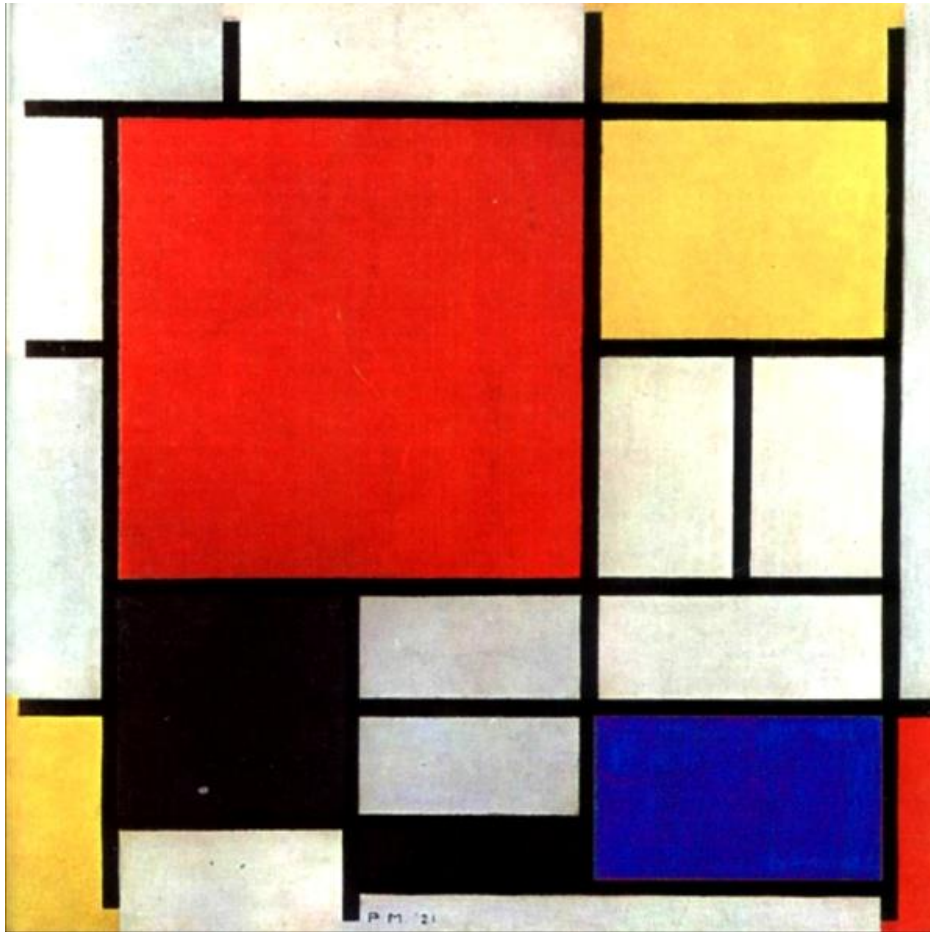
```
body { columns:3; column-gap:0.5in; }  
img { float:right top; width:3gr; }
```


Example



```
body
{ columns: 3; column-gap: 0.5in;
  grid-columns: * * (0.5in * *) [2];
  grid-rows: 20% *; }
img { float:page top left; float-offset: 4gr 1gr; }
```

Advanced layout: grid



Piet Mondrian, Composizione con grande piano rosso (1921)

Advanced layout: grid

Maki-zushi



The rice and seaweed rolls with fish and/or vegetables. There are also more specific terms for the rolls depending on the style.

Nigiri-zushi



The little fingers of rice topped with wasabi and a filet of raw or cooked fish or shellfish. Generally the most common form of sushi you will see.

Temaki-zushi



Also called a hand-roll. Cones of sushi rice, fish and vegetables wrapped in seaweed. It is very similar to maki.

WHAT IS SUSHI?

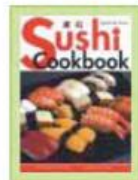
Beginning as a method of preserving fish centuries ago, sushi has evolved into an artful, unique dining experience. In its earliest form, dried fish was placed between two pieces of vinegared rice as a way of making it last. The nori (seaweed) was added later as a way to keep one's fingers from getting sticky.

Sashimi



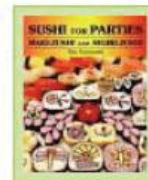
Sashimi is raw fish served sliced, but as-is. That means no rice bed or roll, but it is often served

alongside daikon and/or shiso. This is my favorite style as you really get the flavor of the fish..



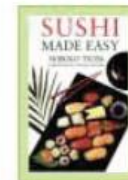
QUICK & EASY SUSHI COOKBOOK

This book has great pictures, however it is not as complete as Sushi Made Easy.



SUSHI FOR PARTIES: MAKI-ZUSHI AND NIGIRI-ZUSHI

This book also has great pictures, with advanced maki (cut roll) making techniques.



SUSHI MADE EASY

A very decent all-around book for the money.

Advanced layout: grid

<h2>Maki-zushi</h2>  <p>The rice and seaweed rolls with fish and/or vegetables. There are also more specific terms for the rolls depending on the style.</p>		<h2>Nigiri-zushi</h2>  <p>The little fingers of rice topped with wasabi and a filet of raw or cooked fish or shellfish. Generally the most common form of sushi you will see.</p>	<h2>Temaki-zushi</h2>  <p>Also called a hand-roll. Cones of sushi rice, fish and vegetables wrapped in seaweed. It is very similar to maki.</p>
<h2>WHAT IS SUSHI?</h2> <p>Beginning as a method of preserving fish centuries ago, sushi has evolved into an artful, unique dining experience. In its earliest form, dried fish was placed between two pieces of vinegared rice as a way of making it last. The nori (seaweed) was added later as a way to keep one's fingers from getting sticky.</p>		<p>Technically, the word "sushi" refers to the rice, but colloquially, the term is used to describe a finger-sized piece of raw fish or shellfish on a bed of vinegared rice or simply the consumption of raw fish in the Japanese style (while sushi is not solely a Japanese invention, these days, the Japanese style is considered the de facto serving standard).</p>	
<h2>Sashimi</h2>  <p>Sashimi is raw fish served sliced, but as-is. That means no rice bed or roll, but it is often served alongside daikon and/or shiso. This is my favorite style as you really get the flavor of the fish..</p>	 <h3>QUICK & EASY SUSHI COOKBOOK</h3> <p>This book has great pictures, however it is not as complete as Sushi Made Easy.</p>	 <h3>SUSHI FOR PARTIES</h3> <h4>MAKI-ZUSHI AND NIGIRI-ZUSHI</h4> <p>This book also has great pictures, with advanced maki (cut roll) making techniques.</p>	 <h3>SUSHI MADE EASY</h3> <p>A very decent all-around book for the money.</p>

Advanced layout: grid

- It is possible to define a grid in which content can flow or be placed, or that remain empty
- There are three ways to define a grid
 - Explicit grid: defined with 'grid-columns' and 'grid-rows' properties
 - Natural grid: automatically created by elements with a natural grid structure (multi-column elements and tables)
 - Default grid: all other block elements define a single-cell grid

Example

- Classic three-column layout

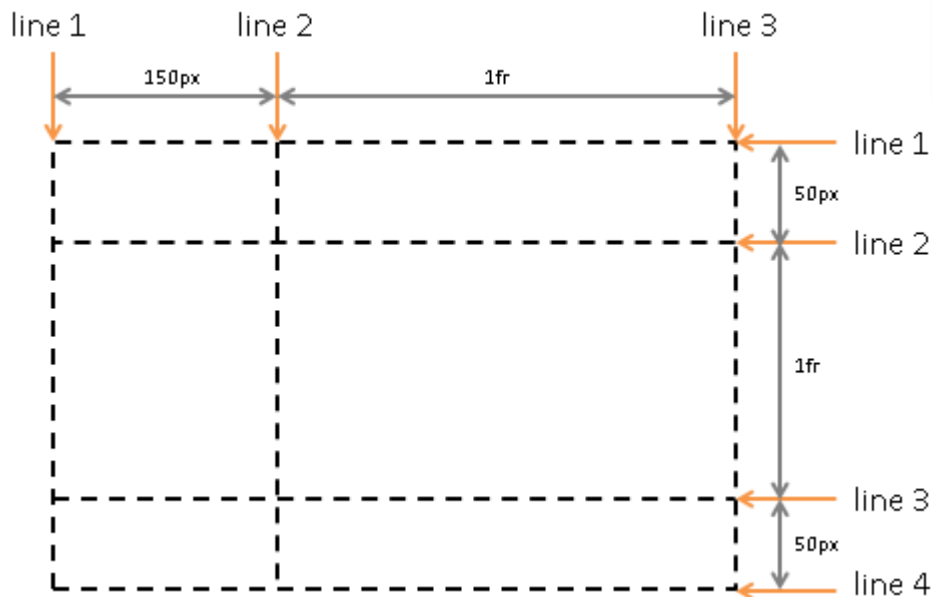
```
<section>
  <header>Title</header>
  <nav>Menu</nav>
  <article>Content</article>
  <aside>Notes</aside>
  <footer>Footer</footer>
</section>
```



Virtual grid

- First step: describe the grid tracks – rows and columns inside the grid

```
#grid {  
  grid-columns: 150px 1fr;  
  /* two columns */  
  grid-rows: 50px 1fr 50px  
  /* three rows */ }
```



Virtual grid

- Second step: to set how exactly the element will be placed on a grid it is necessary to specify to which line (both vertical and horizontal) it will be attached



```
#item {  
  grid-column: 2;  
  grid-row: 2; }
```


Example

```
section {
  display: grid;
  grid-columns: 150px 1fr 200px;
  grid-rows: 50px 1fr 50px; }
section header {
  grid-column: 2;
  grid-row: 1; }
section nav {
  grid-column: 1;
  grid-row: 2; }
section article {
  grid-column: 2;
  grid-row: 2; }
section aside {
  grid-column: 3;
  grid-row: 2; }
section footer {
  grid-column: 2;
  grid-row: 3; }
```



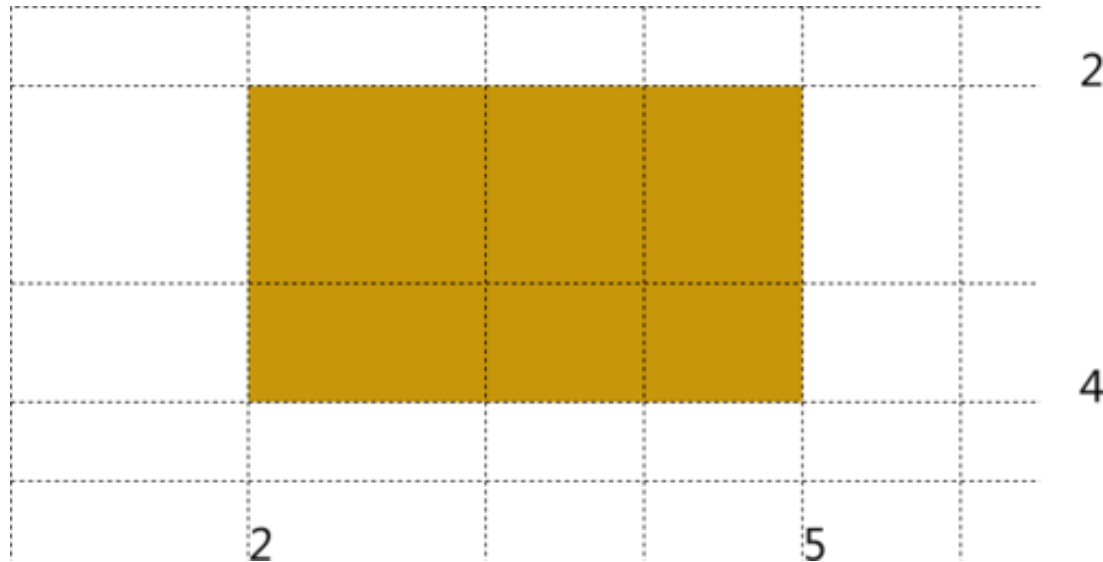
- fr = fraction values
 - new unit applicable to grid-rows and grid-columns properties

Fraction values

- The distribution of fractional space occurs after all or content-based row and column sizes have reached their maximum
- The total size of the rows or columns is then subtracted from the available space and the remainder is divided proportionately among the fractional rows and columns
- Auto: height or width depends on content



Expansion on several cells



```
#item {  
  grid-column: 2;  
  grid-column-span: 3;  
  grid-row: 2;  
  grid-row-span: 2; }
```

Repeated tracks

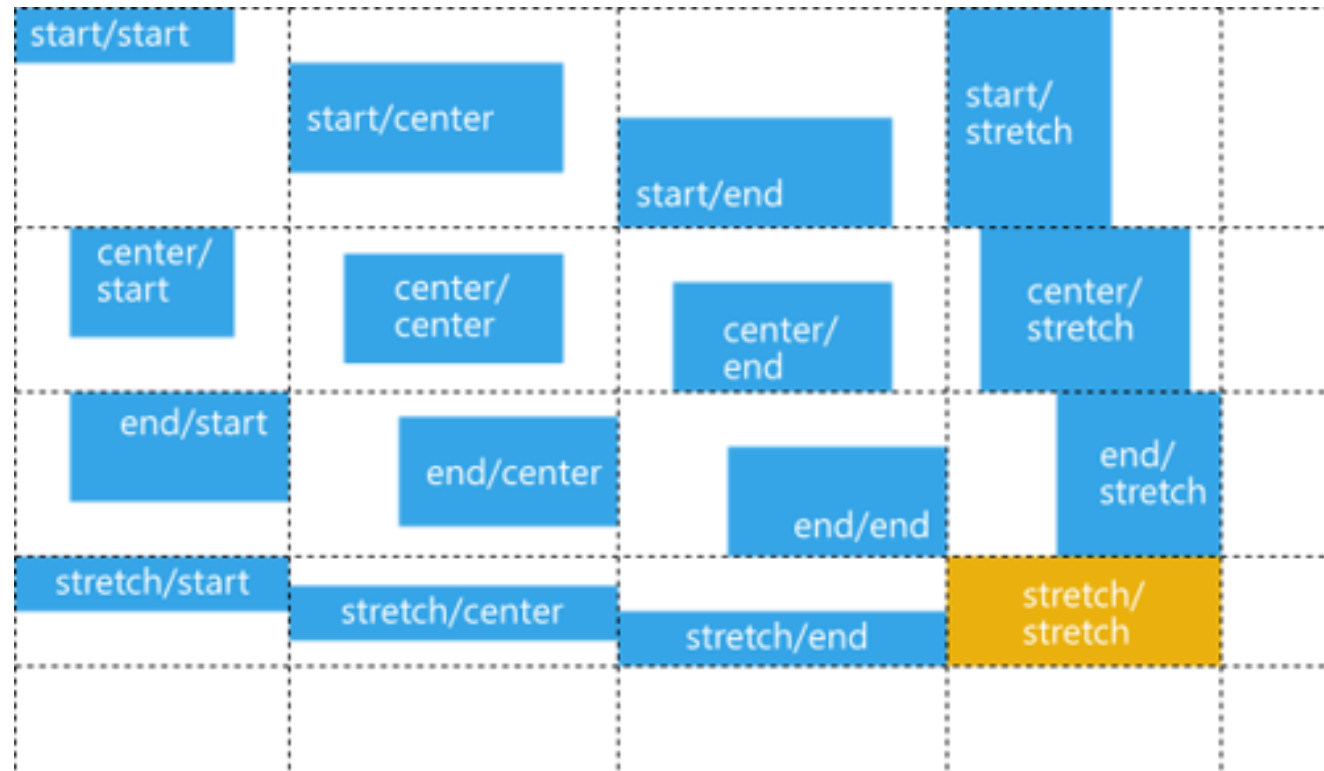
- Compact way to describe a repetitive grid layout



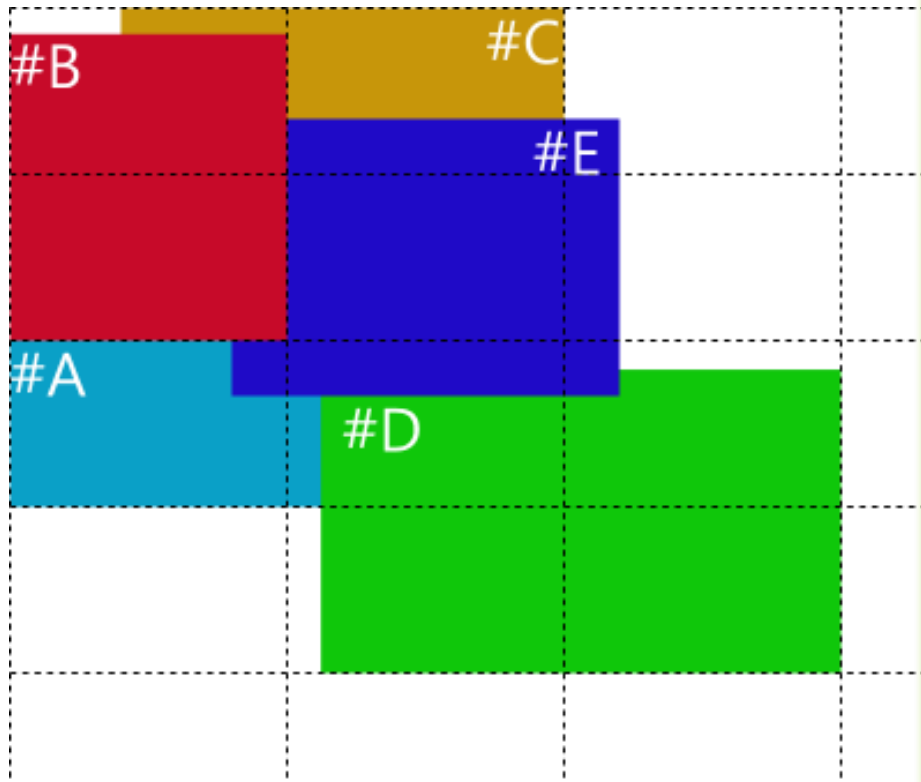
```
#grid {  
  display: grid;  
  grid-columns: 24px (120px 24px) [4];  
  grid-rows: (1fr 24px) [3]; }
```

Binding of the elements

- Controlled by the properties `grid-column-align` and `grid-row-align`
 - Values: `start`, `end`, `center`, `stretch`



Layers control



```
#grid {
  display: grid;
  grid-columns: (1fr)[3];
  grid-rows: (1fr)[4]; }
#A {
  grid-column:1; grid-row:3;
  grid-column-span:2; }
#B {
  grid-column:1; grid-row:1;
  grid-row-span:2;
  z-index:10;
  margin-top:10px; }
#C {
  grid-column:1; grid-row:1;
  grid-column-span:2;
  margin-left:50px; }
#D {
  grid-column:2; grid-row:3;
  grid-row-span:2;
  grid-column-span:2;
  margin:10px 0 0 10px; }
#E {
  grid-column:2; grid-row:2;
  z-index:5;
  margin: -20px; }
```

License

- This work is licensed under the Creative Commons “Attribution-NonCommercial-ShareAlike Unported (CC BY-NC-SA 3,0)” License.
- You are free:
 - to Share - to copy, distribute and transmit the work
 - to Remix - to adapt the work
- Under the following conditions:
 - Attribution - You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
 - Noncommercial - You may not use this work for commercial purposes.
 - Share Alike - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.
- To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/>