


Cascading Style Sheet 3.0

Lesson 04: Layout

December 20, 2015

Proprietary and Confidential

- 1 -


CONSULTING. TECHNOLOGY. OUTSOURCING

Lesson Objectives

- **Layout – Introduction**
- **Positioning**
- **Box Layout**
- **Table Layout**
- **Vendor Prefixes**
- **Working with Columns**



Introduction

- While designing a web page the important thing which we need to consider is the position and alignment of elements on a web page
- Layout properties allow authors to control the visibility, position, and behavior of the generated boxes for document elements
- CSS layout takes care of proper alignment of web page elements by using the following positioning schemes.
 - Relative Positioning
 - Absolute Positioning
 - Fixed Positioning
 - Stacking Contexts
 - Floating and Clearing
 - The Relationship Between display, position, and float

December 20, 2015 Proprietary and Confidential ~ 3 ~



The term "CSS positioning" refers to using CSS to position elements on your HTML page. CSS allows you to position any element precisely where you want it. You can specify whether you want the element positioned *relative* to its natural position in the page or *absolute* based on its parent element.

The position property, together with the float property, controls the way in which the position of the element's generated box is computed.

Absolute positioning

- An element whose position property has the value absolute is said to be absolutely positioned
- The top, right, bottom, left, width, and height properties determine the position and dimensions of an absolutely positioned element.
- An absolutely positioned element will overlap other content unless we make room for it in some way
- Ex:

```
h2
{
  position : absolute;
  left:100px;
  top:150px;
}
```

Both the position and the dimensions can be expressed using all four of the positional properties (top, right, bottom, left). Alternatively, you can specify the position of one corner of the box using top or bottom in combination with left or right, and you can specify the dimensions using width and (optionally) height.

Fixed Positioning

- Fixed positioning is a subcategory of absolute positioning
- The value fixed generates an absolutely positioned box that's positioned relative to the initial containing block
- The position can be specified using one or more of the properties top, right, bottom, and left.
- Ex:

```
h2
{
  position : fixed;
  left:100px;
  top:150px;
}
```

Relative Positioning

- The value `relative` generates a positioned box whose position is first computed as for the normal flow
- An element whose position property has the value `relative` is first laid out just like a static element
- The rendered box is then shifted vertically (according to the `top` or `bottom` property) and/or horizontally (according to the `left` or `right` property).

```
h2
{
  position : relative;
  left:100px;
  top:150px;
}
```

December 20, 2015 | Proprietary and Confidential | - 6 -



As far as the flow is concerned, the element is still in its original position. If the relative shift is significant, it will leave a “hole” in the flow, in which case the rendered box may overlap other content.

The properties `top`, `right`, `bottom`, and `left` can be used to specify by how much the rendered box will be shifted. A positive value means the box will be shifted away from that position, towards the opposite side. For instance, a `left` value of `20px` shifts the box 20 pixels to the right of its original position. Applying a negative value to the opposite side will achieve the same effect: a `right` value of `-20px` will accomplish the same result as a `left` value of `20px`. The initial value for these properties is `auto`, which makes the computed value 0 (zero)—that is, no shift occurs.

Relative positioning is commonly used when we need to shift a box a few pixels or so, although it can also be useful, in combination with negative margins on floated elements, for some more complex designs.

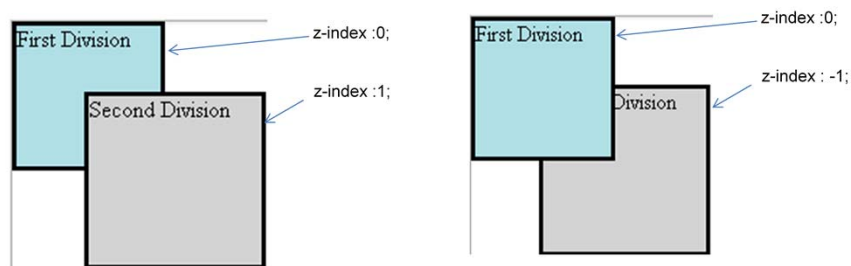
Static Positioning

- The value `static` generates a box that isn't positioned, but occurs in the normal flow. The properties `top`, `right`, `bottom`, `left`, and `z-index` are ignored for static boxes.

```
h1
{
border : solid;
border-color : red;
position : static;
left:100px; //will not work
top:150px; //will not work
}
```

Stacking Contents

- Although we tend to regard a web page as a two-dimensional entity, boxes are positioned in three dimensions. The third dimension is the z axis, which is perpendicular to the screen
- Positioned elements can overlap, since they can be rendered at the same position
- We can specify the stack level via the z-index property as shown below



December 20, 2015 Proprietary and Confidential - 8 -



An integer value—which can be negative—sets the stack level of the box in the current stacking context, and also establishes a new stacking context. The box itself has stack level 0 (zero) in the new context.

The value auto gives the box the same stack level as its parent, and doesn't establish a new stacking context

```
<html>
<body>
  <div id="m1">First Division</div>
  <div id="m2">Second Division</div>
</body>
</html>
```

```
#m1
{
  background-color: powderblue;
  color: RED;
  width: 100px;
  height: 100px;
  z-index: 0;
}
```

```
#m2
{
  background-color: lightgrey;
  color: blue;
  top: 50px;
  left: 50px;
  width: 120px;
  height: 120px;
  position: absolute;
  z-index: 1;
}
```


Floating and Clearing

➤ Float

- With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it.
- Float is very often used for images, but it is also useful when working with layouts
- Elements are floated horizontally, this means that an element can only be floated left or right, not up or down
- If you place several floating elements after each other, they will float next to each other if there is room.

– Ex:

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

➤ Clear

- Using clear property we can avoid flowing of elements around float element
- The clear property specifies which sides of an element other floating elements are not allowed.
- Ex:

```
.text_line
{
  clear:both;
}
```

December 20, 2015 Proprietary and Confidential ~ 9 ~



Float: A floated element is one whose float property has a value other than none. The element can be shifted to the left (using the value left) or to the right (using the value right); non-floated content will flow along the side opposite the specified float direction

Clear: To force an element to start below any floated elements, we can use the clear property with a value of left, right, or both. An element whose clear property is set to left will start below all left-floated boxes in the same block formatting context, while a clear value of right will clear all right-floated boxes. If clear is set to both, the element will start below any floated box in that context

Demo : Positioning

- Pos_Absolute.html
- Pos_Relative.html
- Pos_Fixed.html
- Pos_Static.html
- Positioning_all.html

Box Layout

- In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content.
- The box model allows us to place a border around elements and space elements in relation to other elements
- Image below illustrates the box model



December 20, 2015 Proprietary and Confidential - 11 -



Your understanding of the box model concept, and how it relates to the way in which an element's final dimensions are determined, will be essential to your understanding of how an element is positioned on a web page. The box model applies to block-level elements.

Explanation of the different parts:

Margin - Clears an area around the border. The margin does not have a background color, it is completely transparent

Border - A border that goes around the padding and content. The border is affected by the background color of the box

Padding - Clears an area around the content. The padding is affected by the background color of the box

Content - The content of the box, where text and images appear

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

Implementing the Box Model

- The box model is best demonstrated with a short example
- Total space required to accommodate an element is calculated as follows

Total width = left margin + left border + left padding + width + right padding + right border + right margin

Total height = top margin + top border + top padding + height + bottom padding + bottom border + bottom margin

- Ex:

```
.box {  
    width: 300px;  
    height: 200px;  
    padding: 10px;  
    border: 1px solid #000;  
    margin: 15px;  
}
```

December 20, 2015 | Proprietary and Confidential | - 12 -



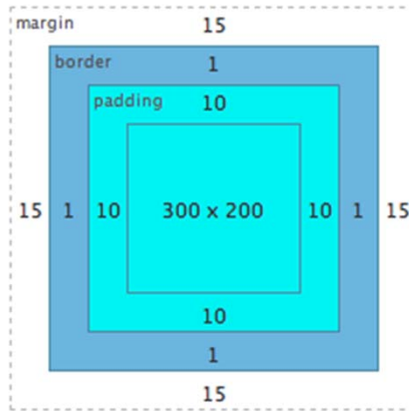
The total size of the element above will be calculated as follows:

Total width = 15 + 1 + 10 + 300 + 10 + 1 + 15 = 352px

Total height = 15 + 1 + 10 + 200 + 10 + 1 + 15 = 252px

Ex Cont...

- With the calculation , Box for the css code in previous slide looks as shown below



December 20, 2015 Proprietary and Confidential - 13 -



In the above picture , we can clearly see the content area in the center, the padding around the content area, the border area, and the margin area. The outer edge of the content area is called the content edge or inner edge; the outer edge of the padding area is called the padding edge; the outer edge of the border area is called the border edge; and the outer edge of the margin area is called—you guessed it—the margin edge or outer edge.

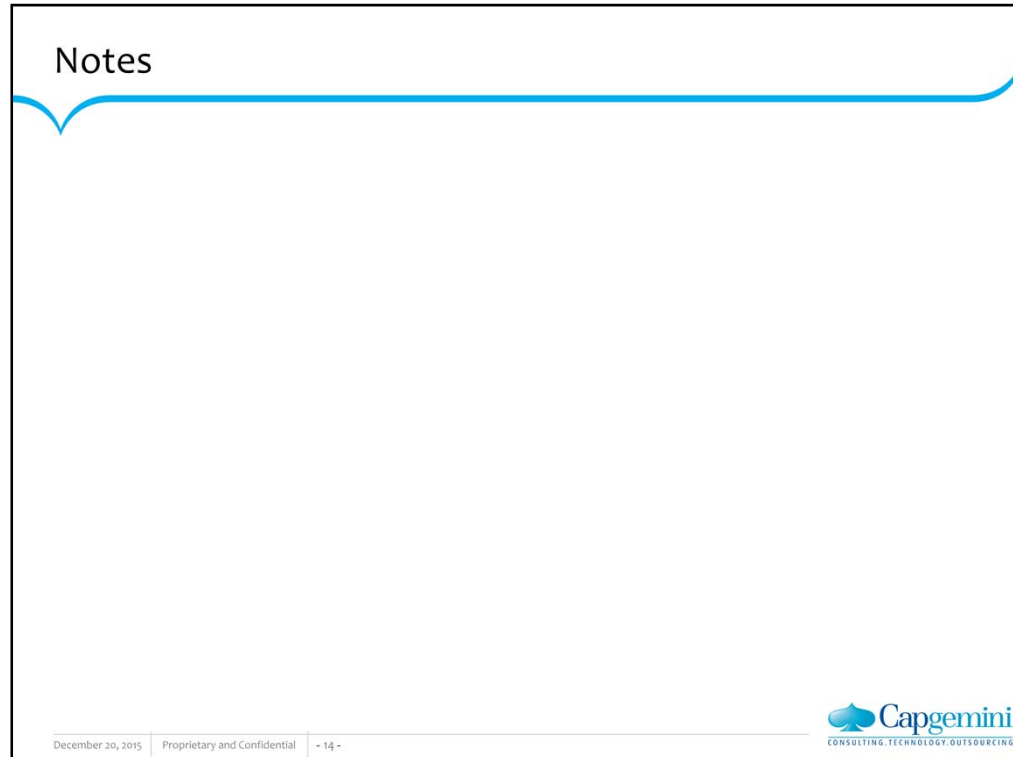
You can see from this short example that, for this element to fit on the page, we'll need a space that's at least 352px wide and 252px high. If the space available is any smaller than this, the element will be misplaced, or will overflow its containing block. Note that Internet Explorer 6 and earlier versions will most likely stretch the containing block to accommodate this extra height, and could severely disrupt the layout. Other browsers will let the element overflow its boundaries, but will ignore the content.

Notes:

An important point to note is that an element that has its width set to 100% (that is, 100% of the content width of its parent element) shouldn't have any margins, padding, or borders applied

It can severely disrupt a page's layout, as content will either overflow or push elements wider than they should be.

The solution, in most cases, is to avoid adding a value for the property width (other than auto), and to apply the margins, padding, and borders only. The width property of a static element will default to auto, and even with padding, borders, and margins added, it will still assume the full available content width..



This approach may not be feasible in some instances, such as cases where the element is not a static element, and requires the definition of a specific width value (as in the case of a floated element that doesn't automatically expand to fill its parent). In these cases, you have two options.

If the available space is of a fixed width, you can simply add the value of each component together to ensure that it matches the available width. For example, if the available space is 500px wide, and you require an element to have 20px padding, simply set the width to 460px and the padding to 20px for that element ($20 + 460 + 20 = 500$). This solution assumes that the length values specified for the element's box properties use the same unit of measurement, since you won't be able to add together a mixture of units ($200\text{px} + 10\%$, for example, makes no sense in this context).

When the available content space has an unknown width—as in the case of a fluid layout—this method can't be used, as percentages and pixels can't be added together. In this case, the solution is to declare a width of 100% for the element concerned, and to apply the padding, border, and margin values to a nested element instead. That nested element has no width declaration, and can display the required padding, borders, and margins without encroaching on the parent element.

Box-ordinal-group Property

➤ Using this property we can specify the display order of the child element of a box

➤ Ex:

```
<body>
<div class="box">
<div class="ord2">First in source</div>
<div class="ord1">Second in source</div>
<div class="ord1">Third in source</div>
</div>
</body>
```

```
.box
{
display : box;
border:1px solid black;
}
.ord1
{
margin:5px;
box-ordinal-group:1;
}
.ord2
{
margin:5px;
box-ordinal-group:2;
}
```

December 20, 2015 Proprietary and Confidential - 15 -



The box-ordinal-group property specifies the display order of the child elements of a box.

Elements with a lower value are displayed before those with a higher value.

Note:

The display order of the elements with the same group value depend on their source order.

Use vendor prefixes for box-ordinal-group property.

It will not work on internet explorer


Demo : Box Ordinal Property

BoxOrdinal.html

December 20, 2015

Proprietary and Confidential

- 16 -




Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

Table Layout

- The **table-layout** property sets the table layout algorithm to be used for a table
- Ex:

```
table
{
  table-layout:fixed;
}
```
- **Properties:**
 - **Auto** : Automatic table layout algorithm
 - **Fixed** : Fixed table layout algorithm
 - **Inherit** : Specifies that the value of the table-layout property should be inherited from the parent element

December 20, 2015 | Proprietary and Confidential | - 17 -



Auto: Automatic table layout algorithm (this is default):

The column width is set by the widest unbreakable content in the cells

Can be slow, since it needs to read through all the content in the table, before determining the final layout

Fixed: Fixed table layout algorithm:

The horizontal layout only depends on the table's width and the width of the columns, not the contents of the cells

Allows a browser to lay out the table faster than the automatic table layout

The browser can begin to display the table once the first row has been received

Inherit: Specifies that the value of the table-layout property should be inherited from the parent element

Demo: Table Layout


➤

Table_layout.html

December 20, 2015

Proprietary and Confidential

- 18 -


CONSULTING. TECHNOLOGY. OUTSOURCING

Flexi Box Layout

➤ The box-flex property specifies whether the child elements of a box is flexible or inflexible in size.

➤ Ex: Define two flexible p elements. If the parent box has a total width of 300px, #p1 will have a width of 100px, and #p2 will have a width of 200px:

```
div
{
  display:-webkit-box;
  width:600px;
  border:1px solid black;
}
#p1
{
  -webkit-box-flex:1.0;
  border:1px solid red;
}
#p2
{
  -webkit-box-flex:3.0;
  border:1px solid blue;
}
```

December 20, 2015 Proprietary and Confidential - 19 -



Note:

Elements that are flexible can shrink or grow as the box shrinks and grows. Whenever there is extra space in a box, flexible elements are expanded to fill that space.

The box-flex property is only supported by Opera.

Firefox supports an alternative, the -moz-box-flex property.

Safari and Chrome support an alternative, the -webkit-box-flex property.


Demo : Flexi Box Layout

> fexi_box_layout.html

December 20, 2015

Proprietary and Confidential

- 20 -



Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

Vendor Prefixes

- Vendors—browser makers—are free to implement extensions to the CSS specifications that, in most cases, are proprietary to their browser.
- In order to accommodate the release of vendor-specific extensions, the CSS specifications define a specific format that vendors should follow
- The format is quite simple: keywords and property names beginning with - (dash) or _ (underscore) are reserved for vendor-specific extensions

'-' + vendor specific identifier + '-' + meaningful name
'_' + vendor specific identifier + '_' + meaningful name

December 20, 2015 | Proprietary and Confidential | - 21 -



They may do this for a number of reasons, such as adding new features for users, or for experiments and debugging. Most often, though, the extensions are used to release and test browser features that have been developed in the preparation of W3C drafts that have not yet reached Candidate Recommendation status—the extensions allow these new properties to be widely tested before they become available as standard CSS properties.


Current format allows any vendor-specific extension to coexist with any future (or current) CSS properties without causing conflicts because, according to the W3C specifications, a CSS property name will never begin with a dash or an underscore

Vendor Prefixes

➤ A number of extensions exist. Their prefixes are outlined below

Prefix	Organisation
-ms-	Microsoft
-mso-	Microsoft Office
-moz-	Mozilla Foundation (Gecko-based browsers)
-o-	Opera Software
-atsc-	Advanced Television Standards Committee
-wap-	The WAP Forum
-webkit-	Safari (and other WebKit-based browsers)
-khtml-	Konqueror browser

December 20, 2015 | Proprietary and Confidential | - 22 -

 Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

Even though vendor-specific extensions are guaranteed not to cause conflicts, it should be recognized that these extensions may also be subject to change at the vendor's whim, as they don't form part of the CSS specifications


Although these extensions can be useful at times, it's still recommended that you avoid using them unless it's absolutely necessary

CSS3 Multiple Columns

- In CSS 3 We can create multiple column display for laying text – like in newspapers
- Following are the Multiple column properties
 - column-count
 - column-gap
 - column-rule
- Ex:

```
div
{
  -webkit-column-count:3;
  -webkit-column-gap:40px;
  -webkit-column-rule:3px outset #ff00ff;
}
```

December 20, 2015 | Proprietary and Confidential | - 23 -



Column Count:

The column-count property specifies the number of columns an element should be divided into:

Column Gap:

The column-gap property specifies the gap between the columns

Column Rule:

The column-rule property sets the width, style, and color of the rule between columns.

Note:

Internet Explorer does not yet support the multiple columns properties.

Firefox requires the prefix -moz-.

Chrome and Safari require the prefix -webkit-.


Demo: Multiple Column Layout

➤ Multi_Column.html

December 20, 2015

Proprietary and Confidential

- 24 -



Capgemini
CONSULTING. TECHNOLOGY. OUTSOURCING

Lesson Summary

➤ **We have so far seen:**

- Layout
- Positioning
- Box Layout
- Table Layout
- Vendor Prefixes
- Working with Columns



Review Questions

➤ **Question 1: Which of the following positioning mechanism makes element to be fixed when the web page is resized.**

- Fixed
- Absolute
- Static
- Relative



➤ **Question 2: Which of the following is not a property of Multiple Column layout**

- column-count
- column-space
- column-rule
- None