Cascading Style Sheet 3.0

Lesson 05: Layout

Capgemini

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

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;
}
```

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).

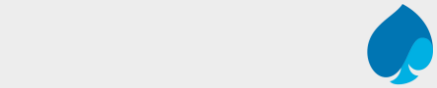
```
h2
{
position : relative;
left:100px;
top:150px;
}
```

Static Positioning

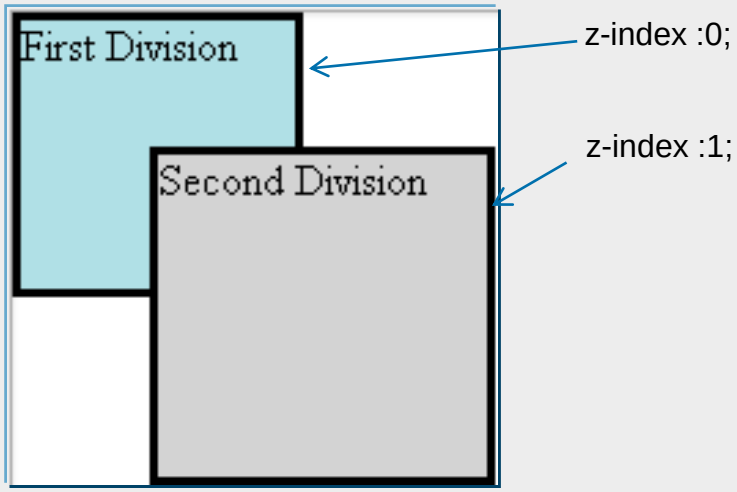
➢The value static generates a box that isn't positioned, but occurs in the normal flow. The properties top, right, bottom, left, and z-index are ignored for static boxes.

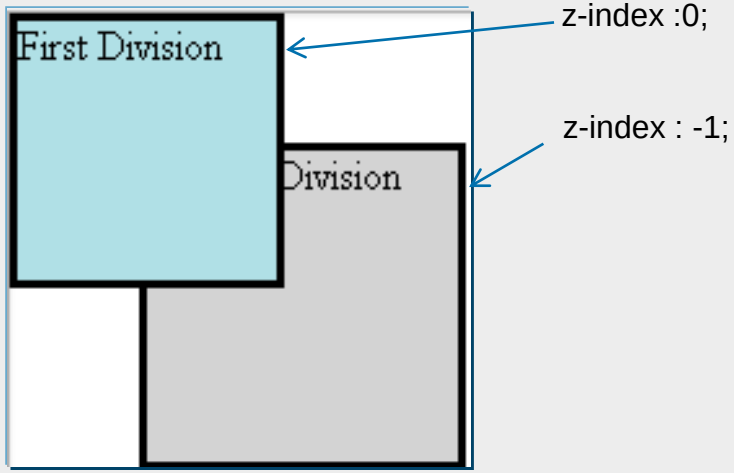
```
h1
{
border : solid;
border-color : red;
position : static;
left:100px; //will not work
top:150px; //will not work
}
```

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





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;
}
```

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

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

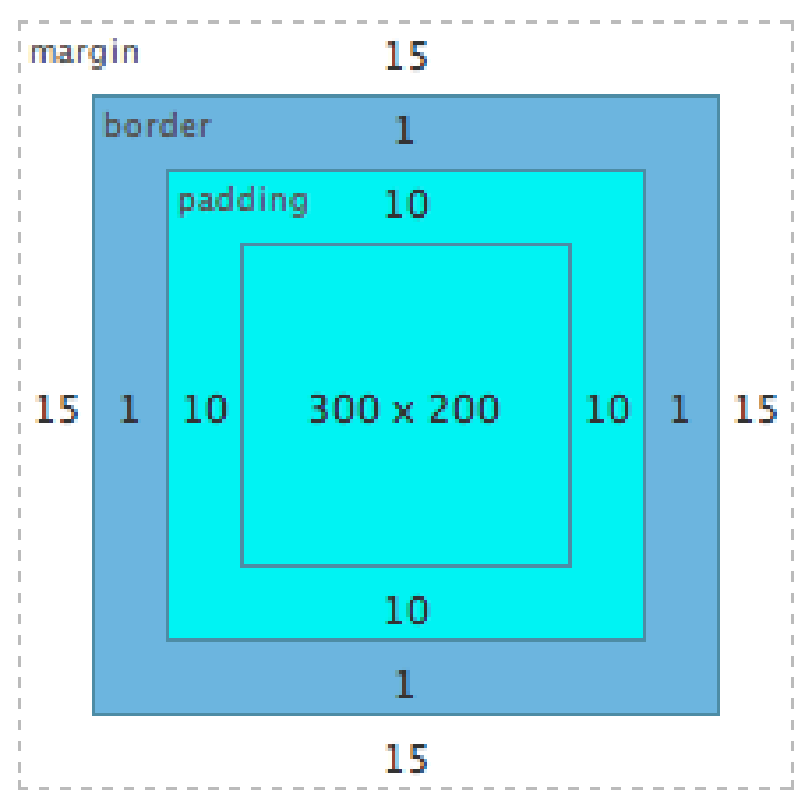
> Total width = left margin + left border + left padding + width + right padding + right border + right margin
>
> Total height = top margin + top border + top padding + height + bottom padding + bottom border + bottom margin

- Ex:

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

Ex Cont…

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



Notes

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;
}
```

Demo : Box Ordinal Property

- BoxOrdinal.html

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

Demo: Table Layout

- Table_layout.html

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;
}
```

Demo : Flexi Box Layout

- fexi_box_layout.html

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

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

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;
}
```

Demo: Multiple Column Layout

- Multi_Column.html

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