Layout and Positioning
Part I of II

XHTML Elements & New Styling Tricks

- Using XHTML elements to create “real” layouts
- Bringing everything that we have learned so far about CSS
- Multi-column layouts

First lets review <div> concepts from last chapter

- <div> enables you to divide your webpage into groups
- Provides structure by dividing page into logical sections
- Logical sections are enclosed within a <div> element
- Logical sections are a group of related elements on the page
- Logical sections should represent the main content areas

Adding an id

- Adding an id to <div> enables you to label what the grouping means or represents
- Using the id attribute provides a unique label for the <div>
- Styling can be specified for the <div id="dogs">
- Give the elixirs <div> and id of "elixirs"

Adding style to <div>

<div id="dogs"> </div>

Creating rules for the <div> within our CSS

#dogs {
  background-image: url("dogs.gif");
}
**Weekly Elixir Specials**

**Lemon Breeze**

The ultimate healthy drink, combines herbal botanicals...

**Chai Chiller**

Not your traditional chai...

**elixirs rules**

```css
#elixirs {
  border: thin solid #007e7e;
  width: 200px;
  padding: 0px 20px 20px 20px;
  margin-left: 20px;
  text-align: center;
  background-image: url("images/cocktail.gif");
  background-repeat: repeat-x;
  line-height: 1;
}
```

Adding `float: right` to elixir CSS

```css
#elixirs {
  border: thin solid #007e7e;
  width: 200px;
  padding: 0px 20px 20px 20px;
  margin-left: 20px;
  text-align: center;
  background-image: url("images/cocktail.gif");
  background-repeat: repeat-x;
  float: right;
  line-height: 1;
}
```

**float: right** viewed through browser

- Adds a two column style format
- More readable and easier to read
- Newspaper type convention
- Very desirable appearance for web reading
Flow and XHTML

- Browser uses flow to layout XHTML elements
- Flow means, browser starts at the top of any XHTML file and follows the flow of elements from top to bottom
- Each new block element causes a line break
- Elements take up the full width of the page

Inline elements

- Inline elements appear within block elements
- Flow next to each other, horizontally, from top left to bottom right
- Inline elements are laid next to one another horizontally, as long as there is room and then they may wrap to fit the browser window’s width
- Shrinking the browser window or shrinking the content area will require more vertical room to display with content being continued onto new lines

Inline element margins

- Each inline element can have a margin
- Setting margins on inline elements is not common though
- Inline elements are placed next to each other so margin between is determined by their combined margin

Block elements margins

- Browser places block elements on top each other
- Margins between block elements become shared margins
- One block element may have a bottom margin and the other may have a top margin
- Shared margin is the size of the larger of the two margins

How to float an element

Give desired element an id

```html
<p id="sidebar">
New Artists
</p>
```

Add a width rule within CSS

```css
#sidebar {
    width: 200px;
    float: right;
}
```

- Previewing in a browser will position floated element all the way to the right
- Floated paragraph has been removed from the normal “flow” of the other block elements
How to best float an element

- Element should appear within XHTML where you want float to begin
- Move element with float property to desired beginning location
- Everything below the floated element is going to move up and wrap around it

Place floated element where you want to appear

```html
<body>
  <div id="header">
    <img src="images/header.gif" alt="header desc" />
  </div>
  <p id="sidebar">
    New Artists
    <p id="name">
    New Artists
  </p>
  <div id="main">
    <h1>Music store info</h1>
    <p>At Showbiz music we have the most popular selection of new music…</p>
  </div>
  <div id="footer">
    <img src="images/footer.gif" alt="footer desc" />
  </div>
</body>
```

Reviewing the steps to float an element

1. Give desired element an id `<p id="name">`
2. Position element where you want float to begin
3. Within CSS add a rule to set the width on the element `width: 200px;`
4. Within CSS add a rule to set the float either to the left or right `float: right;`
5. Test page

Creating a two column style format

- Display of page will vary depending on size of the browser window
- Solving problems with CSS layout is more of an art than science
- Several techniques can be used to achieve a desired look

Two column styles typically have a header and footer

- Header and footer aren’t two column though
- Resizing of browser can adversely affect the desired layout

Another design, add a margin to the main section to achieve a true two column look

- We can add a margin to the main section
- Taking into account the width of the sidebar and adding a right margin of 330px
- Adding a margin creates a “gutter” between the main and side columns
- Gutter used in the publishing world to refer to the “alley or space” between columns of text in a page layout
Changing main’s right margin to 330px will add a gutter in between main and sidebar

```css
#main {
  background: #efe5d0 url("images/cocktail.gif") top left;
  font-size: 105%;
  padding: 15px;
  margin: 0px 330px 10px 10px;
}
```

Remember the box model

```
<table>
<thead>
<tr>
<th>Content area goes here</th>
</tr>
</thead>
<tbody>
<tr>
<td>Padding</td>
</tr>
<tr>
<td>Border</td>
</tr>
<tr>
<td>Margin</td>
</tr>
</tbody>
</table>
```

Trying to create the ideal 2 column look

- By expanding the margin of the main `<div>` we’re creating the illusion of a two column layout
- Gutter appears between
- Still problematic since size of browser affects the appearance

Fine tuning our two column page

- Sidebar essentially floats on top of the page and floats on the right side of the regular flow
- If the browser is resized to be wider, the footer and the sidebar start to overlap at bottom
- Overlap meaning the main section is no longer long enough so footer has moved up and overlaps with the sidebars content

Fine tuning our two column page

- Page contains 4 sections:
  
  - header, main, sidebar, and footer
- Since sidebar has a right float property it is not part of the regular flow
- Regular flow is header, main, and footer
- Regular flow wraps around the floated element

Trying to solve the two column problems

- Setting a width on the main area will not achieve desired results
- When the browser is too wide the two columns separate with an undesired margin in-between them
- When the browser is too narrow the two columns will actually overlap
- Using a clear property will help to fix our problem
**clear: property**
Set this property on an element to not allow floating content on either its left, right, or both sides

```css
#footer {
    background-color: #675c47;
    color: #efe5d0;
    text-align: center;
    padding: 15px;
    margin: 10px;
    font-size: 90%;
    clear: right;
}
```

**clear: property**
- Browser interprets by looking to see if there is a floating element to the right side of the footer
- If there is a floating element then the footer will be moved down until there is no longer a floating element to its right

**CSS advantages and limitations**
- Remember CSS is an art not science
- No layout solution will be perfect
- Many ways to do layouts using CSS
- Make sure to always preview on multiple browsers and platforms

**FAQ’s about using float**
- Inline elements can be floated, for example you can float an image left or right
- Margins do not collapse on floated elements
- The margins of floated elements aren’t actually touching the margins of the elements in the normal flow, so they can’t be collapsed

**CSS advantages and limitations**
- Remember, XHTML is meant to be used to author our content, marking up the content in order to display within a browser
- Content is styled using CSS
- Content needs to be viewable on any device, whether or not CSS is used
- CSS does a good job of gracefully degrading for various viewing environments and devices even if CSS is not supported
FAQ's about using float

• Floated elements are ignored by block elements
• Block elements are flowed onto the page as normal
• Inline content nested inside a block element always flows around a floated element, observing the boundaries of the floated element
• One disadvantage to CSS is the unintended layout when CSS is not supported in viewer, float will not appear and content appears in sequence without float

Liquid layouts

• Expand to fill whatever width we resize the browser to
• By expanding, they fill the space available and allow users to make good use of their screen space

Frozen layouts

• Lock the elements down to a fixed size so they can't move at all
• Problems related to the window expanding are thus avoided

How to experiment with Frozen layouts

First, add the following within your XHTML
• Adding a new <div> element with an id of "allcontent"
• <div> will go around all the content in your page
• Add the opening <div> after the opening <body>
• Add the closing <div> before the closing </body>

```html
<body>
  <div id="allcontent">
    <div id="header">
      <img src="i/header.gif" alt="logo" />
    </div>
  </div>
</body>
```

How to experiment with Frozen layouts

Next, add the following within your CSS
• Adding a new rule for the <div> with an id of "allcontent"
• Setting a width constrains everything to fit within 800 pixels
• Adding padding and background color

```css
#allcontent {
  width: 800px;
  padding-top: 5px;
  padding-bottom: 5px;
  background-color: #675c47
}
```

Problems with Frozen layouts

• Page is now frozen to 800 pixels in width
• Resizing the browser to be wider than this causes empty space to appear
Hybrid design using frozen and liquid techniques

- We can create a hybrid between the liquid and the frozen layouts in an attempt to solve some of our display problems.
- Add a margin-left and a margin-right setting to our allcontent rule.

```
#allcontent
{
  margin-left: auto;
  margin-right: auto;
  width: 800px;
  padding-top: 5px;
  padding-bottom: 5px;
  background-color: #675c47
}
```

Hybrid design using frozen and liquid techniques

- Setting margins to auto allows browser to expand content area as much as necessary.
- Left and right margins will be the same, expanded as much as is necessary.

margin: auto; property

Enables browser to be resized either larger or smaller than 800 pixels and still have a desired appearance.

```
#allcontent
{
  width: 800px;
  padding-top: 5px;
  padding-bottom: 5px;
  background-color: #675c47
  margin-left: auto;
  margin-right: auto;
}
```

margin: auto; property

- Enables content to be in the correct order, useful for browsers that will not display using CSS.
- Versus using the earlier method of a sidebar where we floated right.

- Using sidebar and float did not allow browser to be resized very wide without messing up our intended layout.
- Using sidebar and float also made viewing page on browsers that do not support CSS difficult since content was not displayed in the correct order.
- Sidebar came before main content, without CSS content would not display in order.

Layout and Positioning continued  
Part II
Absolute positioning

- Another alternative to float is called absolute positioning
- Absolute positioning useful for multi-column as well as other types of styling

position: absolute; property

Adding the position property to specify that the element will be positioned absolutely
- Top and right properties set position
- Bottom and right properties are also available
- Width sets the width of the <div>

```html
# sidebar {
  position: absolute;
  top: 100px;
  right: 200px;
  width: 280px;
}
```

Sidebar is absolutely positioned
- Removed from the flow and positioned according to properties specified
- Sidebar is considered out of the flow, other elements don’t know its there
- Elements in the flow do not wrap their inline content around an absolutely positioned element
- Elements are not affected by sidebar being on the page
- Each positioned element has a property called z-index that specifies its placement

position: absolute; property

Adding another <div> within our XHTML
- Adding another rule within CSS will place this new <div> and its content on top of the sidebar from the last example
- Sidebar and ad are layered on the page
- Ad has the greater z-index so it is on top

```html
# ad {
  position: absolute;
  top: 150px;
  right: 100px;
  width: 400px;
  z-index: 99;
}
```

The position property is set to “static” by default
- Static positions the element in the normal document flow, positioned by the browser
- Using the float property takes an element out of the flow
- Float property can position element left or right

position: absolute; property

- Absolute positioning tells browser exactly where to position elements
- You can position any element, not just <div>
- For example you can position an inline element like <img> although other elements are not common
- When using absolute positioning element is removed from the normal flow of the page
position: property

There are four position properties:

Static, absolute, fixed and relative

Static - element is placed in the normal document flow, this is the default value for positioning.

Absolute - tells browser exactly where to position elements.

Fixed - places an element in a location that is relative to the browser window, fixed elements never move.

Relative - takes an element and flows it on the page in its normal way but then offsets it before displaying it on page.

position: property

Width property when used with absolute position is optional however if not used width the block element will take up the entire width of browser.

```
# a d {
  position: absolute;
  top: 150px;
  right: 100px;
  width: 400px;
}
```

position: property

• Using pixels to define positioning is optimal however you can use percentages.
• Percentages are based on how wide your browser is displayed.
• Good if you want an element to only be 10% of the window no matter how wide it is displaying.

```
# a d {
  position: absolute;
  top: 10%;
  right: 10%;
  width: 100%;
}
```

position: absolute; property

Applying absolute positioning to our sidebar example.

```
# sidebar {
  background: #efe5d0 url("i/bg.gif") bottom right;
  font-size: 105%;
  padding: 15px;
  margin: 0px 10px 10px 10px;
  position: absolute;
  top: 128px;
  right: 0px;
  width: 280px
}
```

position: absolute; property

Applying absolute positioning to our main example.

```
#main {
  background: #efe5d0 url("i/cocktail.gif") top left;
  font-size: 105%;
  padding: 15px;
  margin: 0px 330px 10px 10px;
}
```

position: absolute; property

• Preview in browser.
• Sidebar is positioned 128 pixels from the top and flush against the right side of page.
• Sidebar aligns with the top of the main <div>.
• Main <div> is flowed just below the header.
• Right margin is the same.
position: absolute; property

- Page looks good when previewed in a browser
- Looks just like the float example except the sidebar is being positioned absolutely
- As you resize browser the page continues to look good until you make it so wide that the main <div> becomes smaller than the sidebar <div>

Two column layout techniques reviewed

- The floating layout
- The hybrid (in between floating and frozen)
- The absolute layout

Two column layout techniques disadvantages

The floating layout

We used a float property with a clear on the footer, float method is often considered the most flexible solution

Two column floating disadvantages

Puts content in an order that may not be optimal for non CSS displays

```css
#sidebar {
    width: 200px;
    float: right;
}
#footer {
    background-color: #675c47;
    clear: right;
}
```

Two column layout techniques disadvantages

The hybrid (in between floating and frozen)

Created a frozen layout by wrapping a fixed <div> around all the content and then added the “auto” property value which allowed the margins to expand

Two column hybrid disadvantages

Content doesn’t expand to fill the entire browser window, stays 800 wide in example (not expanding may not matter to some viewers?)

```css
#allcontent {
    width: 800px;
    background-color: #675c47;
    margin-left: auto;
    margin-right: auto;
}
```
Two column layout techniques disadvantages

The absolute layout
Used absolute positioning, to create the liquid layout along with the content ordered in a more logical way for non CSS viewers.

Two column absolute layout disadvantages
Since the clear property cannot be used with absolute elements, the footer can appear under the sidebar when the browser is wide.

Best technique is up to your design needs

• Using the last example, the absolute layout, we can add a margin size to the footer which matches the main content area.

Adding a margin size to the footer to match the main content area

• See next slide for footer modifications to make

Best technique is up to your design needs

• Footer now remains under the main content area only and doesn’t appear under sidebar.

absolute positioning

Use ancestor’s positioning when positioning within an already absolute positioned <div>.

<div> with an id equal to tv is nested within <div> with an id equal to sidebar

<div id="sidebar">
    <div id="tv">
        <img src="image.gif" alt="image desc" />
    </div>
    <p class="guide">
        Guide to current events
    </p>
</div>

</div>
absolute positioning

- Closest positioned ancestor is the sidebar \(<div>\)
- The positioning will be relative to the edges of the sidebar, not the page
- Closest ancestor is the nearest containing block that is positioned
- Bottom property refers to bottom of browser window and not webpage

absolute positioning example

```css
#tv{
  position: absolute;
  top: 100px;
  right: 200px;
  width: 280px;
}
```

fixed positioning

- You specify the position of an element just like you do with absolute positioning
- Except position is an offset from the edge of the browser’s window rather than the page
- Once you placed content with fixed positioning it stays right where you put it, it doesn’t even move when page is scrolled

fixed positioning example

```css
#coupon{
  position: fixed;
  top: 300px;
  left: 100px;
}
```
fixed positioning

Disadvantages older browser like IE6 and older don’t support fixed positioning, won’t display properly

```css
#coupon{
    position: fixed;
    top: 300px;
    left: 100px;
}
```

fixed positioning tricks

- Adding rules to modify the coupons appearance
- Specify a negative property value to create a negative offset

```css
#coupon{
    position: fixed;
    top: 300px;
    left: -90px;
}
#coupon img{
    border: none;
}
#coupon a: link{
    border: none;
}
#coupon a: visited{
    border: none;
}
```

fixed positioning tricks

- Using fixed positioning covers up content below coupon

relative positioning tricks

- Using relative positioning allows elements to still be part of the flow, however the browser does offsets its position
- Versus absolute positioning which takes the document completely out of the flow
- Relative positioning enable you to keep the element in the flow, yet have a reserved space and still be able to offset
relative positioning tricks
Add the following rule:
```
.beans img {
  position: relative;
  left: 120px;
}
```

relative positioning tricks
• We’re using a descendant selector to select only images inside the beans
• Inline elements like images can use any of the positioning techniques including float
• Position of relative
• Offset is from the position where it is placed in the flow
• Image will be displayed 120 pixels from the left of where it sits in the flow of the document
• For offsets use right, left, top, and/or bottom

relative positioning tricks
• Preview in browser
• Image actually extends beyond the sidebar and into the margin and off the edge of page
• Browser first flows a relative element onto the page, and only then does it offset where it is displayed
• Just to review, relative positioning enable you to keep the element in the flow, yet have a reserved space and still be able to offset

Are 3 column designs possible?
Easy to create a 3 column design, one way is…
• Create your header (optional)
• Create a `<div>` that is floated left
• Create a `<div>` that is floated right
• Create a `<div>` for the content (center column) this div should follow the two floated `<div>`’s
• Create a footer `<div>` with a clear property set to both

Chapter Review
Remember z-index for layers/layering
```
# ad {
  position: absolute;
  top: 150px;
  right: 100px;
  width: 400px;
  z-index: 99;
}
```