cfunited

A ColdFusion, Flex & AIR Conference

Hands-on CSS

Lansdowne Resort, Leesburg VA August 12- 15, 2009

www.cfunited.com

DEE SADLER

ADOBE USER GROUP MANAGER

ADOBE COMMUNITY EXPERT

ADOBE CERTIFIED EXPERT AND INSTRUCTOR

ADOBE FREELANCE PROFESSIONAL

GIRL GEEK

Our Project

- **Layout - We'll build the initial layout**
- **Add some navigation**
- **Take a look at dynamic CSS**

But first.....
Some CSS Basics

CSS Basics

- Selectors
- Cascading
- Box Model
- Principals
- Best Practices (we'll talk about this more as we go along)

Selectors

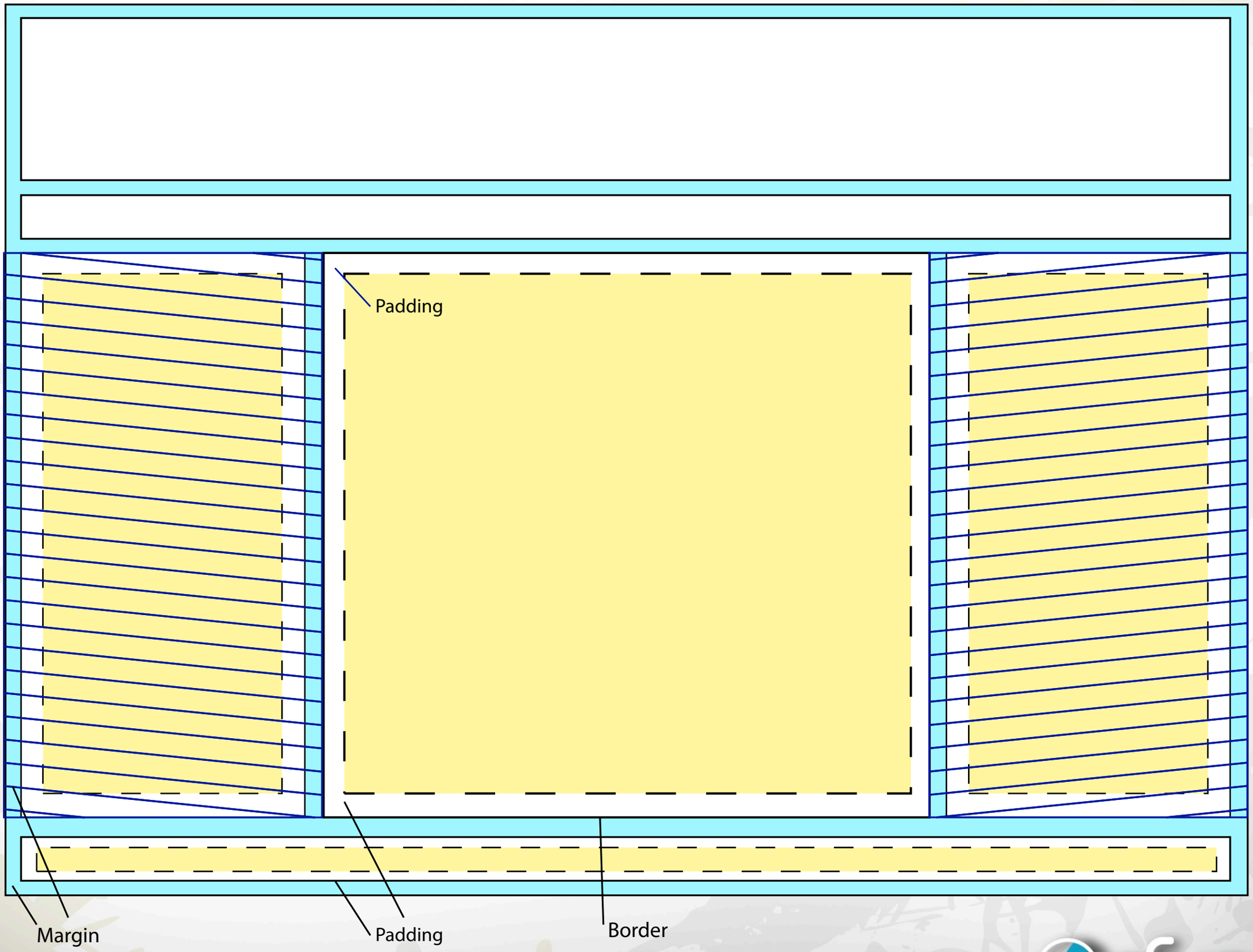
- ID
  - #only1PerPage
- Class
  - .repeatableItem
- Tag
  - body, p, a, ul, h1-h6
- Advanced/compound
  - ul li a, #footer a

Cascading

- user/browser/our style sheets
- inline/embedded/style sheet/order on style sheet
- Using tools like Firebug can help understand the true cascade
- inline/ID's/(pseudo-)classes/attributes/elements

Use the cascade, be the cascade

Box Model - it all adds up!



Principles

- Layout
- Color
- Imagery
- Typography
- Form

Measurements - size does matter

Unit	Description
%	percentage
in	inch
cm	centimeter
mm	millimeter
em	1em is equal to the current font size. 2em means 2 times the size of the current font. E.g., if an element is displayed with a font of 12 pt, then '2em' is 24 pt. The 'em' is a very useful unit in CSS, since it can adapt automatically to the font that the reader uses
ex	one ex is the x-height of a font (x-height is usually about half the font-size - or 1/2 em)
pt	point (1 pt is the same as 1/72 inch)
pc	pica (1 pc is the same as 12 points)
px	pixels (a dot on the computer screen)

We use ems mostly, with a little % and a little px thrown in because they scale.

body tag thoughts

- body {font-size:62.5%;}
  - In this case the font-size is reset to 1em and 10px = 1em. So when you want a 18px font, you will only have to say font-size:1.8em. And in this way you are not forced to calculate ems
- From there on out... we use ems for all font measurements and anything else we want to expand with the browser window

Layout

- Liquid
  - % of your screen size (and for font size of body default for all text)
- Elastic
  - based on ems
- Hybrid
  - em for sidebar (example) % for content/ wrappers
- fixed

Layout Best Practices

- Avoid inline styling whenever *possible*
- Use consistent semantic markup when possible
  - h1 should look the same throughout site, but some things can't be helped like; #footer a or #sidebar1 ul li a
- Module vs content
- Avoid non-standard browser fonts
  - use: Arial, Verdana, Helvetica, Tahoma, Trebuchet MS and maybe Geneva (for san serif) Georgia, Times/Times New Roman, perhaps Palatino for Serif
- Choose bling with caution = rounded corners, drop shadows etc.

Floats - no not the ice cream kind

- Anything that has a float, is above items that don't in the page structure

How Elements Float

Elements are floated horizontally, this means that an element can only be floated left or right, not up or down.

A floated element will move as far to the left or right as it can. Usually this means all the way to the left or right of the containing element.

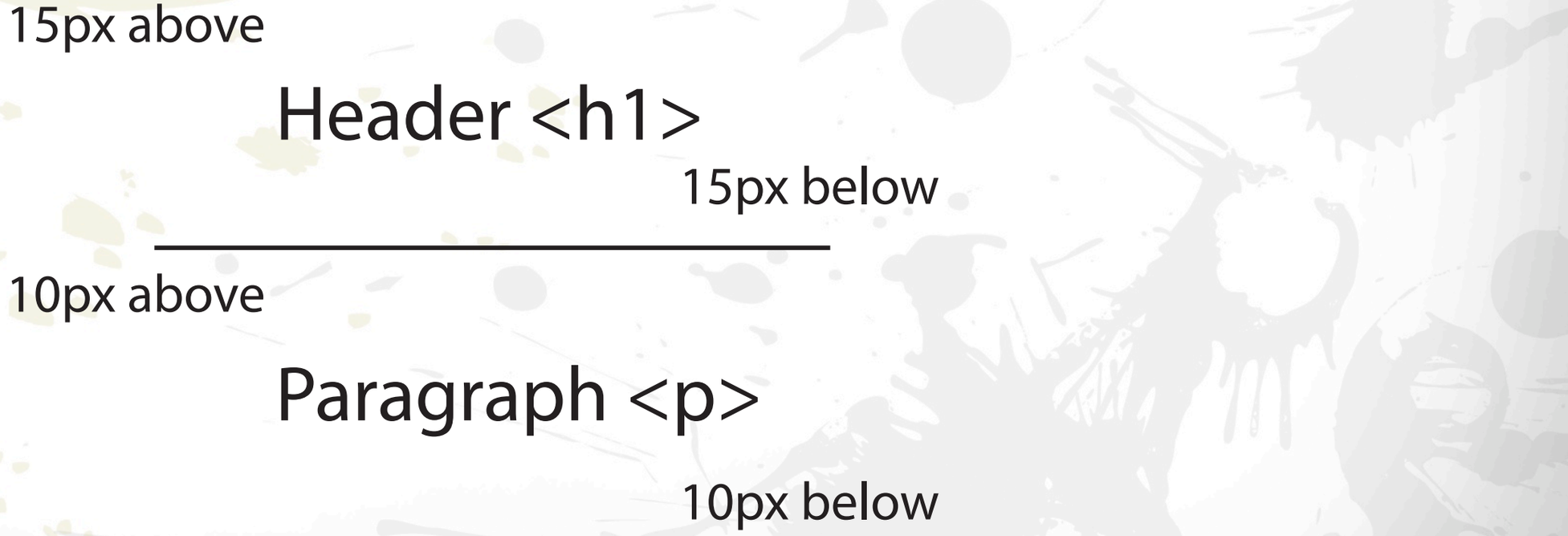
The elements after the floating element will flow around it.

The elements before the floating element will not be affected.

If an image is floated to the right, a following text flows around it, to the left:

Collapsing Margins

- Text elements “collapse” to the largest number. In example below, there would be 15px between the h1 and p elements, not added together.



Lets make something

Navigation

- using images and live text
- Dropdowns without .js – unless you want it
- selected items stay styled
- using good old ul tags make it easy
- The LoVe HAte rule – always in that order

a:Link
a:visited
a:hover
a:active

Lets Play!

Forms -what we can change via CSS

- Text: includes both the text for the form label and the text within the form field.
- Background: either a solid color or background image can be add contract to make fields stand out without being overwhelming.
- Padding
- Borders
- Margins

- Position= no tables needed
- Dimensions
- other(unfortunately, you can change the cursor and how it appears)
- Fieldsets: define the area around the form to set it apart from the rest of the page.
- Legends: label for fieldset
- Labels: Can be set using any tag, or none at all, but the advantage in using them is being able to click the label element itself. (Has to be a block element for this to work.)
  - These can be set with a width and alignment so no tables need to be harmed to create a simple form.

Example

```
label {
    font-weight: bold;
    line-height: normal;
    text-align: right;
    margin-right: 10px;
    position: relative;
    display: block;
    float: left;
    width: 125px;
}
```

- You also want a class associated with the label tag called something like fieldLabel. This class is used to override the general label style when you need a label that sits directly next to the form element with which it is associated (as with the radio buttons).

label.fieldLabel {display: inline; float: none;}

- Input: Not the best idea to set a universal style for input as there are many things associated with input.
- Text Input: Default appearance we can't alter, but we can change the border, background, forground color and font. (Safari doesn't allow you to change the border. A psudo-class can be applied for a rollover effect. And... of course, IE6 could care less because we'd be using a :hover and possibly :focus on elements other than links.
- Select Input: some control here. Safari still doesn't allow for you to change the Aqua theme.
- Radio/check Input: Not exactly a CSS styling dream here.
- Button Input: Firefox and IE allow a customized graphic and border change.

Lets form it

data data everywhere
What we already know, and can define

- **<table> Defines a table**
- **<th> Defines a table header**
- **<tr> Defines a table row**
- **<td> Defines a table cell**
- **<caption> Defines a table caption**
- **<colgroup> Defines groups of table columns**
- **<col> Defines the attribute values for one or more columns in a table**
- **<thead> Defines a table head**
- **<tbody> Defines a table body**
- **<tfoot> Defines a table footer**

CSS Table Properties

- border-collapse -- specifies whether or not table borders should be collapsed
  - values - collapse, separate, inherit
- border-spacing -- specifies the distance between the borders of adjacent cells
- values - length length, inherit
- caption-side -- specifies the placement of a table caption
- values - top, bottom, inherit
- empty-cells -- specifies whether or not to display borders and background empty cells in a table
- table-layout -- sets the layout algorithm to be used for a table
- values - auto, fixed, inherit

Lets Do it!

Frameworks in CSS?

- Frameworks can be a problem if you work with many other hands that might be involved.
- CSS Frameworks are made for ease of development, not for maintainability.
- Examples: Blueprint CSS, 960.gs, etc...

- In my opinion, Framework classes can be absolutely
meaningless outside the context of the framework.
- Frameworks are great at times, but they decrease maintainability.

IE6 (cough-- must die)

- IE as conditional statements in a separate style sheet, or in the same file?
- .IE6 .givemeMoreRoom {height:400px;} - same style sheet as main style or;
- <!--[if lt IE7]>
- (heigh:400px;)
- <![endif]-->
- in a IE specific style sheet
- (Prototype, YUI, Dojo, jQuery. JS libraries can provide a layer of abstraction and smooth out browser diferences, but they can also let you get lazy - meaning you can rely on js when a navigation might be better off straight CSS)

Resources

- www.w3schools.com/css
- http://www.positioniseverything.net/
- http://css.maxdesign.com.au/
- http://www.dezwozhere.com/links.html
- http://www.htmldog.com/
- http://www.sitepoint.com/
- http://cssglobe.com/lab/tablecloth/