

CIS 4004: Web Based Information Technology Spring 2013

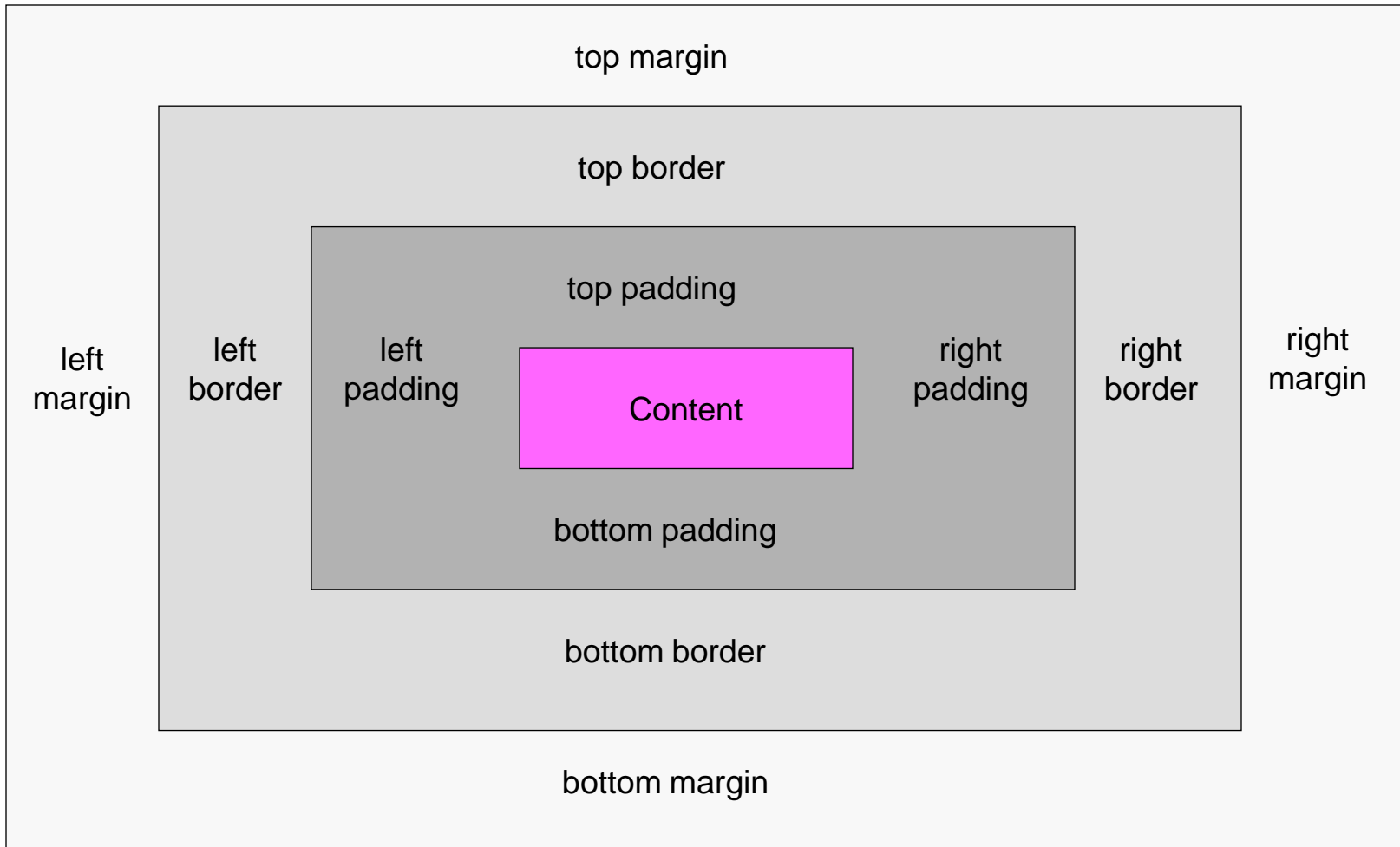
Advanced Page Layouts – Part 2

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
<http://www.cs.ucf.edu/courses/cis4004/spr2013>

Department of Electrical Engineering and Computer Science
University of Central Florida



The CSS Box Model



Three-Column Fluid Center Page Layouts

- In Advanced Page Layouts – Part 1, we examined several different approaches to handling a fixed-width multi-column page layout.
- The two primary approaches that we took were the more traditional nested `<div>s` and the new CSS3 `box-sizing:border-box` property. Hopefully, you have taken a closer look at both cases by now and are familiar with the techniques that were utilized in both cases.
- Our primary concern, in addition to achieving an overall pleasing appearance for our page, was to prevent “breaking” the page when the size of the content changed over time. The two approaches that we took were designed to prevent “float-slip”.



Three-Column Fluid Center Page Layouts

- In this set of notes, we focus on a three-column layout with a fluid center.
- In other words, we no longer have a fixed overall width to our page layout, but allow it to expand and contract as the user alters the size of their browser window.
- There are two basic ways to achieve a fluid center column:
 - Use negative margins to position the right column as the center changes size, or
 - Make the column containers behave like table cells using CSS3.
- The negative margins work well with older browsers, but the CSS3 table properties method is more simple and will work with the newer generation browsers, so we'll focus on that technique.

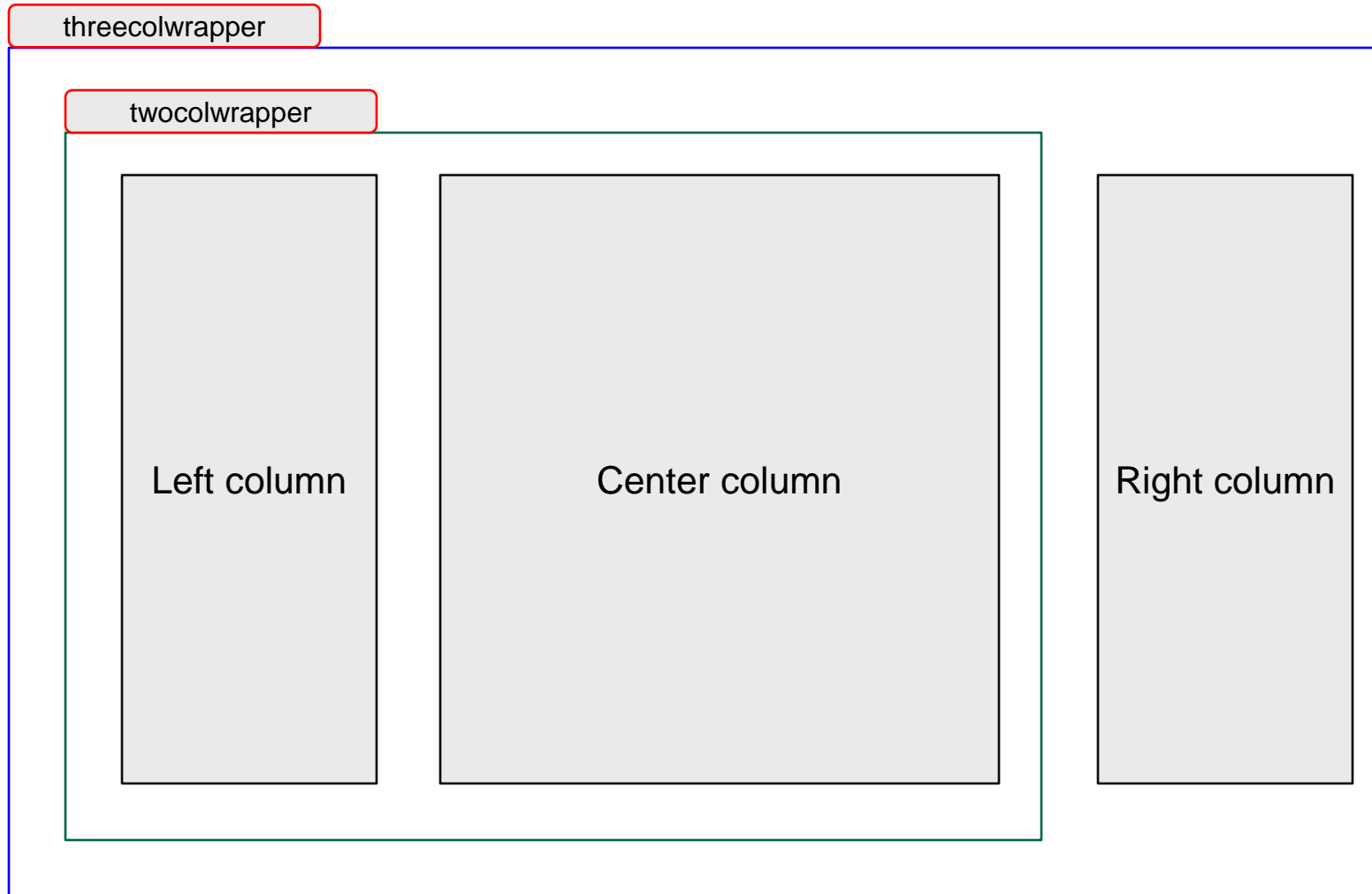


Three-Column Fluid Center Layout With Negative Margins

- The issue with creating a three-column layout where the center content area is fluid is managing the right column, and controlling its relationship to the layout as the center content area resizes.
- The solution using negative margins was originally developed by Web designer Ryan Brill. His solution manipulates the margins of two wrapper elements (`<div>` elements): the first one around all three columns and a second one around just the left and center columns. The effect is shown on the next page.



Three-Column Fluid Center Layout With Negative Margins



Three-Column Fluid Center Layout With Negative Margins

- Since we dealt with the fixed width layout fairly extensively in the previous set of notes, I won't develop this layout in the same step-by-step fashion.
- Instead, I'll point out some of the highlights in the CSS that allows for the center column to be fluid. The next couple pages illustrate the fluid nature of the center column. Notice that as the browser window is expanded the center column gets wider (it also gets shorter since the content moves up). Notice too, that the left and right columns do not change size, only the center column is fluid.
- As shown in the previous page, the CSS effect that allows this behavior is the multiple layers of `<div>` elements, combined with negative margins applied to the right column.



A Three-Column Fluid Center Layout

- Nav Link 1
- Nav Link 2
- Nav Link 3
- Nav Link 4

About This Layout

This page is styled with CSS. This layout uses negative margins to create this feature-rich layout.

Key Features

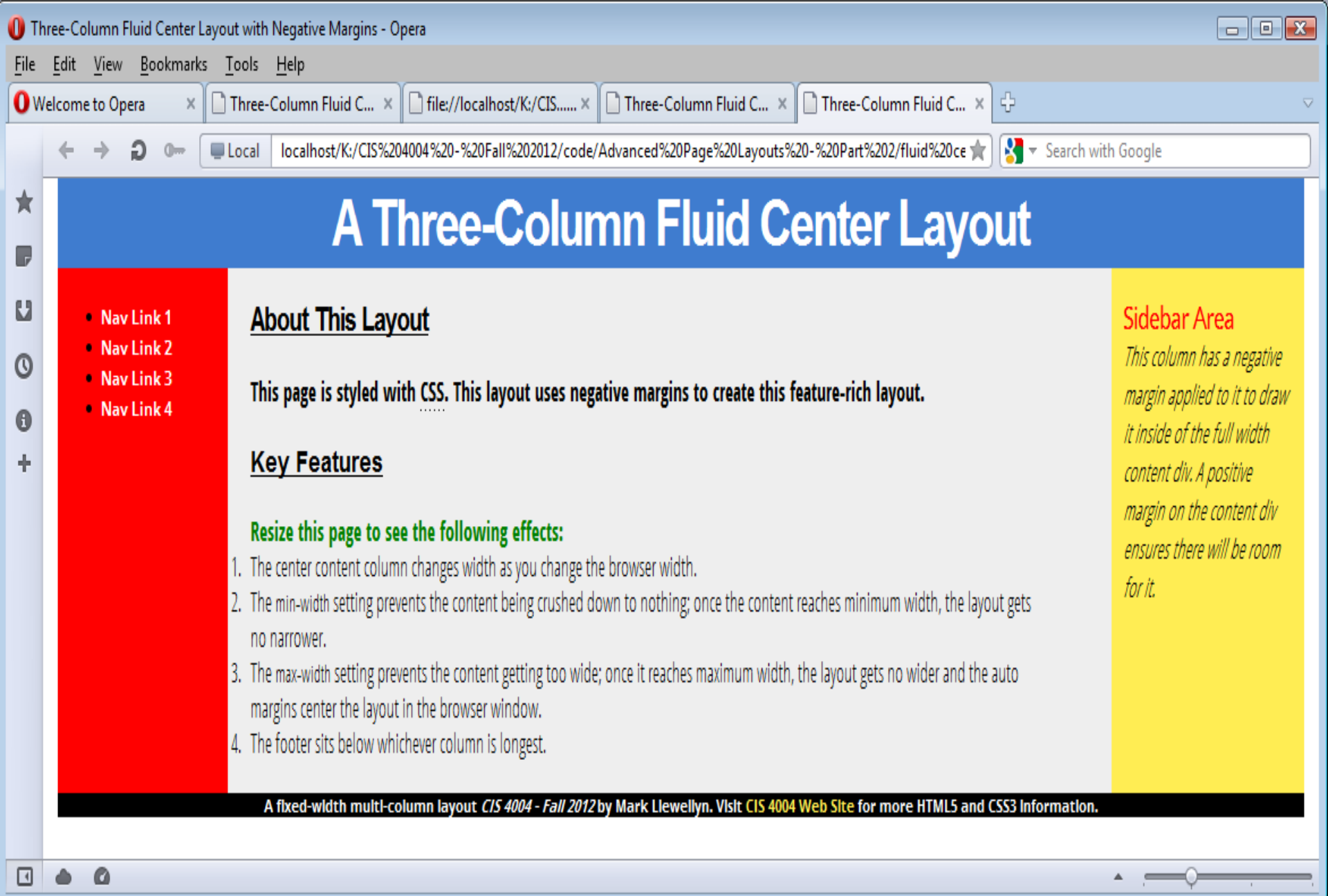
Resize this page to see the following effects:

1. The center content column changes width as you change the browser width.
2. The min-width setting prevents the content being crushed down to nothing; once the content reaches minimum width, the layout gets no narrower.
3. The max-width setting prevents the content getting too wide; once it reaches maximum width, the layout gets no wider and the auto margins center the layout in the browser window.
4. The footer sits below whichever column is longest.

Sidebar Area

This column has a negative margin applied to it to draw it inside of the full width content div. A positive margin on the content div ensures there will be room for it.





A Three-Column Fluid Center Layout

- Nav Link 1
- Nav Link 2
- Nav Link 3
- Nav Link 4

About This Layout

This page is styled with CSS. This layout uses negative margins to create this feature-rich layout.

Key Features

Resize this page to see the following effects:

1. The center content column changes width as you change the browser width.
2. The min-width setting prevents the content being crushed down to nothing; once the content reaches minimum width, the layout gets no narrower.
3. The max-width setting prevents the content getting too wide; once it reaches maximum width, the layout gets no wider and the auto margins center the layout in the browser window.
4. The footer sits below whichever column is longest.

Sidebar Area

This column has a negative margin applied to it to draw it inside of the full width content div. A positive margin on the content div ensures there will be room for it.

A fixed-width multi-column layout: CIS 4004 - Fall 2012 by Mark Llewellyn. Visit [CIS 4004 Web Site](#) for more HTML5 and CSS3 Information.



Three-Column Fluid Center Layout With Negative Margins

- Here's how the fluid center column with negative margins works. You should look at the markup and CSS to help you understand what's happening.
- The right column (the `<aside>`) is 210 pixels wide. There is a right margin of 210 pixels on the center column (the `<article>` element) to create space for the right column, although doing this pushes the right column 210 pixels away. In other words, it would not have room for it and would slip under the left column. However, a negative right margin of 210 pixels is applied to the two-column wrapper around the left and center columns, which pulls the right column back up and into the space created by the right margin on the `<article>` element.



Three-Column Fluid Center Layout With Negative Margins

- The `<article>` element (center column) which is sized at 100%, still claims 100% of the space that remains after the left column is floated into position, but its right margin leaves space for the right column to be pulled into position by the negative margin on the inner wrapper.
- Another feature of this layout is the full height columns. Notice that the left and right columns are now the same height as the center column in this layout. In our previous version (the fixed-width multi-column layout), the left and right columns extended only as far as their content allowed.
- One technique for accomplishing this effect, and the one that I used here, is known as “faux columns”.



Three-Column Fluid Center Layout With Negative Margins

- Faux columns actually only make you think that the columns are full height, actually, they are not, hence the name.
- This technique involves adding background graphic elements that are the same width and background color as the columns into the page's wrapper elements behind the actual columns.
- The wrapper elements, unlike the columns themselves, are the full height of the content area, so by repeating the background graphics vertically, they visually extend the columns down the page.
- Just to make the effect more pronounced here, I used different color backgrounds for the left and right columns as well as for the fluid center column.



Three-Column Fluid Center Layout With Negative Margins

- Again, look at the CSS as you read this explanation and it will help you understand what is happening.
- The `main_wrapper` `div` element is used for the left column's graphic and the `threecolwrap` `div` element is used for the right graphic and it is positioned to the `div` element's right edge.
- The background of the `main_wrapper` is set to the color of the background of the center column. This background color actually fills the background of the entire layout, but the graphics on the side columns, the header, and the footer overlay the `main_wrapper`, (recall that child elements overlay parent elements), so you only see its color in the center area.



Three-Column Fluid Center Layout With Negative Margins

- One addition item that you may have noticed looking at the CSS is that I added horizontal margins to the content elements using the child-star selector, e.g., `nav > * {margin: 0 10px;}`, instead of adding this padding to the inner `div` elements.
- Since the star selector means “all elements”, if you write a selector followed by the star (asterik) such as `someSel *`, then you’ll select all descendant elements of the `someSel`.
- The child selector `>` means “child of”, so if you add the child selector before the `*`, such as `someSel > *` you will select all the immediate child elements, but no other descendants.
- This is exactly the effect we want in this situation in order to push all the top-level elements inside a container away from the edge.



Three-Column Fluid Center Layout With Negative Margins

- For example, `article > * {margin: 0 20px:}` will add 20-pixel left and right margin to every child element, but not to other descendant elements.
- If you need to add more indentation on one or more of the child elements in addition to this amount, you would need to set that by applying the padding or margins to the child element itself.
- So there you have a fluid center three column layout with full CSS styling. The full markup is available on the course website.



Three-Column Fluid Center Layout With CSS Table Properties

- We've spent the entire semester thus far making it very clear that content, presentation, and behavior of a web page must be separated.
- This would mean that it is completely unacceptable to add table elements to your HTML markup in order to create a multi-column layout.
- However, using CSS3 to make your layout behave like a table is completely acceptable and will become the norm as CSS3 becomes more universally adopted.
- Using CSS3 to make a layout behave like a table does not cause a permanent table-based presentation that cannot be restyled for, say presentation in a single-column layout on a mobile device.



Three-Column Fluid Center Layout With CSS Table Properties

- Tables actually behave in very desirable ways when it comes to creating layouts.
- Tables are, in their most basic form, three elements: a table wrapper `<table>` containing row wrappers `<tr>` for each row of table data cells `<td>`, like this:

```
<table>
  <tr>
    <td> Cell 1</td> <td> Cell 2</td> <td> Cell 3</td>
  </tr>
  <tr>
    <td> Cell 1</td> <td> Cell 2</td> <td> Cell 3</td>
  </tr>
</table>
```

- The CSS3 display property of any HTML element can be changed to table, table-row, or table-cell and it will then simulate the behavior of its HTML table equivalent.



Three-Column Fluid Center Layout With CSS Table Properties

- The big advantages of marking up the CSS columns in a layout as table cells are:
 - Table cells sit side by side like columns without floating, so padding can be added directly to them without breaking the layout, i.e., there is no float slip.
 - All table cells in a row are the same height by default so no faux column effects are needed.
 - Any column that is not explicitly given a set width is fluid.
- These three behaviors solve the problems of the layouts that we just been dealing with, however, there is a slight catch. CSS3 table properties are not supported by IE7 and below. However, if you are willing (and your clients are willing) to drop IE7, this is the way to go and will quickly become a web standard.



Three-Column Fluid Center Layout With CSS Table Properties

- Using this technique, you don't even need `<div>` wrappers to act as the `<table>` and `<tr>` elements – you just need to set the `display` property of the three columns to `table-cell`.
- Browser rendering engines automatically add a table row element around a contiguous set of table cells if the table row is missing, and wrap a table element around table rows if the table element is missing, so you don't need any extra markup.
- A sample of the markup that uses this technique is shown on the next page with some sample renderings on the following pages. The entire markup for this version is available on the course web site.



```
font-weight:400;
font-size:4em;
letter-spacing:-.05em;
color:#fff;
padding:0 0 5px 10px;
}

nav {
width:150px;
display:table-cell; /*was float:left; */
padding:10px;
background:#dcd9c0;
}

nav li {
list-style-type:none;
}

nav a {
font-family:'Open Sans Condensed';
font-weight:700;
font-size:1.2em;
color:#616161;
}

article {
/* removed original width setting was width:600px; */
display:table-cell; /*was float:left; */
padding:10px 20px;
background:#ffed53;
}

article h1 {
font-family:'Droid Sans';
font-weight:700;
```



A Multi-row, Multi-column Layout

- Let's focus on a more complex, more “real-world” page layout framework. This might be used as the basis for a corporate Web site.
- The layouts we've focused on so far have contained only one `<article>` element, one `<nav>` elements, etc., so selecting these elements is easy – you merely use the tag name.
- As you create more complex pages, you will create multiple instances of such elements in one page. You will then need to use contextual selectors to differentiate between them. One of the main things we'll focus on in the next layout is the proper way to add a minimal number of classes and ids into the markup to enable you to accurately select any element in the page.




A Multi-row, Multi-column Layout

- We looked at CSS3 properties such as `box-sizing: border-box` in the last section of notes that allows you to use inner `div` elements to pad content. However, inner elements, whether `div` elements or other types of elements, can also be used for a variety of stylistic purposes, and we'll illustrate some of those in this next layout.
- The layout, as illustrated on the next page, consists of six full-width rows, where the fourth row has three columns and the fifth row has four columns.



A Multi-row, Multi-column Layout Using CSS3

Navigation menus go here
[Link 1](#) [Link 2](#) [Link 3](#) [Link 4](#) [Link 5](#)



This is just some text. It is repeated over and over to give some width and height to this element. This is just some text. It is repeated over and over to give some width and height to this element. This is just some text. It is repeated over and over to give some width and height to this element. This is just some text. It is repeated over and over to give some width and height to this element.

This is just some text. It is repeated over and over to give some width and height to this element. This is just some text. It is repeated over and over to give some width and height to this element. This is just some text. It is repeated over and over to give some width and height to this element. This is just some text. It is repeated over and over to give some width and height to this element.

This is just some text. It is repeated over and over to give some width and height to this element. This is just some text. It is repeated over and over to give some width and height to this element. This is just some text. It is repeated over and over to give some width and height to this element. This is just some text. It is repeated over and over to give some width and height to this element.

This is the text in the promo area. As before it is repeated several times to add some depth. This is the text in the promo area. As before it is repeated several times to add some depth.

This is the text in the promo area. As before it is repeated several times to add some depth. This is the text in the promo area. As before it is repeated several times to add some depth.

This is the text in the promo area. As before it is repeated several times to add some depth. This is the text in the promo area. As before it is repeated several times to add some depth.

This is the text in the promo area. As before it is repeated several times to add some depth. This is the text in the promo area. As before it is repeated several times to add some depth.

A multi-row, multi-column layout using CSS3 table properties CIS 4004 - Fall 2012 by Mark Llewellyn. Visit CIS 4004 Web Site for more HTML5 and CSS3 information.



A Multi-row, Multi-column Layout

- As page layouts get more complex, the same HTML elements types (section, article, nav, etc.) are used in many places – for example the layout shown on the previous page has seven article elements in it.
- You need a way to differentiate between these identically named tags in order to select the one you want.
- The wrong solution to this is to label the seven article elements with seven different class names, which is a common approach to newcomers to CSS.
- This is an inappropriate use of classes – they are intended to identify elements with common characteristics, i.e., ones that belong to the same “class”.



A Multi-row, Multi-column Layout

- Overuse of classes clutters the markup and makes the CSS hard to read and even harder to deal with because you have to constantly refer to the markup to remind yourself where each class name is located.
- A much better approach is to add IDs to the top level of each main section of your markup (this is the correct use for IDs, which should be used to identify unique elements on a page). These IDs then act as your means to hook into the HTML when used at the start of the contextual selectors that define the path to the descendants you want to select.
- This approach minimizes the class and ID attributes in your markup, and the resultant contextual selectors state a clear path to the element(s) that they reference, making the CSS more readable.





```
40     section#feature_area article {
41         float:left;
42         width:320px;
43         padding:10px 0;
44         background:#fff;
45         border-top:4px solid #f7be84;
46     }
47     section#feature_area article .inner {
48         margin:10px 20px;
49         padding:5px;
50         background:#fff;
51         border:5px solid;
52     }
53     section#feature_area article:nth-child(2) {
54         background:#e5e8ed;
55     }
56     section#feature_area article:nth-child(1) .inner {
57         border-color:#d7dd6f;
58     }
59     section#feature_area article:nth-child(2) .inner {
60         border-color:#f6dec5;
61     }
62     section#feature_area article:nth-child(3) .inner {
63         border-color:#d1d8e4;
```

