Page Layout

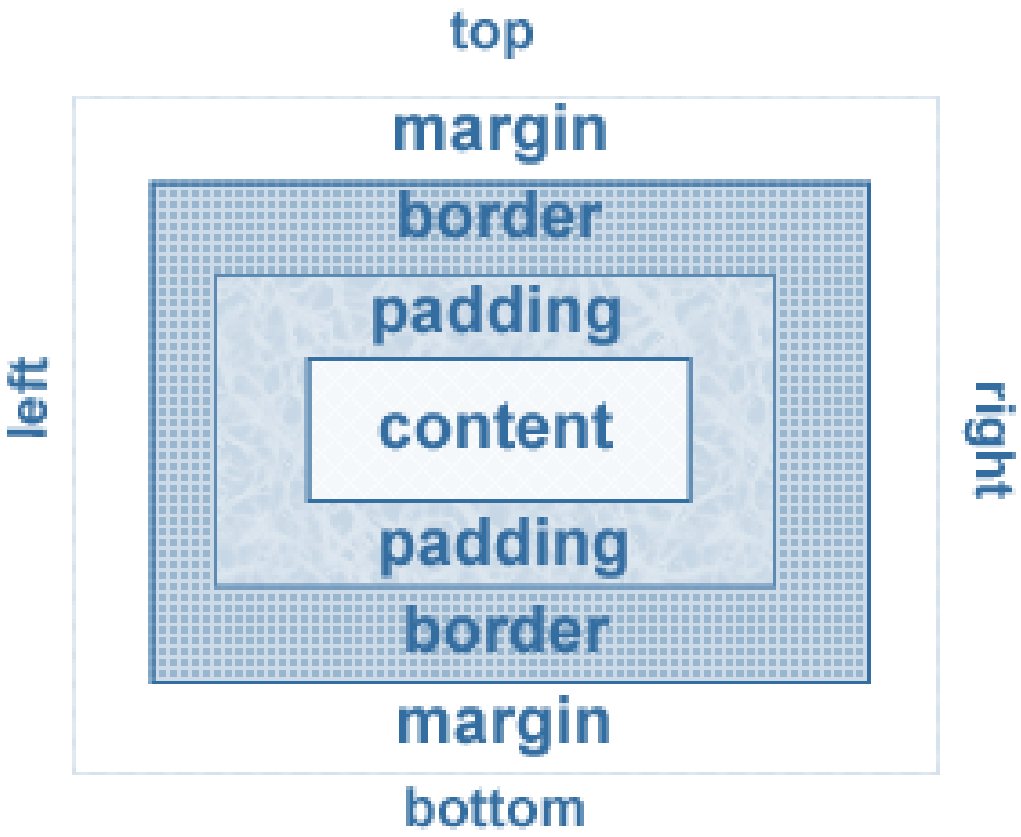
With HTML5 & CSS3

Agenda

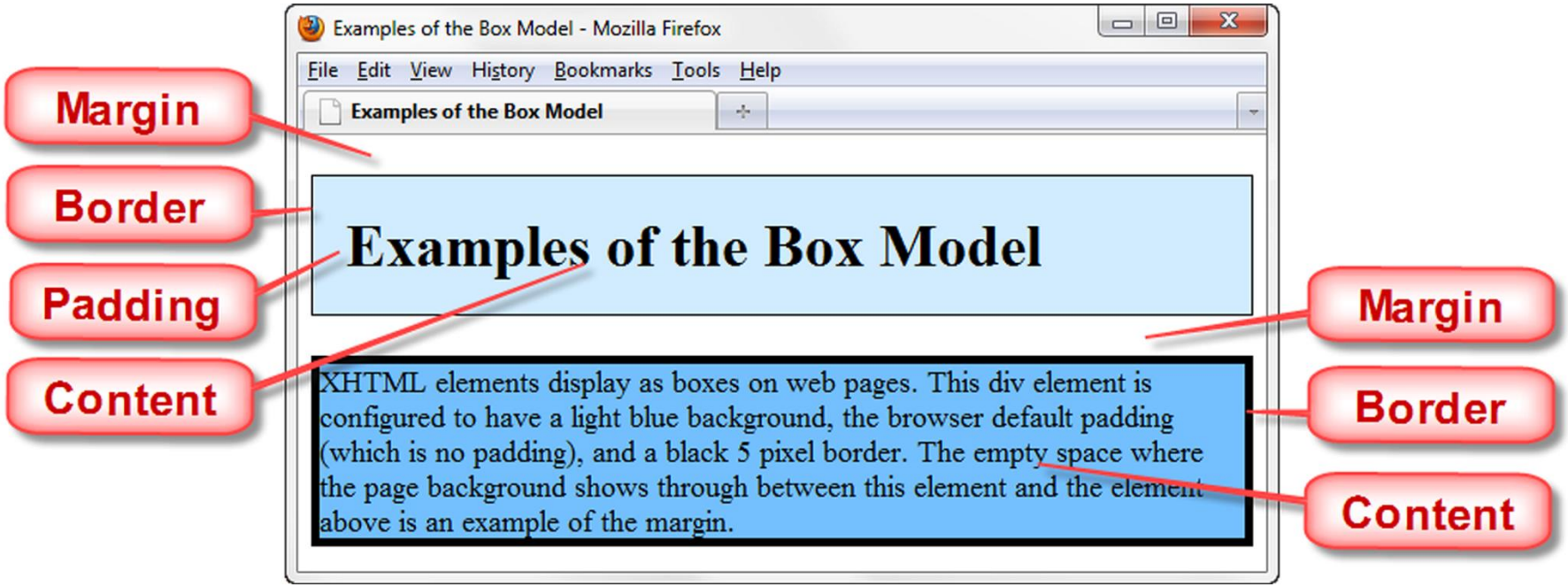
- The CSS Box Model
- Positioning with CSS
- Columns
- Navigation lists

The Box Model

- **Content**
  - Text & web page elements in the container
- **Padding**
  - Area between the content and the border
- **Border**
  - Between the padding and the margin
- **Margin**
  - Determines the empty space between the element and adjacent elements



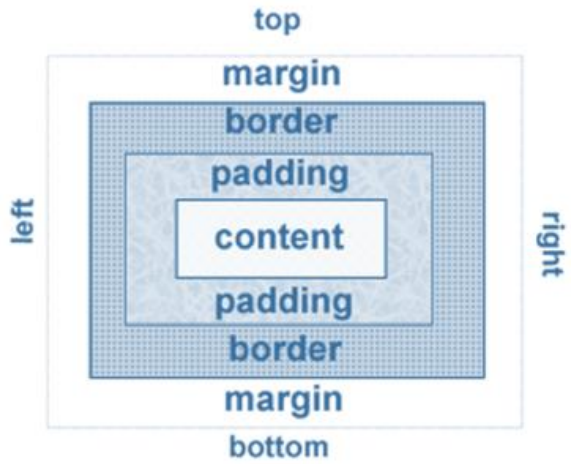
Box model in Action



Configure Margin with CSS

- The margin property
  - Configures empty space between the element and adjacent elements
- Related properties:
  - margin-top, margin-right, margin-bottom, margin-left
- Syntax examples

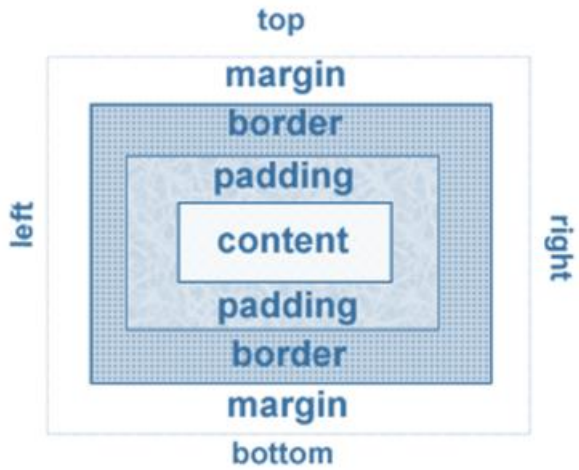
```
h1 { margin : 0; }
h1 { margin : 20px 10px; }
h1 { margin : 10px 30px 20px; }
h1 { margin : 20px 30px 0 30px; }
```



Configure Padding with CSS

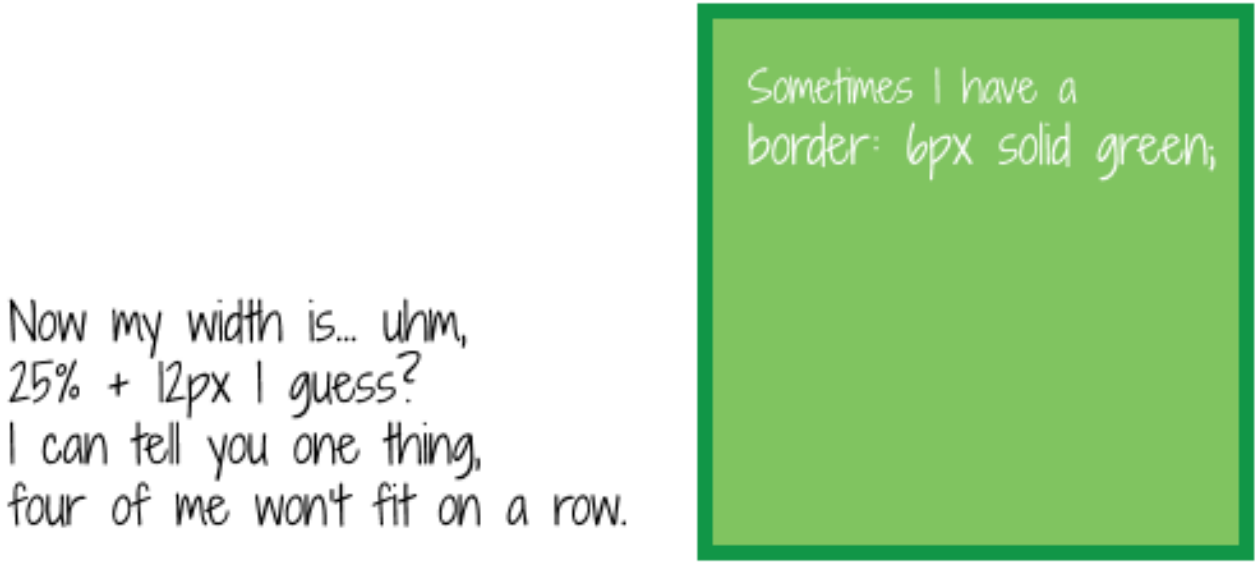
- The padding property
  - Configures empty space between the content of the HTML element (such as text) and the border
- Related properties:
  - padding-top, padding-right, padding-bottom, padding-left
- Syntax examples

```
h1 { padding : 0; }
h1 { padding : 20px 10px; }
h1 { padding : 10px 30px 20px; }
h1 { padding : 20px 30px 0 30px; }
```



Overriding box-sizing

- The default value for box-sizing is content-box!
  - With the default box-sizing, as soon as an element has either padding or border applied, the actual rendered width is wider than the width you set
  - **Actual width = width + border-left + border-right + padding-left + padding-right**



Overriding box-sizing

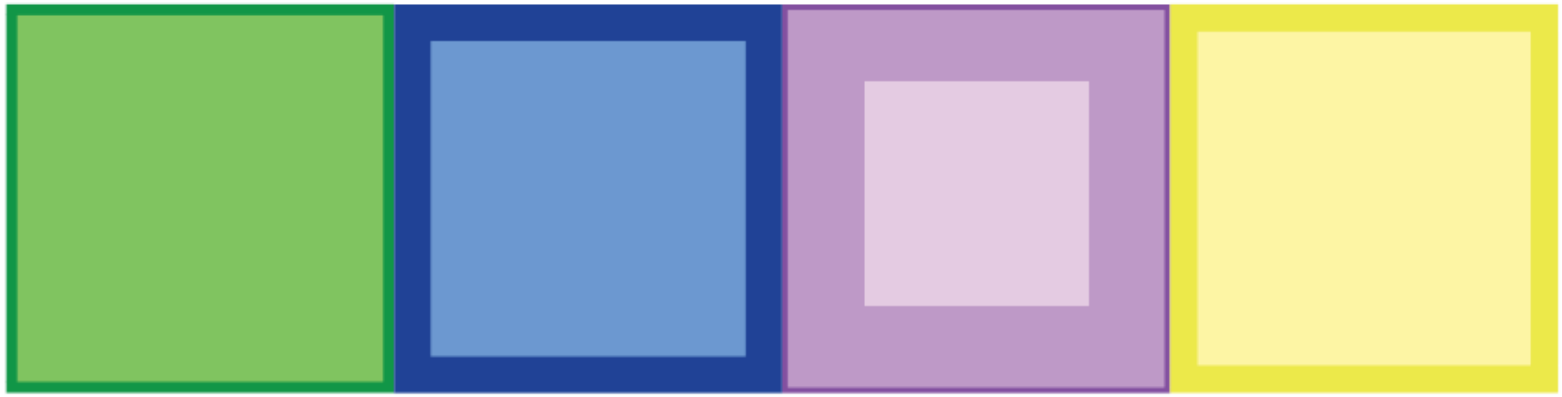
Normalize and Bootstrap will do this for you!

- With

  `box-sizing: border-box`

  the padding and border press their way inside the box rather than expand the box
- The result is a box the exact width you set it to be and can count on

```
*, *:before, *:after {
      box-sizing: border-box;
}
```



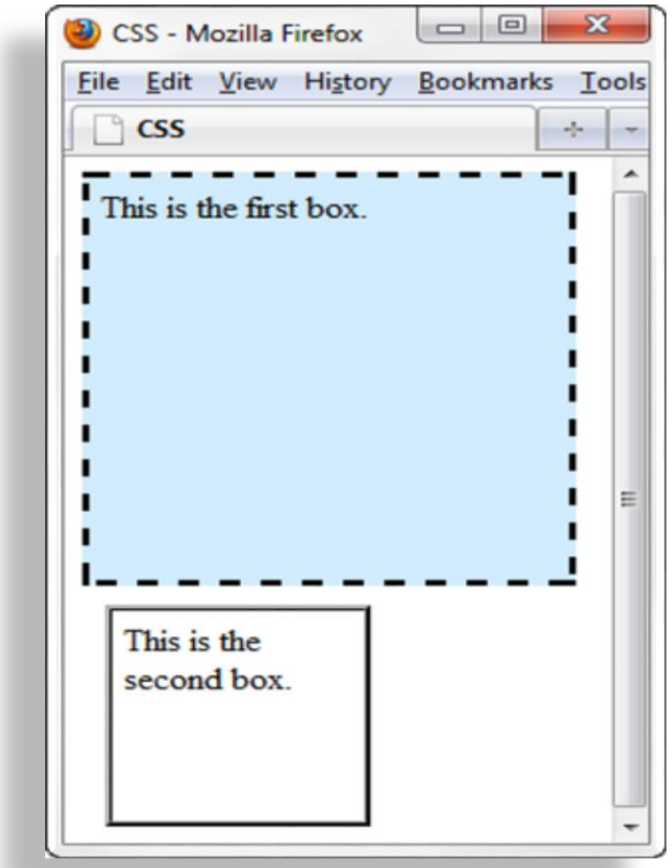
Four of me can sit in a row no matter what border and padding we have.

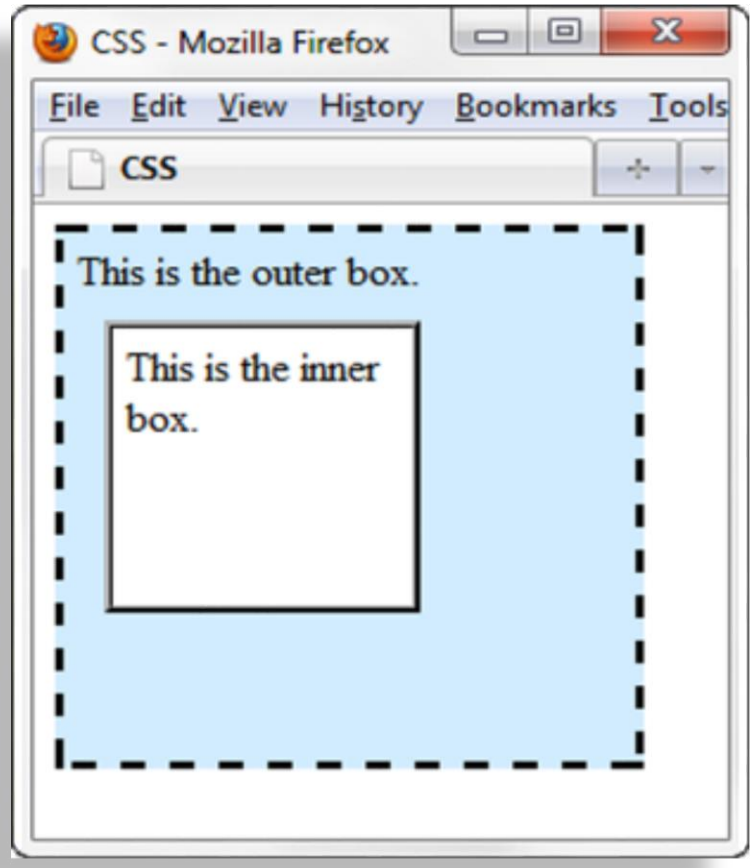
life = easy street.

POSITIONING WITH CSS

Normal Flow

- Browser display of elements in the order they are coded in the Web page document

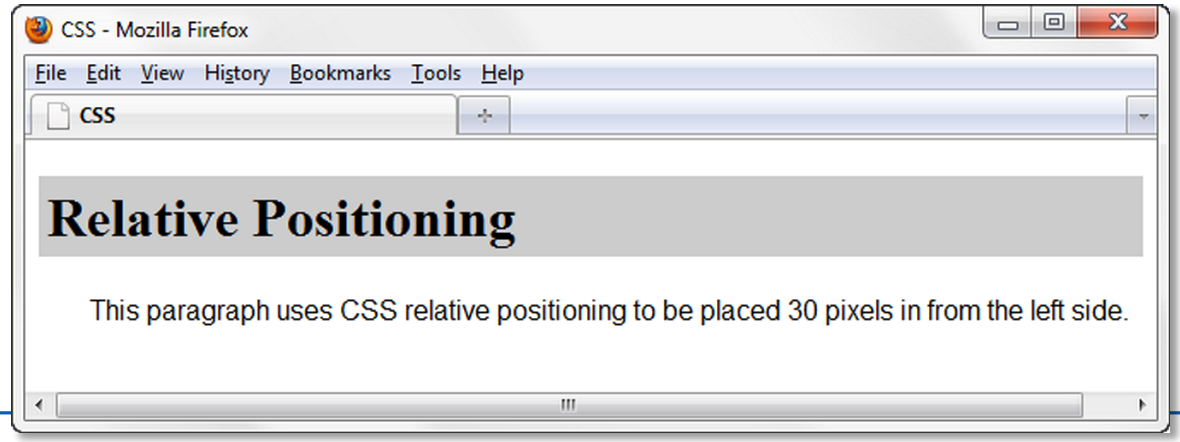




Relative Positioning

- Changes the location of an element in relation to where it would otherwise appear

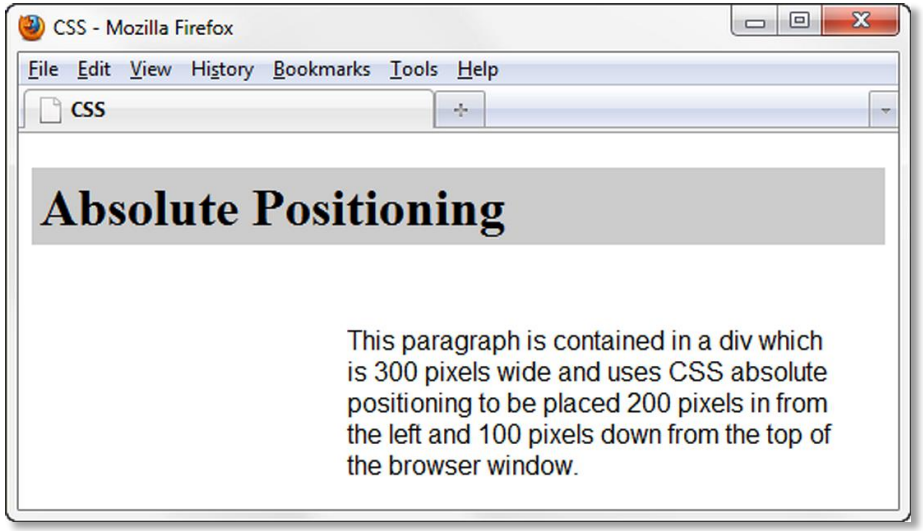
```
h1 {
        background-color:#cccccc;
        padding: 5px;
        color: #000000;
}
#myContent {
        position: relative;
        left: 30px;
        font-family: Arial,sans-serif;
}
```



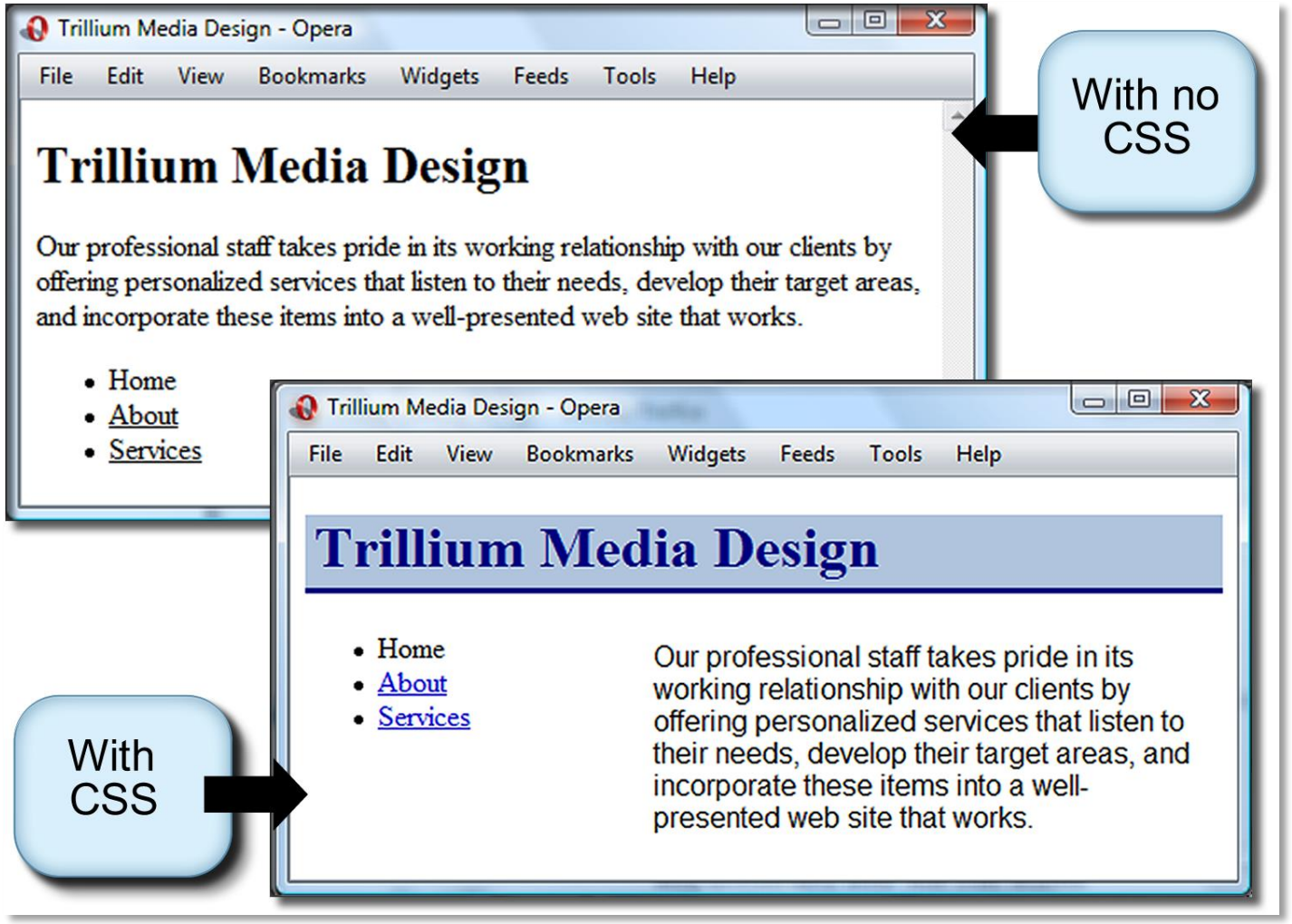
Absolute Positioning

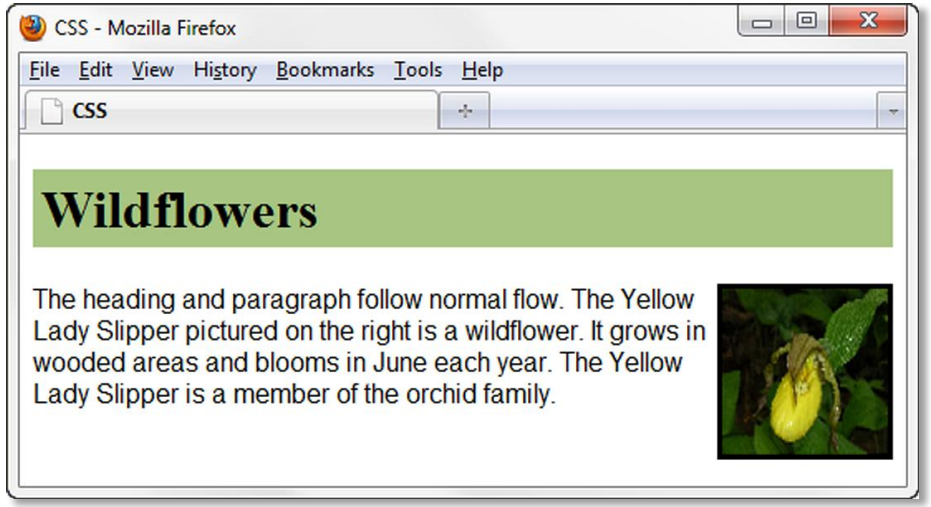
- Precisely specifies the location of an element in the browser window

```
h1 {
        background-color: #cccccc;
        padding: 5px;
        color: #000000;
}
#content {
        position: absolute;
        left: 200;
        top: 100;
        font-family: Arial,sans-serif;
        width: 300;
}
```



Absolute Positioning Example





float Property

- Elements that seem to "float" on the right or left side of either the browser window or another element are often configured using the float property

```
h1 {
        background-color : #cccccc;
        padding : 5px;
        color : #000000;
}
p {
        font-family:Arial,sans-serif;
}
#yls {
        float : right;
        margin : 0 0 5px 5px;
        border : solid;
}
```

clear Property

The h2 text is displayed in normal flow.

- Useful to "clear" or terminate a float
- Values are left, right, and both

`clear: left;` was applied to the **h2**. Now the h2 text displays AFTER the floated image.

overflow Property

The background does not extend as far as you'd expect.

- Intended to configure the display of elements on a Web page.
- However, it is useful to "clear" or terminate a float before the end of a container element
- Values are auto, hidden, and scroll

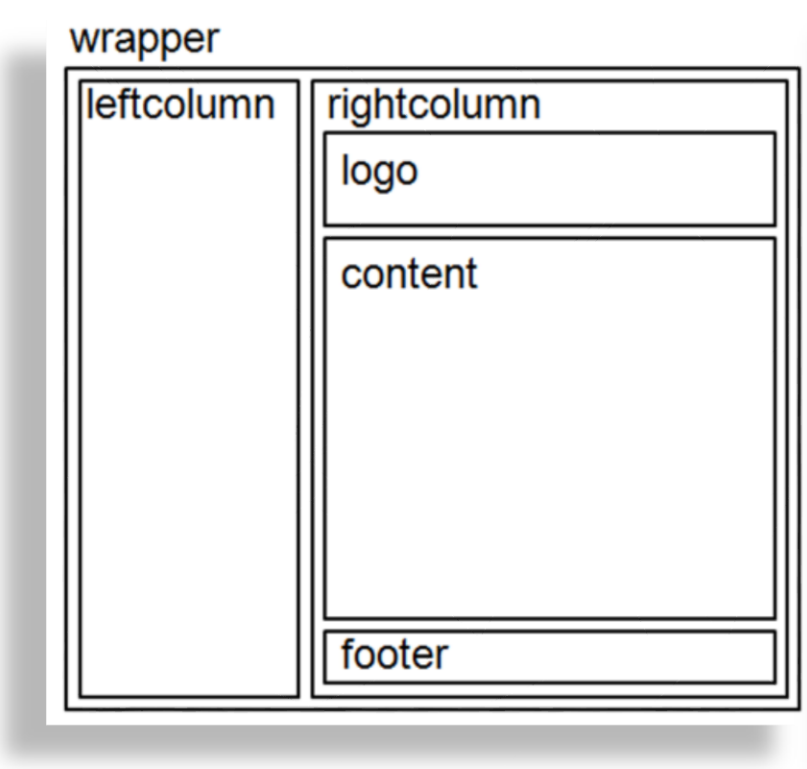
`overflow: auto;` was applied to the **div** that contains the image and paragraph. Now the background extends and the h2 text displays AFTER the floated image.

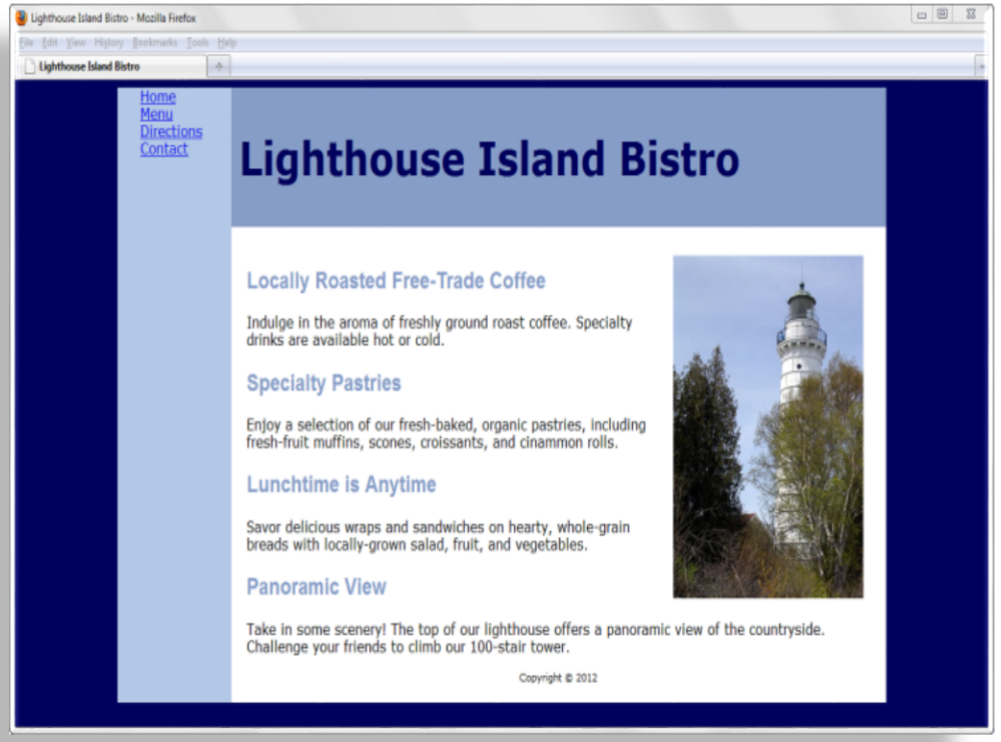
Display Property

- Configures how and if an element is displayed
  - display: none;
    - The element will not be displayed.
  - display: block;
    - The element is rendered as a block element – even if it is actually an inline element, such as a hyperlink.
  - display: inline;
    - The element will be rendered as an inline element – even if it is actually a block element – such as a <li>.

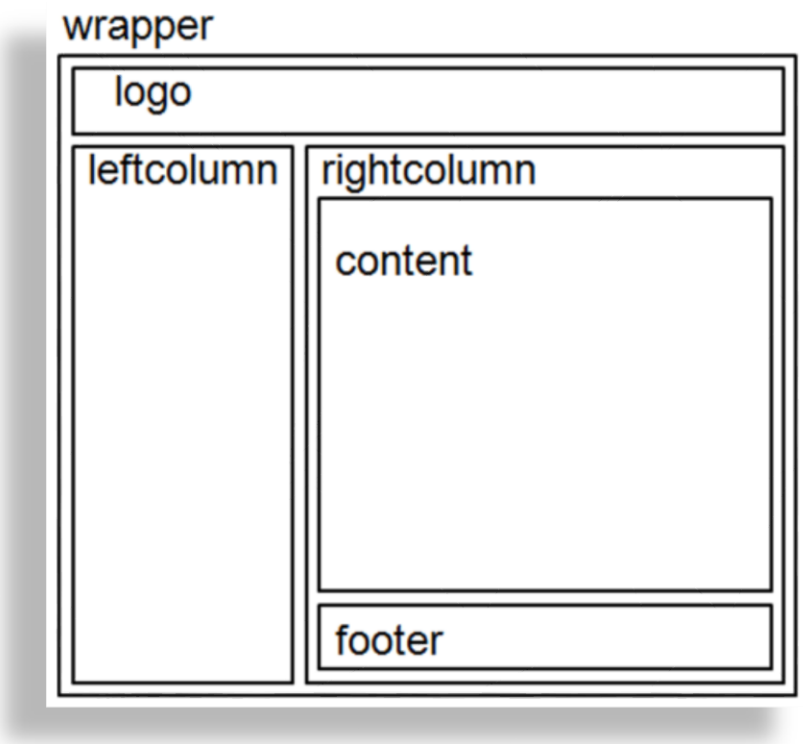
COLUMNS

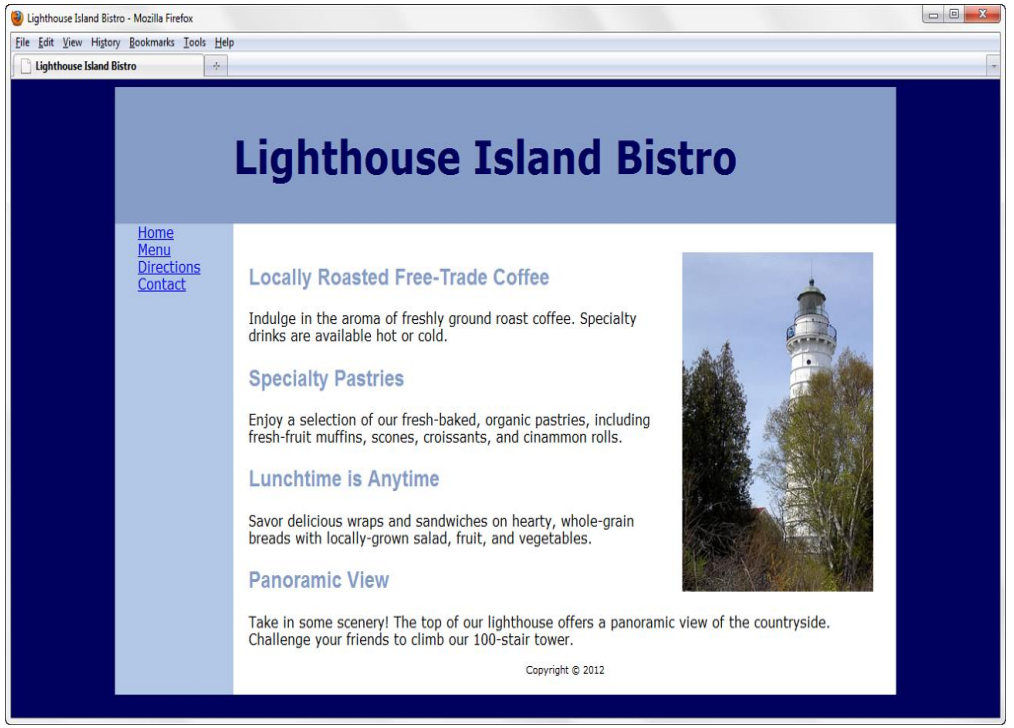
Two Columns (left nav)





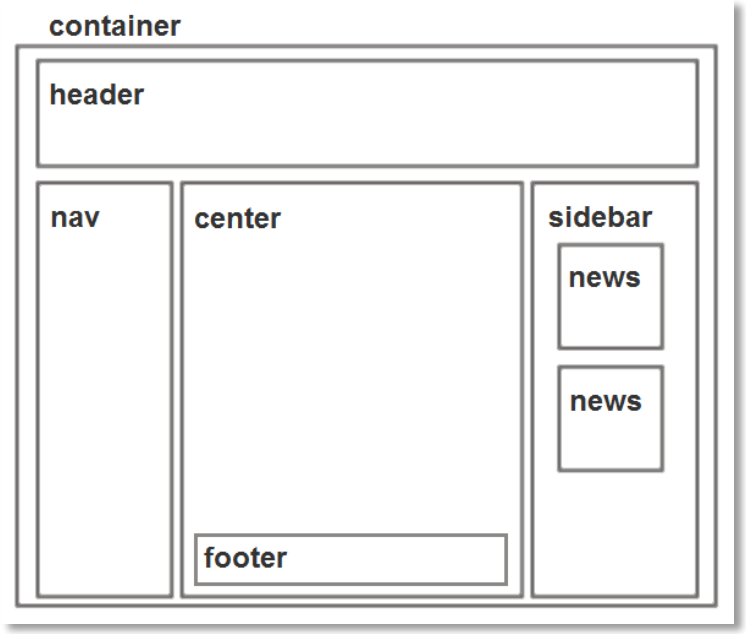
Two Columns (top logo, left nav)



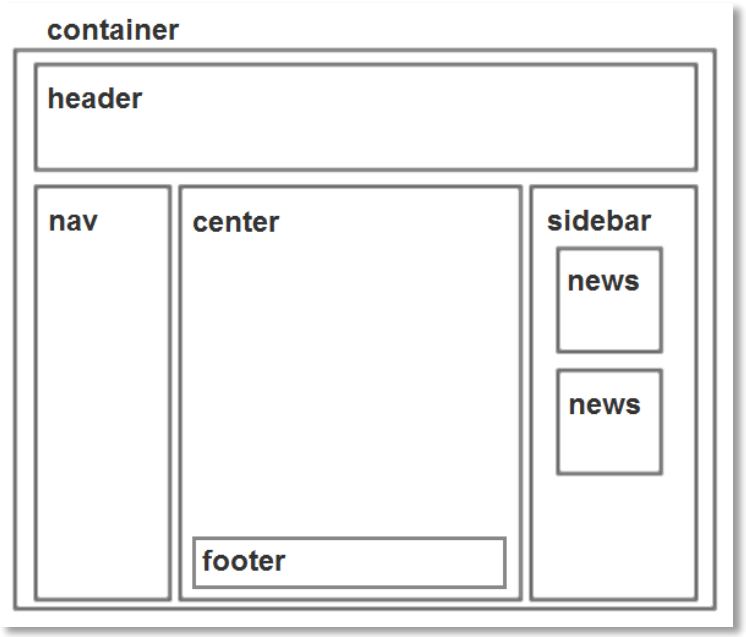


Three Column Page Layout

- A common web page layout consists of a header across the top of the page with three columns below:
  navigation, content, and sidebar.



Three Column Layout



- **Container** sets default background color, text color, font typeface, and a minimum width
- **Left-column navigation**
  - float: left; width:150px;
- **Right-column content**
  - float: right; width: 200px;
- **Center column**
  - Uses the remaining screen room available room after the floating columns display
  - margin: 0 210px 0 160px;
- **Footer** – clears the float
  - clear: both;

NAVIGATION LISTS

Configure Hyperlinks in an Unordered List

- **Vertical Navigation**

```
<div id="leftcolumn">
   <ul>
      <li><a href="index.html">Home</a></li>
      <li><a href="menu.html">Menu</a></li>
      <li><a href="directions.html">Directions</a></li>
      <li><a href="contact.html">Contact</a></li>
   </ul>
</div>
```

- CSS removes the list marker and underline:

```
#leftcolumn ul { list-style-type: none; }
#leftcolumn  a { text-decoration }
```

Configure Hyperlinks in an Unordered List

- **Horizontal Navigation**
- HTML:

```
<div id="nav">
<ul>
      <li><a href="index.html">Home</a></li>
      <li><a href="menu.html">Menu</a></li>
      <li><a href="directions.html">Directions</a></li>
      <li><a href="contact.html">Contact</a></li>
</ul>
</div>
```

- CSS removes the list marker, removes the underline, adds padding, and configures the list items for inline display

```
#nav ul { list-style-type: none;}
#nav a { text-decoration: none;
         padding-right: 10px; }
#nav li {  display: inline; }
```

Deciding to Configure a class or id

- Configure a class:
  - If the style may apply to more than one element on a page
  - Use the . (dot) notation in the style sheet
  - Use the class attribute in the HTML

- Configure an id:
  - If the style is specific to only one element on a page
  - Use the # notation in the style sheet
  - Use the id attribute in the HTML

Choosing a Name for a class or an id

- A class or id name should be descriptive of the purpose:
  - such as nav, news, footer, etc
  - Bad choice for a name: redText, bolded, blueborder, etc

CSS Debugging Tips

- Manually check syntax errors
- Use W3C CSS Validator to check syntax errors
  - http://jigsaw.w3.org/css-validator/
- Configure temporary background colors
- Configure temporary borders
- Use CSS comments to find the unexpected
  - /* the browser ignores this code */
- Don’t expect your pages to look exactly the same in all browsers!
  - Unless you use normalize.css
- Be patient!

ARIA Roles

- As part of the Web Accessibility Initiative (WAI), the Accessible Rich Internet Applications Suite (ARIA), defines a way to make Web content and Web applications more accessible
- It is used to improve the accessibility of dynamic content and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies
- ARIA roles work now in many browsers and screen readers. When they don't, they are harmless

```
<header role="banner">
<nav role="navigation">
```

- More info: http://www.webteacher.ws/2010/10/14/aria-roles-101/

References & Links

- normalize.css
  http://necolas.github.io/normalize.css/
- Modern normalize
  https://github.com/sindresorhus/modern-normalize