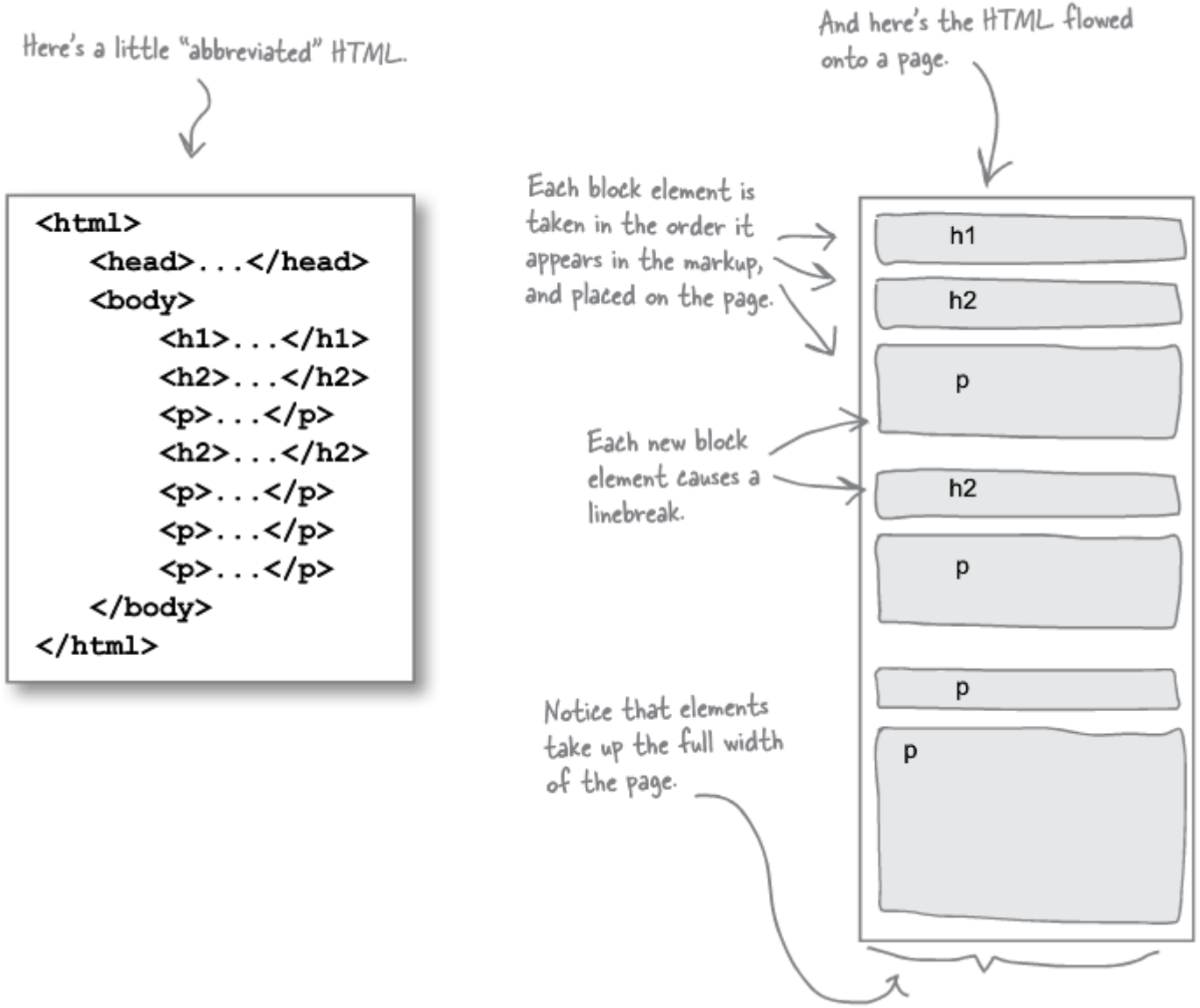
Lecture 11: Flow, Arranging elements, Common Layout Techniques

Dr. Jason Schanker

Flow: How the browser lays out pages

- The browser uses flow to lay out the pages. Unless told otherwise, the browser puts the block elements listed at the top of the HTML on the top of the page and proceeds downward through the HTML, displaying each block element below the previous one, separating the blocks with linebreaks.

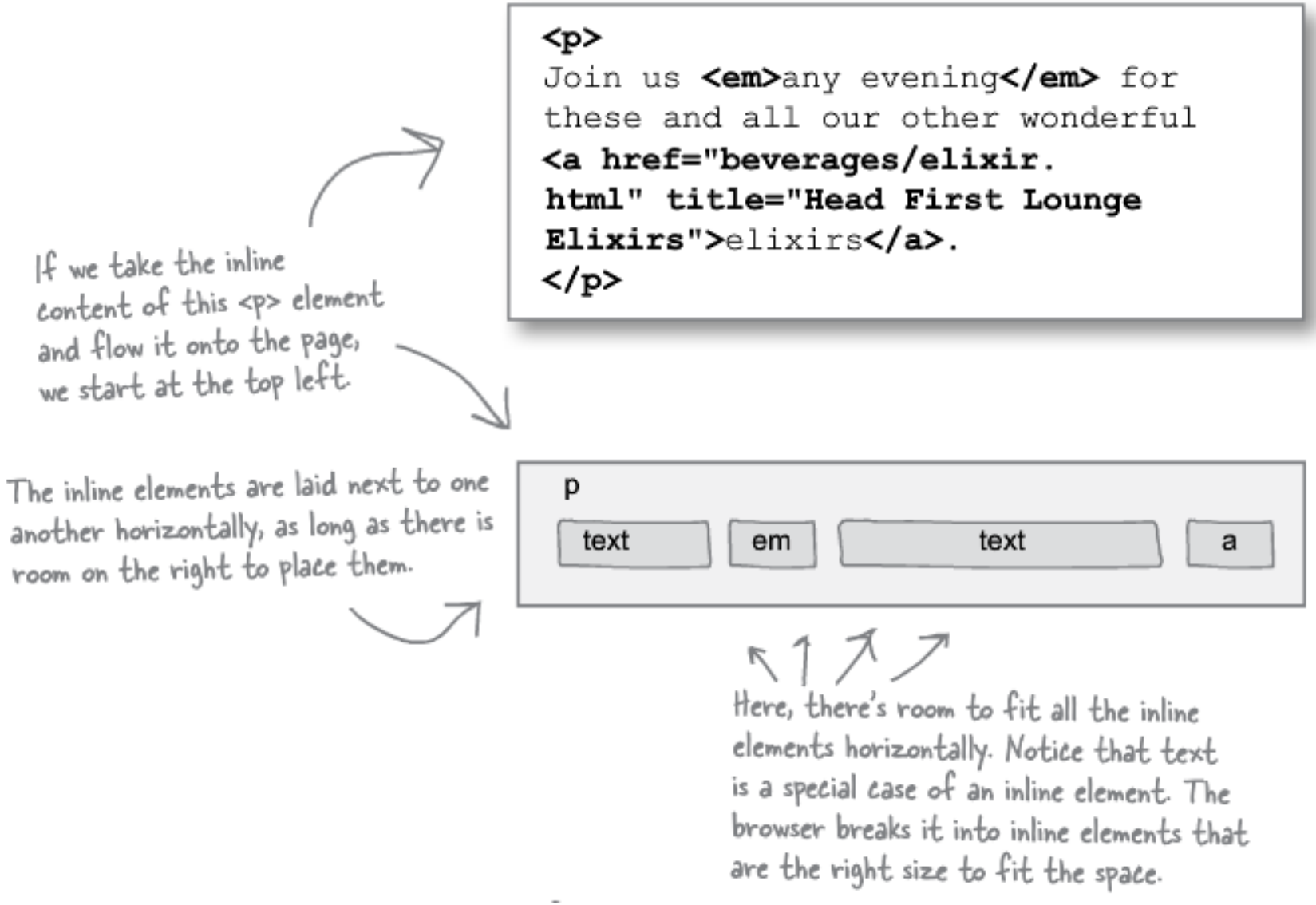


Block flow exercise

- Look at the lounge.html file and draw the flow of the block elements (`h1`, `h2`, multiple `p` and `div` elements, `ul`).

Flow (cont.): Laying out inline elements

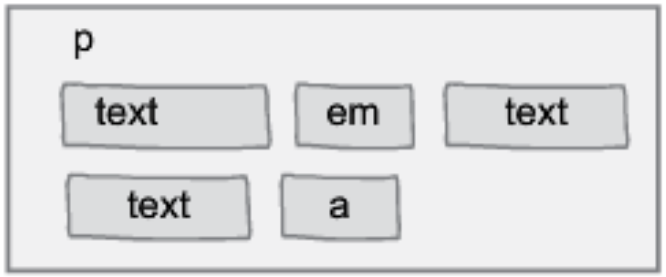
- Within the block elements, inline elements are laid out in the order they appear in the HTML from left to right; top to bottom. When laying out the inline elements, the browser tries to fill the available horizontal space within the block in which they are nested, introducing line breaks only when necessary (or when explicitly instructed with e.g.,
).
- The text is treated as inline elements (with their own boxes).



Flow (cont.): Laying out inline elements

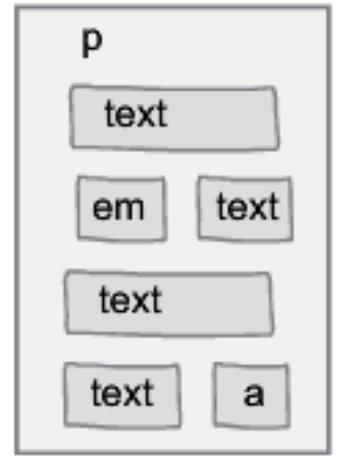
- If the width of the available space of the block element decreases (e.g., because of browser resizing), the inline elements are adjusted accordingly, possibly breaking up text inline elements and making the block taller, as necessary to fit the content.

So what if we make the browser window a little thinner, or we reduce the size of the content area with the width property? Then there's less room to place the inline elements in. Let's see how this works.



Now the content has been flowed left to right until there's no more room, and then the content is placed on the next line. Notice the browser had to break the text up a little differently to make it fit nicely.

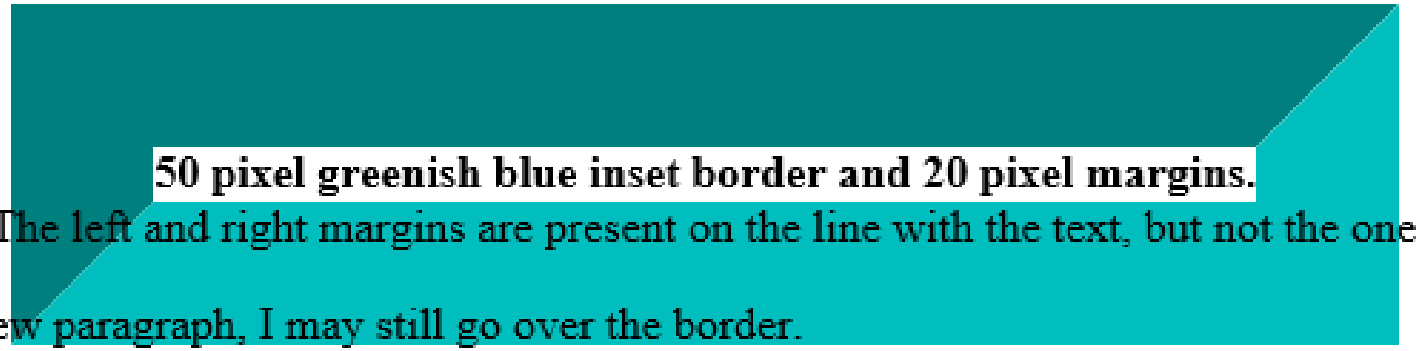
And if we make the content area even thinner, look what happens. The browser uses as many lines as necessary to flow the content into the space.



Padding/Margins of Inline Elements

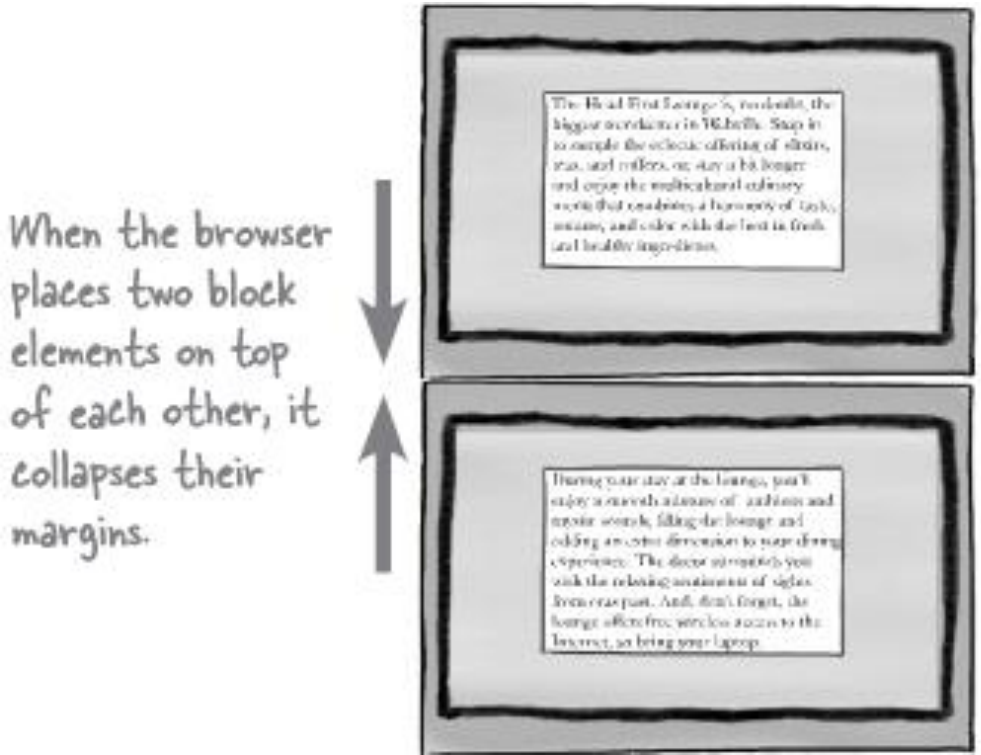
- When the browser lays out inline elements, it fully accounts for left and right margins for content appearing on the same line. However, top and bottom margins of nonreplaced inline elements* (ones in which dimensions are not determined externally, e.g., images) do **not** affect the flow of the other elements around them nor do the left and right margins affect the flow on lines below.

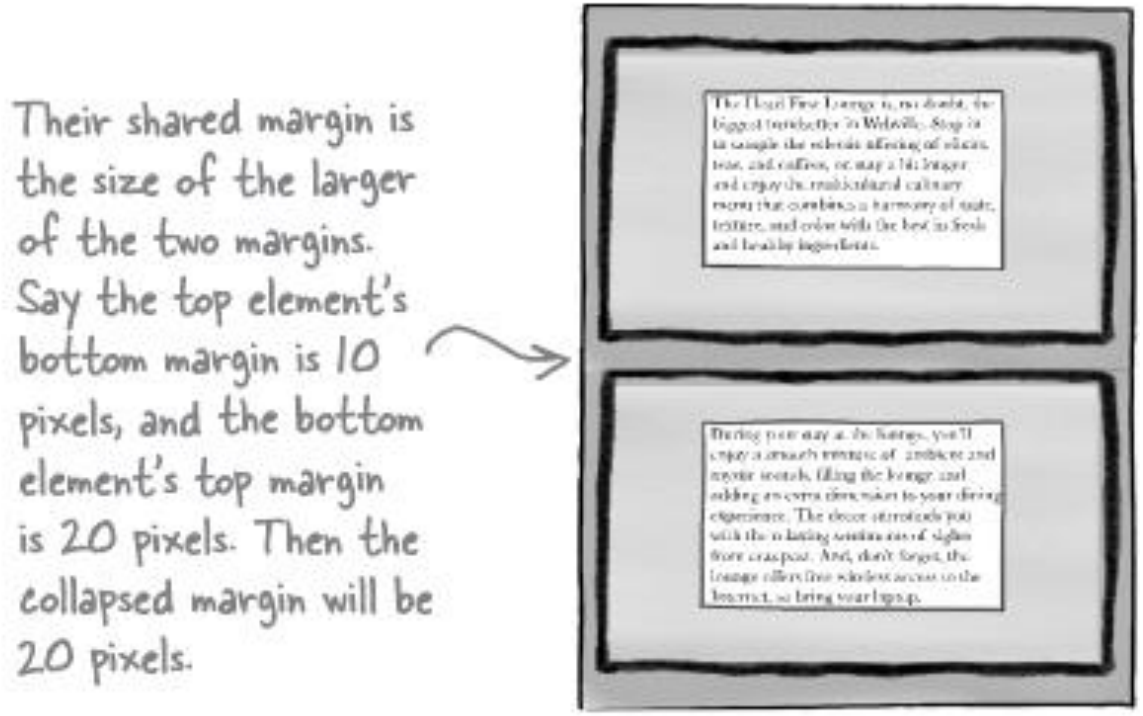
  * http://stackoverflow.com/questions/10324527/margin-top-in-inline-element
  http://reference.sitepoint.com/css/replacedelements



Padding/Margins of Block Elements

- When the vertical margins of two block elements touch, only the larger of the two is used; their margins are **not** summed. This can even happen with a block element which is nested inside another.





Wrapping around block elements

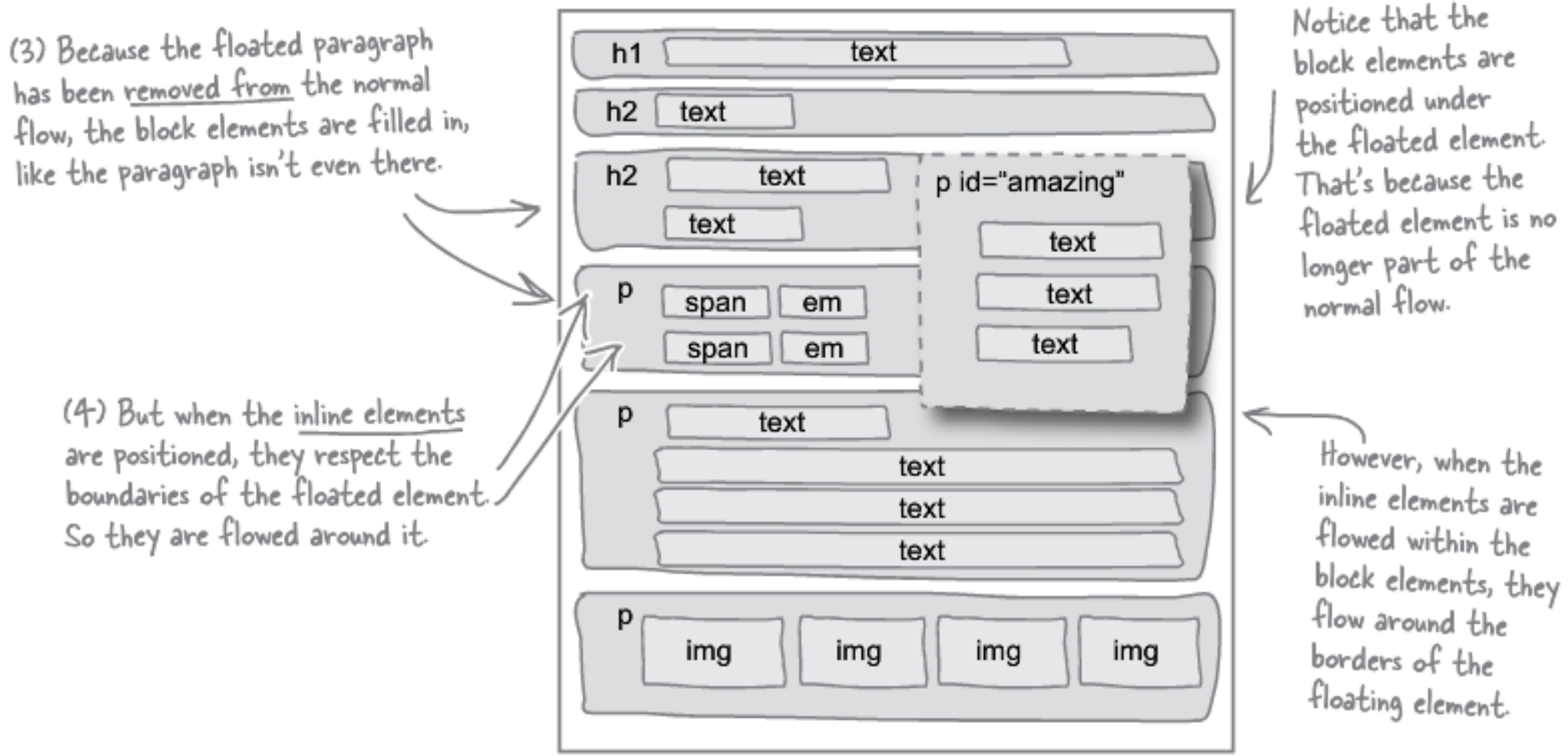
- Unless specified otherwise, blocks appear one on top of the other. Do you think adjusting the width of a block element will make block elements below it move up? Why or why not? Try adding the following `width` property to the lounge.css file to see what happens. (200 pixels is 1 quarter of the typical 800 pixels desktop browser window.)

```
#elixirs {
    ⋮
  width:200px;
    ⋮
}
```

- What rule might you consider adding to make the Elixirs section appear on the right with the blocks below it wrapping around it? ➔

Floating Elements

- When you float a block element to the right, it aligns the element on the right in the vertical position it would be normally placed. But it removes the element from the block flow so that other block elements below it move up as if it's not present. However, inline elements within these blocks are placed as to not encroach upon the boundaries of the floated element.

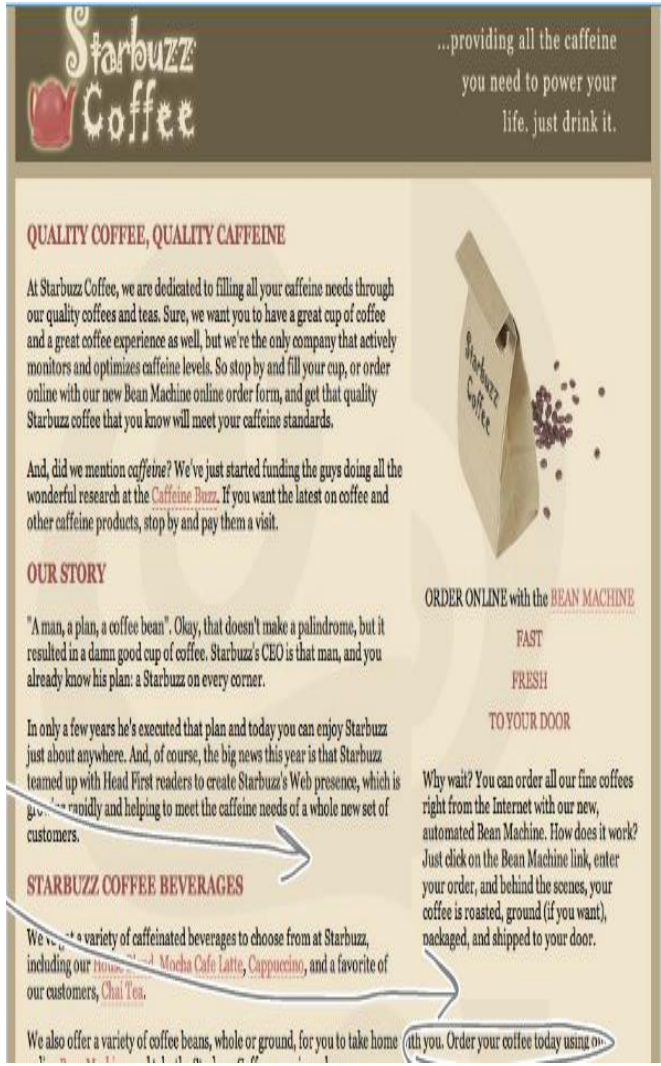


Floating Exercise I

- Reposition the elixirs section in the HTML and add `width:200px;` and `float:right;` to the elixirs rule in the CSS so that the top of the Elixirs section floats on the right with its top lining up with the “Welcome to the Head First Lounge” text.

Starbuzz Exercise: Sidebar I

- Save the source of: http://wickedlysmart.com/hfhtmlcss/chapter11/starbuzz/index.html and get the CSS file from: http://wickedlysmart.com/hfhtmlcss/chapter11/starbuzz/starbuzz.css and get save the following in an images directory placed in the same folders as the files: http://wickedlysmart.com/hfhtmlcss/chapter11/starbuzz/images/bag.gif http://wickedlysmart.com/hfhtmlcss/chapter11/starbuzz/images/header.gif
- Modify the HTML and CSS as necessary to make the sidebar have a width of 280 pixels and float on the right below the header `div`. Once you get this to work, adjust the browser window so that the text wraps underneath the right sidebar. To get the two-column look back, adjust the main section's padding (i.e., Add padding on the right to account for the TOTAL width of the sidebar, padding and margin included.) The space between the columns is called the gutter.
- **Problem with this technique**: Main content comes *after* the sidebar content in the HTML. Structurally, this does not seem correct because the main content is more important. More importantly, we often only want one column on mobile devices with small screens. In this one-column scenario, the sidebar will appear before the main content.



Fixing one last problem: Clear property

- Since we didn't adjust the margin of the footer, the sidebar floats over it. To fix this (and allow the footer to extend underneath), we can add `clear: right;` to the footer rule, which requests that no content on the right float over it (other possible values are `left` and `both`). To do this, the footer will move down as necessary.
- **Problem**: If main column is shorter than sidebar, there will be a gap between the main column and the footer.

Exercise: Sidebar Exercise Attempt II

- Instead of making the sidebar float to the right, have the main content float to the left. Make the width of the main content bar 420 pixels. Do this by mirroring the previous approach. The sidebar plays the role of the main content (e.g., no width) while the main content takes the role of the sidebar.
- **Tradeoff**: Often, no approach will be perfect. Whereas this approach is better in terms of information (main content will be above sidebar content in the HTML), it is worse for design (sidebar expands in width as we make the browser window wider). Ideas for other approaches?

Liquid and Frozen Designs

- Designs so far expanded to fill the browser width, making them **liquid layouts**.
  - **Potential Problem**: Liquid layouts are more difficult to design well and manage.
- Ones in which the layout remains unchanged with browser resizing is a **frozen layout**.
  - **Potential Problem**: Frozen layouts don't make the best use of the available space, leaving behind empty areas devoid of content. If made too wide, they can cause the undesirable horizontal scrolling requirement. Also, frozen layouts are generally poor choices for mobile.
- Designs in between liquid and frozen layouts are **jello layouts**. This takes a frozen layout but makes it look better by centering it, distributing the empty space.
  - **Potential Problem**: Jello layouts still suffer from the problems of frozen layouts.

Exercise: The two-column Frozen Layout

- Enclose all of the `body` element content in between `<div id="allcontent">` and `</div>`. Then include the following CSS:

Now we're going to use this to constrain the size of all the elements and content in the "allcontent" to a fixed width of 800 pixels. Here's the CSS rule to do that:

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

We're going to set the width of "allcontent" to 800 pixels. This will have the effect of constraining everything in it to fit within 800 pixels.

While we're at it, since this is the first time we're styling this , let's add some padding and give it its own background color. You'll see this helps to tie the whole page together.

The outer "allcontent" is always 800 pixels, even when the browser is resized, so we've effectively frozen the to the page, along with everything inside it.

Exercise: Morphing to the Jello Layout

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

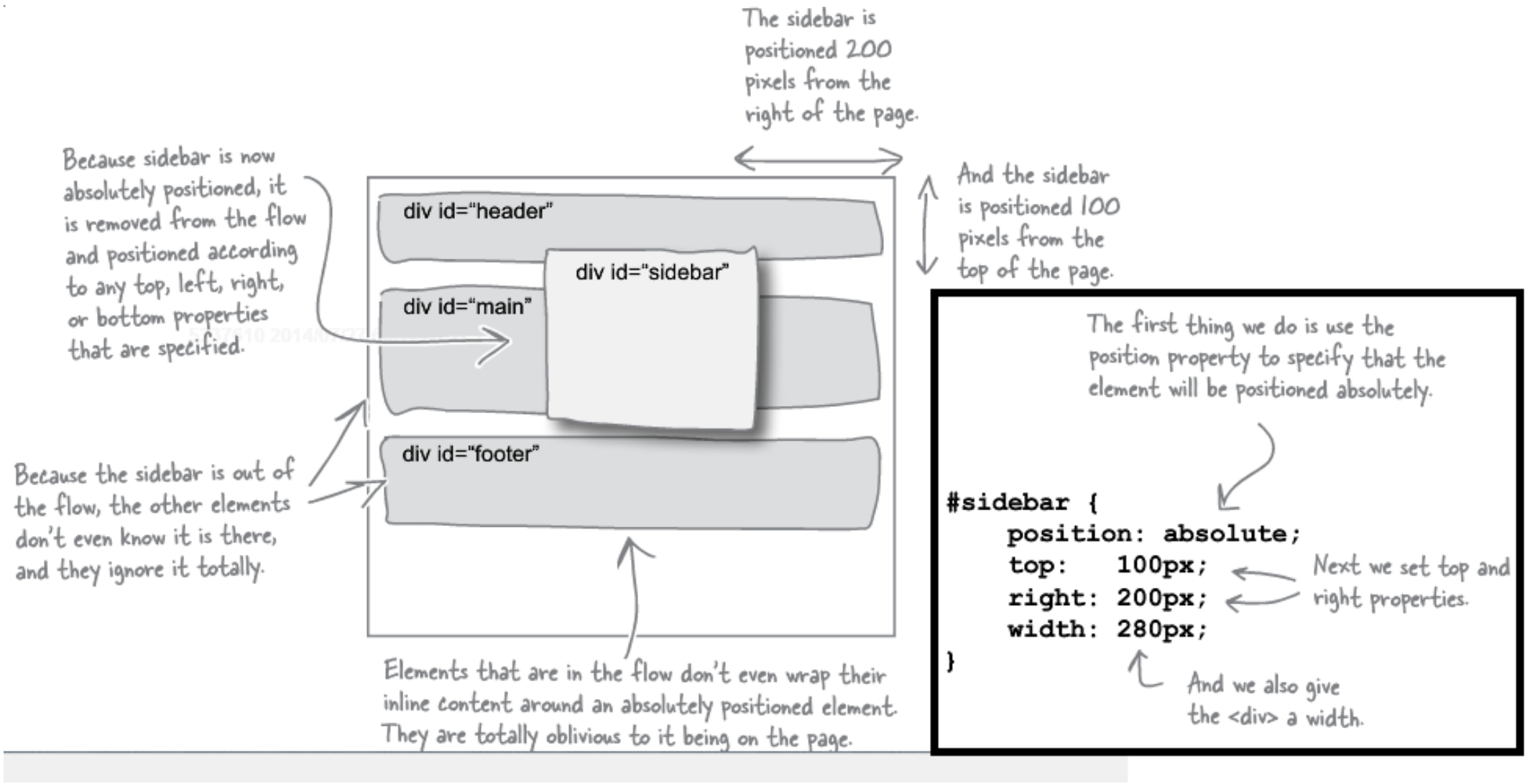
Rather than having fixed left and right margins on the "allcontent" , we're setting the margins to "auto".

If you remember, when we talked about giving a content area a width of "auto", the browser expanded the content area as much as it needed to. With "auto" margins, the browser figures out what the correct margins are, but also makes sure the left and right margins are the same, so that the content is centered.

Absolute positioning

- We can also specify an absolute position for an element in which case it is completely removed from the flow and all other elements, block or inline, are positioned without regard for this element. In this case, text and other content may be hidden underneath absolutely positioned elements.
- Absolute positioning can be specified by setting the `position` property to `absolute` and then specifying a values for `top`, `bottom`, `left`, or `right` in either pixels or percentages to specify the distance away from the top, bottom, left, and right borders of the browser window **WHEN it's showing the top left corner of the web page**, respectively.
  - Example: `top: 20px; right: 10%;` positions the element 20 pixels down from the top and 10% of the width of the browser window to the left of the right border so e.g., 80 pixels if the browser is 800 pixels wide.
- The default value for the position property is `static`, which gives the browser the decision of exact element placement.
- Absolutely positioned elements can be layered via the `z-index` attribute; if one element has a higher z-index value than another one, then it is placed on top.

Absolute Positioning Example

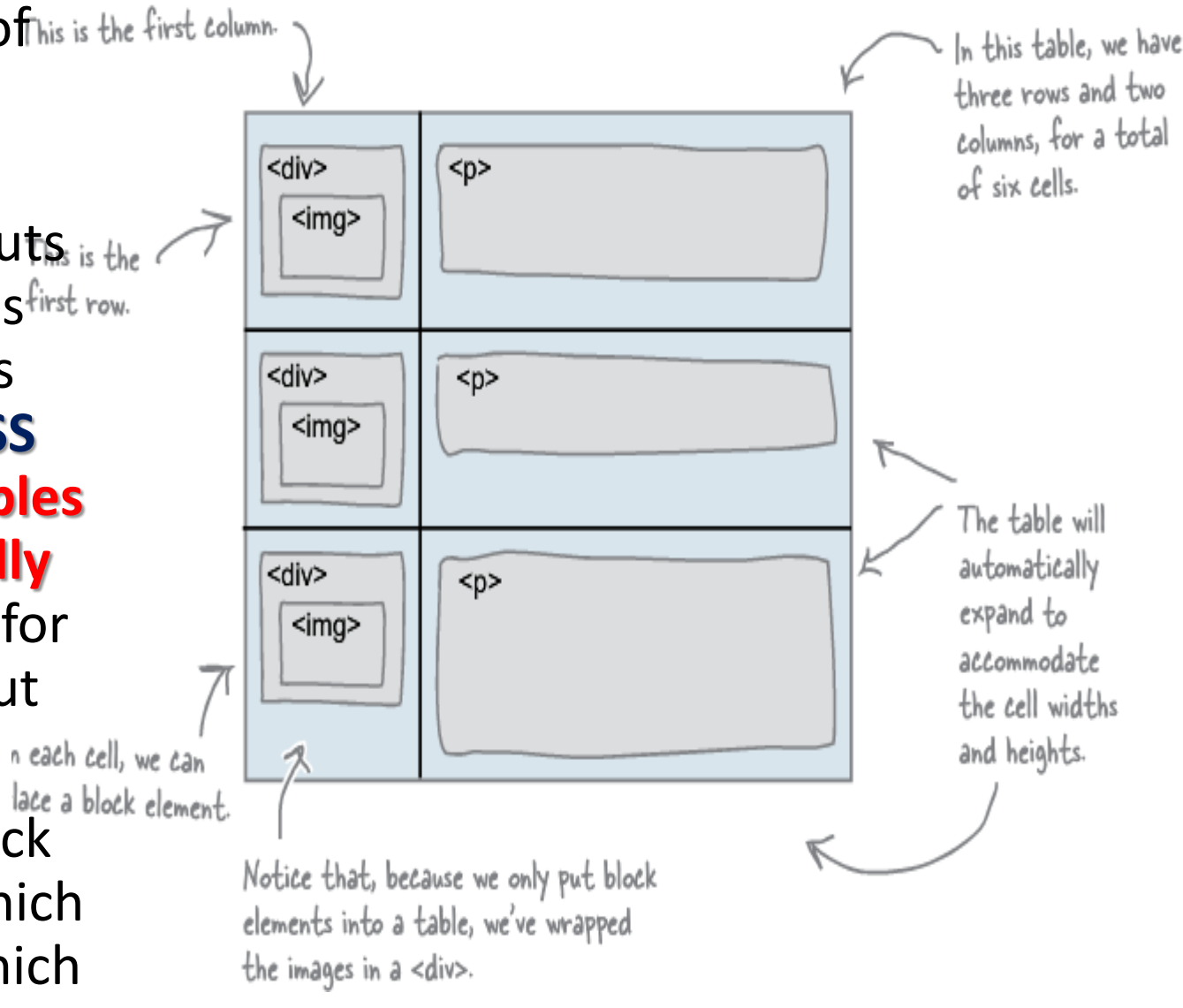


Absolute Positioning Sidebar Exercise

- Set the sidebar to have an absolute position of 128 pixels below the top of the page and 0 pixels from its right (touching the right side). Then specify its width to be 280 pixels. For the main content area, set the right margin to be 330 pixels to account for (width + 2×cellpadding + 2×margin = 280 + 2×15 + 2×10 = 280+30+20)
  - Q: Where did the 128 pixels come from? See if you can visualize how the page will be laid out before viewing.
- **Problem**: When the width is small, the right column may go on top of the footer and the clear property won't fix this since the absolutely positioned sidebar is completely ignored by other elements being placed on the page.

CSS Table Display Layout

- Tables are like spreadsheets, a grid of cells which may or may not be surrounded by borders.
- In the mid 1990's, multicolumn layouts or layouts in which you wanted items to line up were done with borderless HTML tables. However, now with **CSS tables** available for layout, **HTML tables** should only be used when **structurally** the content is tabular in nature, not for e.g., a 2-column layout which is about **presentation**.
- A CSS table is created by nesting block elements; one has a table display which contains blocks with row displays which contain blocks with cell displays.



CSS Table Two-column Exercise Part I

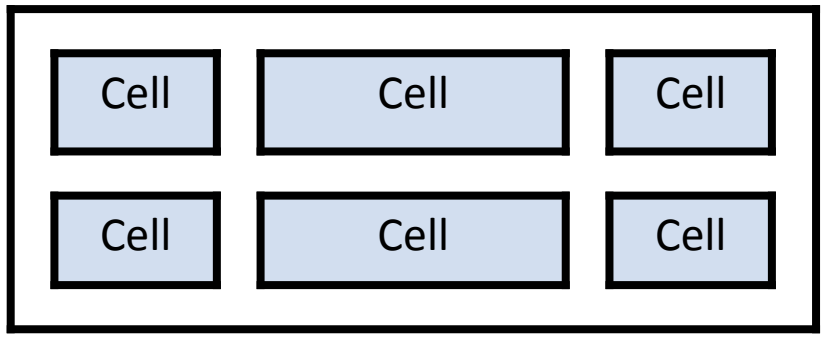
I. Sketch the setup for creating a table for the two-column layout.

II. Introduce new `div` elements as necessary to represent the `table` and single `row` with appropriately named `id` attributes into HTML code. Do this so that the elements are nested properly for implementing the table setup.

III. Add new CSS rules for the elements representing the table and the table row in which you specify `display:table;` and `display:table-row;` , respectively. Add `display:table-cell;` to the rules selecting the `div` elements serving as the cells.

Border-spacing and Vertical-align Properties

- **Border-spacing**: In a block element styled with a table display, the value of this attribute specifies the spacing between borders (effectively acting as a combined margin attribute for the table's block elements representing cells: margin space surrounds borders in box model).



← Border spacing in white

- **Vertical-align**: In a block element styled with a table-cell display, you can specify top, middle or bottom so that the contents of the cell are aligned at its top, middle, or bottom, respectively.

CSS Table Two-column Exercise Part II

IV. Specify 10 pixels of border spacing in the table rule and remove the margins in the block elements working as table cells. Then set the vertically align the content of the table cells at their tops.

V. Does the spacing look a little off to you at the top and bottom? Try correcting it. **Hint**: Border spacing and margins do **NOT** merge when they touch.

Exercise: 3-column CSS

```
<div id="drinks">
  <h1>BEVERAGES</h1>
    <p>House Blend, $1.49</p>
    <p>Mocha Cafe Latte, $2.35</p>
    <p>Cappuccino, $1.89</p>
    <p>Chai Tea, $1.85</p>
  <h1>ELIXIRS</h1>
    <p>
      We proudly serve elixirs brewed by our friends
      at the Head First Lounge.
    </p>
    <p>Green Tea Cooler, $2.99</p>
    <p>Raspberry Ice Concentration, $2.99</p>
    <p>Blueberry Bliss Elixir, $2.99</p>
    <p>Cranberry Antioxidant Blast, $2.99</p>
    <p>Chai Chiller, $2.99</p>
    <p>Black Brain Brew, $2.99</p>
</div>
```

- Add a column to appear on the left that fills 20% of the width of the browser window. Include 15 pixels of cellpadding, make its content be aligned at the top of the cell, and use a background color with hex code #efe5d0. The HTML for this column is on the left.

Summary of Layout Tools

- Floating Layout:
  - Common Use: Placing an image or a column of content and having the other content wrap around it.
  - Possible Disadvantage: Content may need to be rearranged in the HTML in an order that is not reflective of the relative importance (e.g., sidebar's content placed before main content)
- Frozen/Jello Layout
  - Common Use: For pages primarily viewed on more or less fixed-sized screens (e.g., only desktop computers); Good for having exact control over your layout.
  - Possible Disadvantage: Does not make good use of available space; bad when your users view the page on many different screen sizes (e.g., mobile and desktop)
- Absolute Positioning Layout:
  - Common Use: When you want to have exact control over one or more parts of your page, but want a liquid layout to make good use of the available space.
  - Possible Disadvantage: To design the page to your liking, you may need to accept an undesirable overlapping of content for certain browser window sizes.
- Table Display Layout:
  - Common Use: For grid-like layouts; to tile your page layout into columns (and/or) rows.
  - Possible Disadvantage: More complex HTML (adding `divs`) that's harder to modify in response to needed changes in layout, particularly when you want to have a cell in a column that's wider than ones above/below.

Header Does Not Expand and Contract

When the browser window is more than 800px wide, you get all this extra space over here to the right.

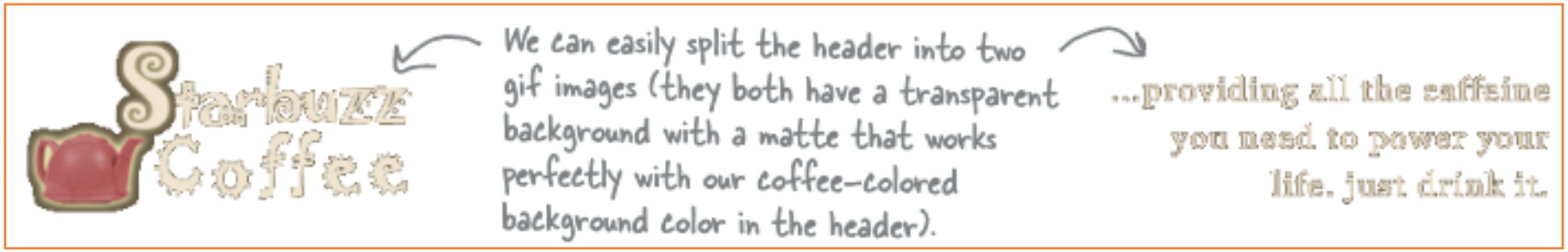
And when the browser is narrower than 800px wide, the slogan part of the header image falls off the edge of the browser window!

The rest of the page resizes nicely as you widen and narrow the browser window.

Exercise: Fixing the Header

- Use the following HTML for the header and modify the CSS (and header HTML if necessary) to have the slogan stay glued at a fixed width to the right of the browser window.

```
<div id="header">
  <!--<img src="images/header.gif" alt="Starbuzz Coffee header image">-->
  <img id="headerLogo"
       src="images/headerLogo.gif" alt="Starbuzz Coffee logo image">
  <img id="headerSlogan"
       src="images/headerSlogan.gif"
       alt="Providing all the caffeine you need to power your life.">
</div>
```



Exercise: Positioning an Award Top and Center

- Add the image at the URL http://www.starbuzzcoffee.com/images/award.gif to the page at the position shown to the right (30 pixels from the top and 365 pixels from the left, overlapping the `header` and `main` `div`s) by using HTML and CSS. Give the image an alt of, "Roaster of the Year award".

Fixed Positioning

- You can fix an element in a certain position in the user's viewport (part of the web page that's visible in the browser window) by setting the `position` property to `fixed` and top, left, right, or bottom accordingly in the rule selecting the element. You can even use ***negative values*** to make the element partially run off the page by that amount (e.g., `right:-20px;` pushes the element 20 pixels out of the right side of the viewport.)
- Unlike absolute positioning which positions the element with respect to the browser window's borders ***when displaying the top left corner of the web page***, fixed positioning positions the element with respect to the borders **of whichever part of the page is in the user's viewport at the moment**, making it appear "stuck" or "fixed" at that position on the page.

Exercise: Fixing the ticket

I. Add the image at the URL http://www.starbuzzcoffee.com/images/ticket.gif to the page and have it link to http://wickedlysmart.com/hfhtmlcss/chapter11/starbuzz/freecoffee.html . Give the link a title of, "Click here to get your free coffee!" and the image an alt of "Starbuzz coupon ticket" . Make this HTML the content of a `div` element with an `id` of coupon.

II. Remove any potential borders with the following in the CSS:

```
#coupon a, img {
    border: none;
}
```

III. Give the image the fixed position shown to the right (350 pixels below the top and 90 pixels LEFT of the left of the browser window so it appears to be coming out of the left side of the browser window.

Extras: Relative positioning and NO CSS

- You can also set position to `relative` in which case the element remains in the flow of the page and is moved relative to where it would've be (e.g., left: 50px moves it 50 pixels left of where it would've been). In particular, it does not "fix" the element to the screen as you scroll. See: http://www.w3schools.com/cssref/pr_class_position.asp
- To see how much difference the CSS makes, comment out the place where you link in the sheet in the HTML by surrounding it with <!-- and --> (<!-- and --> enclose HTML comments, which are ignored when rendering the web page, but can be used as notes for yourself or other people looking/using your HTML).

Notes

- This is primarily a summary of Chapter 11 of *Head First HTML and CSS*, 2nd Edition by Elisabeth Robson and Eric Freeman, 2012. It contains images, exercises, and code from the book.