CSS Floats and Layouts

CSS Layouts

- Two primary layout types:
  - Fixed width
  - Liquid/float-based
- Fixed width gives you more control but at the cost of less flexibility/responsiveness for different screens.
- Liquid or float-based layouts give more flexibility but at the cost of complexity.

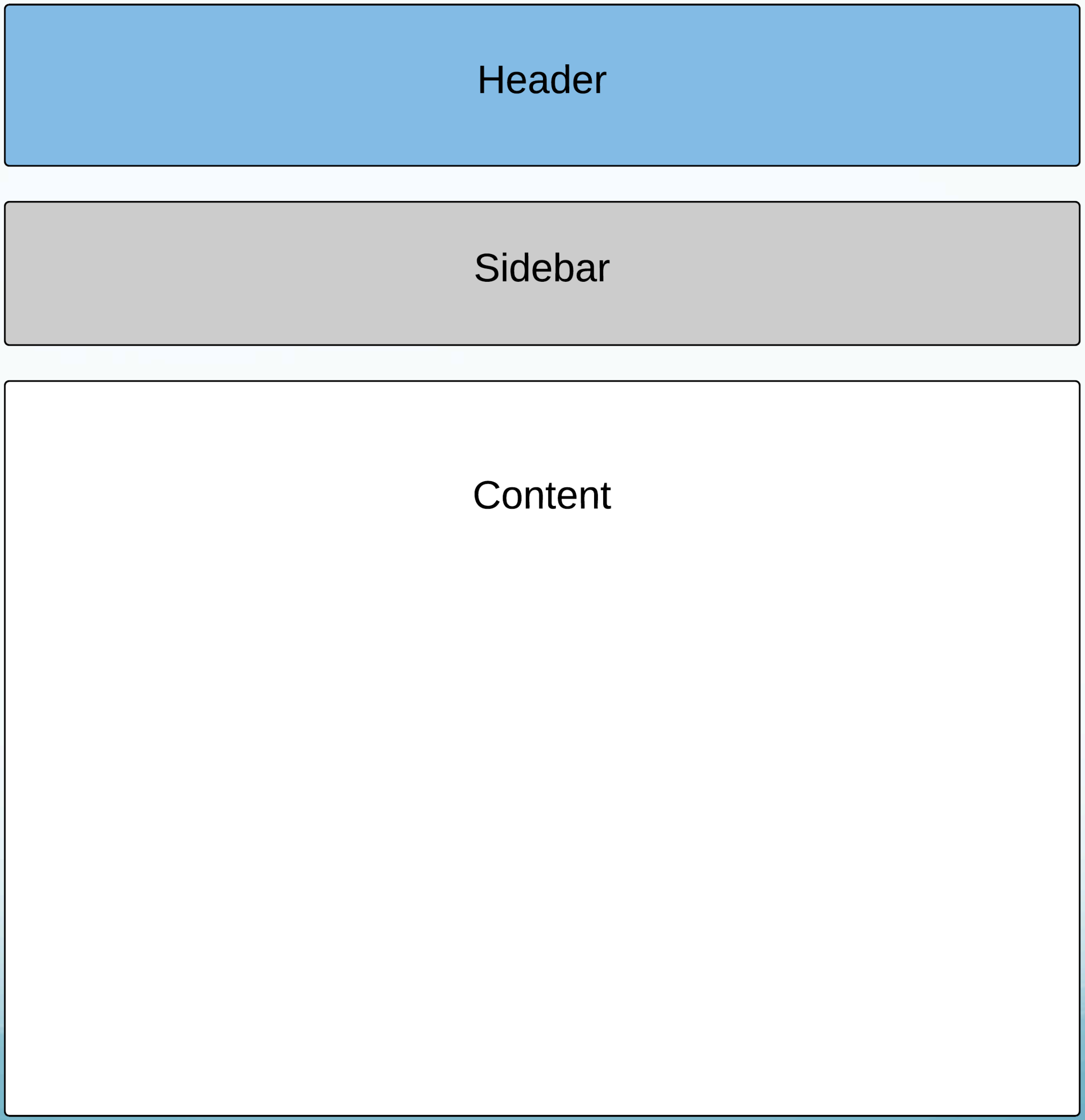
Float-based Layouts

- Recall the **float** property and its values (left, right, none)
- Always give floated elements a width.
- Use floats to make complex layouts such as two and three (or more) column layouts.
- Any percentages given for width are based off of containing element.

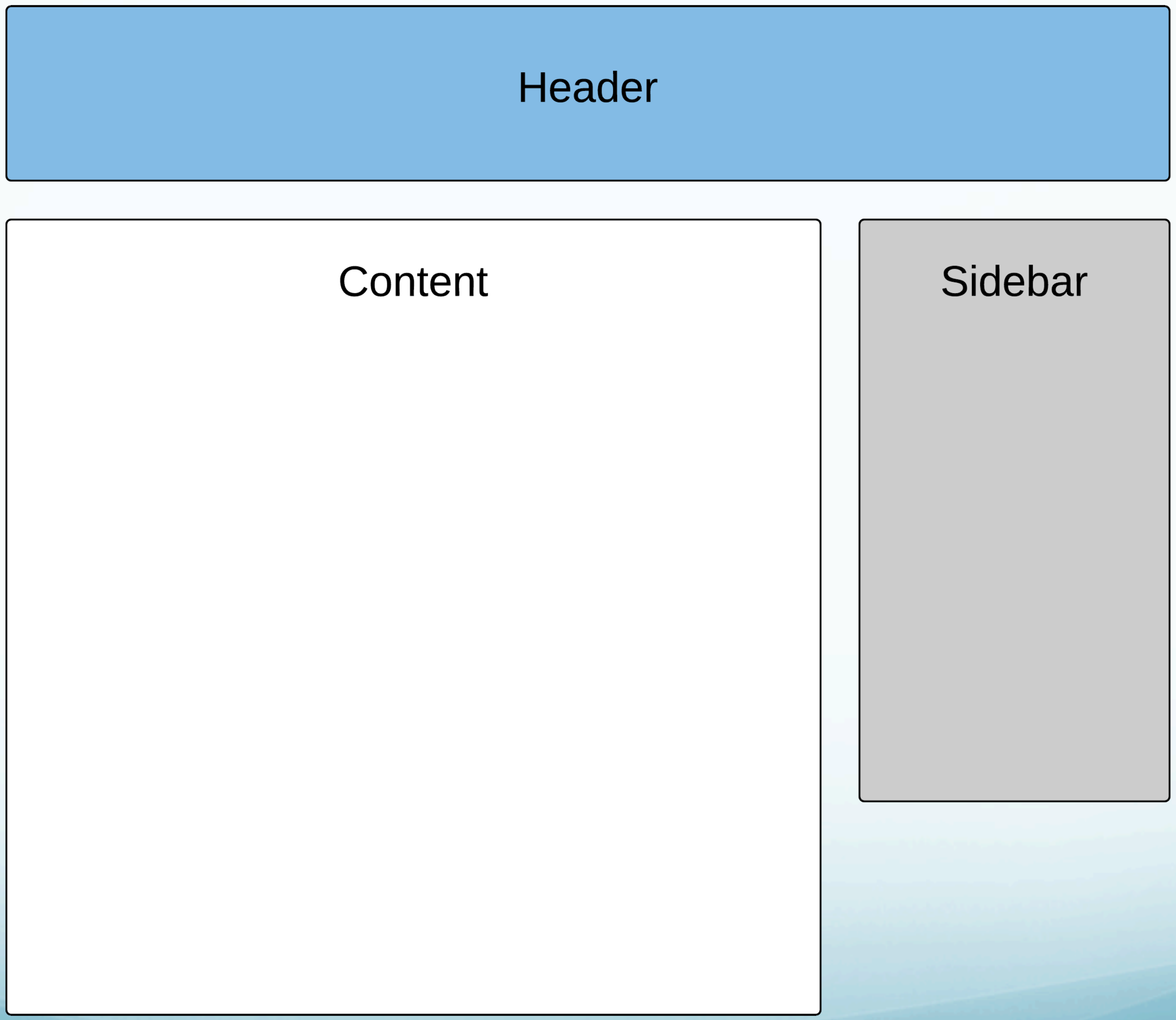
HTML Source Order

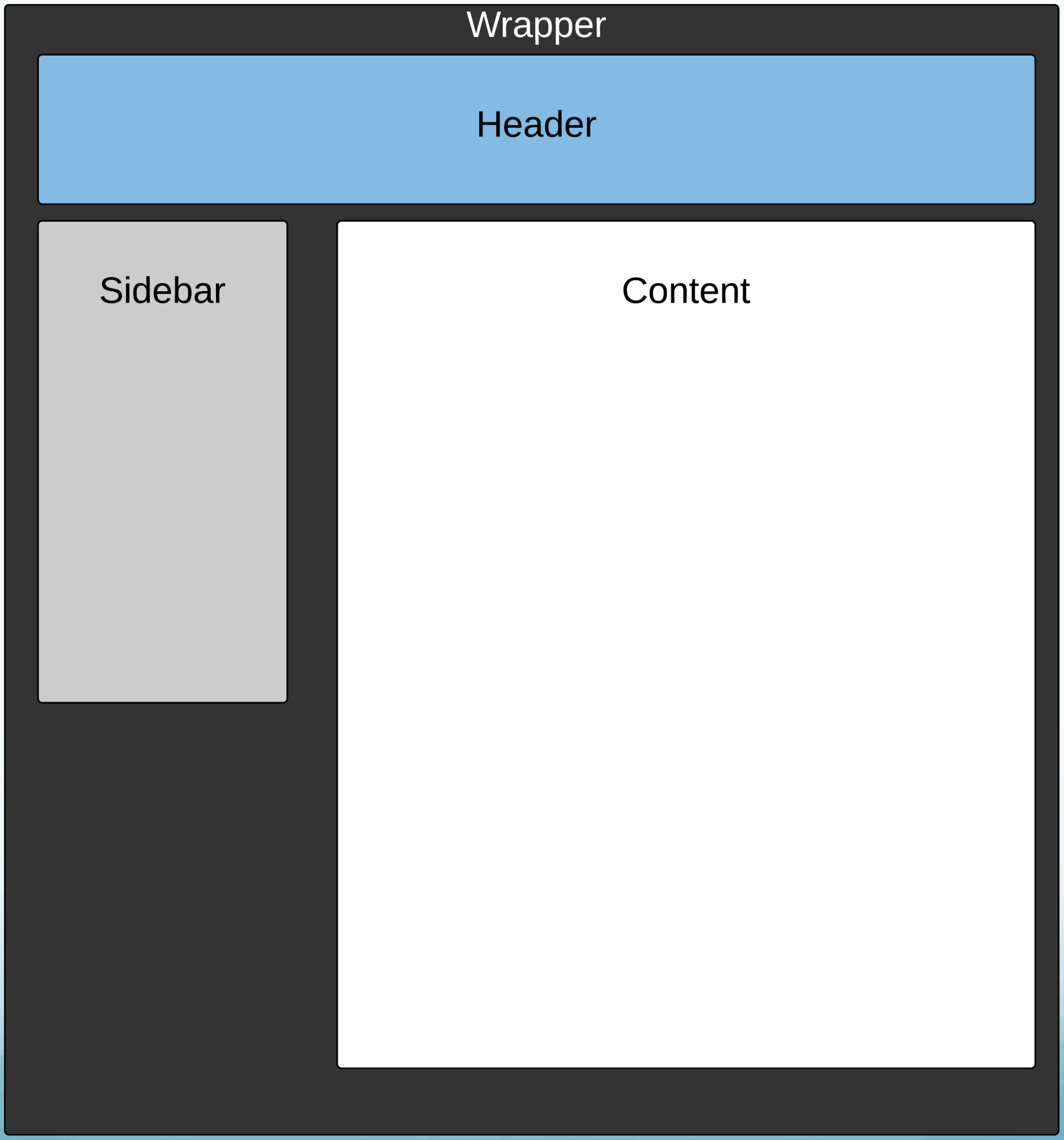
- The order of HTML is important when using floats.
- Things that will float should be placed first in the HTML.

HTML Source Order



CSS Layout (Float Right)




Wrapper
Header
Sidebar
Content

Liquid Two Column Practice

- Create an HTML5 page with the following elements:
  - header
  - aside
  - section
- Create an external style sheet and link it to the page.
- Float the aside – (hint: add borders to see the boxes)
- Set a width for the aside
- Optionally add a margin to one side of the main content area to prevent the underneath wrap.

Two Columns Demo

- Note the left column and the HTML structure
- Notice how the Copyright symbol is connected to the main content.
- We can fix it.

Floats Practice

- Start with the page you used for the previous practice.
- Add some content to your aside to make it longer than the content section.
- Add a copyright notice to a new footer section.
- View in browser

Clearing Floats

- The **clear** property stops the wrapping behavior around floated elements.
- To clear the float from one side use left or right:
  - clear: left;
  - clear: right;
- To clear the float from both sides use both:
  - clear: both;

Clearing Floats Practice

- Clear the float on your footer.
- View in browser

CSS Positioning

- From floats we move to another way to position things
- The **position** property controls where a browser displays an element on a page.
- Four values:
  - absolute
  - relative
  - fixed
  - static

Positions

- Absolute
  - left, right, top, bottom using percentages, pixels, and ems
  - Absolute positioning detaches the element from the HTML
- Fixed
  - Same rules as the background-attachment (see that for more info)
- Relative
  - In relation to where it would normally appear
- Static
  - Follows normal top-down flow (the default)

Viewports

- A "viewport" describes the browser window, the window through which the HTML is viewed.
- The viewport changes based on the size of the browser window.
- Each edge of the viewport has a CSS property:
  - left
  - right
  - top
  - bottom
- If vertical or horizontal value is not set, browser uses default

Positioning Practice

- Create a new HTML5 page and inside place a single section.
- Give the section a width and height (400px each) along with a background color or border (I used #99CCFF).
  - Note that the width and height along with color/ border are used as helpers so we can see the section easier.
- Give the section an absolute position that's 50px from the top and 100px from the left.

Absolute Positioning is Relative

- An absolute position is relative to its closest positioned ancestor
- Two rules:
  - Position is relative to browser window when an element is absolute positioned and not inside any other tag that's been positioned
  - Position is relative to edges of element when it exists inside of another element that has been positioned.

Relative Absolute Practice

- Add an <article> within the <section> and give it some content along with a border or different background color.
- Position the article absolute with a left of 100px and top of 50px.
- Note how this positioning is relative to the <section> and not the browser window. (Firebug's Layout tab is nice here)

Positioning in 3 Dimensions

- So far we've dealt with horizontal and vertical positioning.
- CSS can also be used to control the third dimension or where elements appear in relation to one another as if viewed in three dimensions.
- The **z-index** property is used to control the layering or stacking order for elements.

z-index Values

- z-index takes a number as its value.
- The larger the number the higher in the stack (closer to the top/front)
- Negative values are ok
- Protip: Use 10's, like 10, 20, 30 for elements. Doing so gives you the ability to add something in-between, like 11 or 12 to appear above 10 but below 20.

Visibility

- Recall the "display: none" CSS rule.
- display: none; causes the element to not be shown on the page.
- There is also a **visibility** property
- Two values:
  - hidden
  - visible

Visibility versus Display

- visibility: hidden; leaves a hole where the content would have been.
- With display: none; the browser never renders the element.
- Also see "opacity: 0;" as a means to hide content.