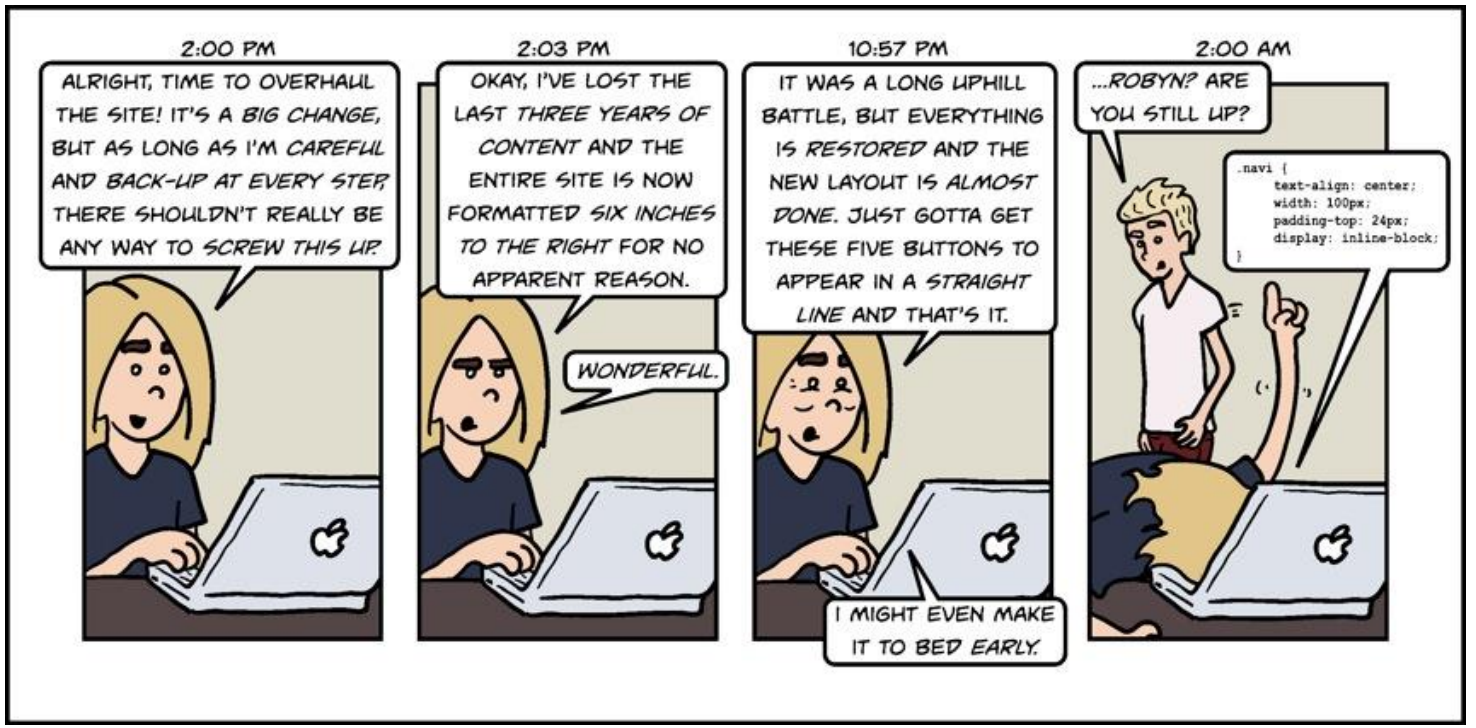
CSc 337



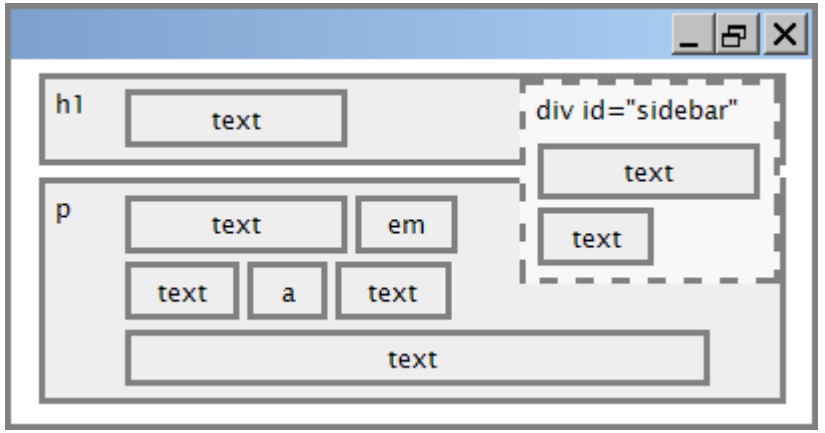
LECTURE 4: POSITIONING

The CSS float property

property	description
float	side to hover on; can be left, right, or none (default)

- a ***floating*** element is removed from normal document flow

- underlying text wraps around it as necessary



Float example

```
<img src="images/koala.jpg" alt="Koala" class="headericon" />

Lorem ipsum dolor sit amet, consectetur adipiscing elit....
```
HTML

```
img.headericon {
    float: left;
}
```
CSS

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam scelerisque purus ut dui mollis, sed malesuada leo pretium. Morbi bibendum mi at lacus rutrum convallis. Duis id eros dolor. In id eros blandit lectus viverra facilisis at commodo velit. Cras pretium nunc id nisl elementum, at interdum odio blandit. Donec luctus rutrum iaculis. Praesent luctus ante et cursus suscipit. Nullam congue egestas lorem nec luctus. Donec tincidunt tortor mi, nec ultricies orci bibendum a. Aliquam viverra metus nec ligula varius feugiat. In lacinia ligula accumsan tortor porttitor ornare. Donec interdum mattis purus sit amet ultrices.

output

Floating content and width

I am not floating, no width set

I am floating right, no width set

I am floating right, no width set, but my text is very long so this paragraph doesn't really seem like it's floating at all, darn

I am not floating, 45% width

I am floating right, 45% width

- often floating elements should have a `width` property value
  - if no `width` is specified, other content may be unable to wrap around the floating element

The clear property

```
p { background-color: fuchsia; }
h2 { clear: right; background-color: cyan; }
```
CSS

XKCD a webcomic of romance, sarcasm, math, and language...

My XKCD Fan Site

property	description
clear	disallows floating elements from overlapping this element; can be left, right, both, or none (default)

Common error: container too short

```
<p><img src="images/xkcd.png" alt="the man in the hat" />
  XKCD a webcomic of romance, sarcasm,
    math, and language...</p>
```
HTML

```
p { border: 2px dashed black; }
img { float: right; }
```
CSS

XKCD a webcomic of romance, sarcasm, math, and language...

- We want the p containing the image to extend downward so that its border encloses the entire image

The overflow property

```
p { border: 2px dashed black; overflow: hidden; }
```
CSS

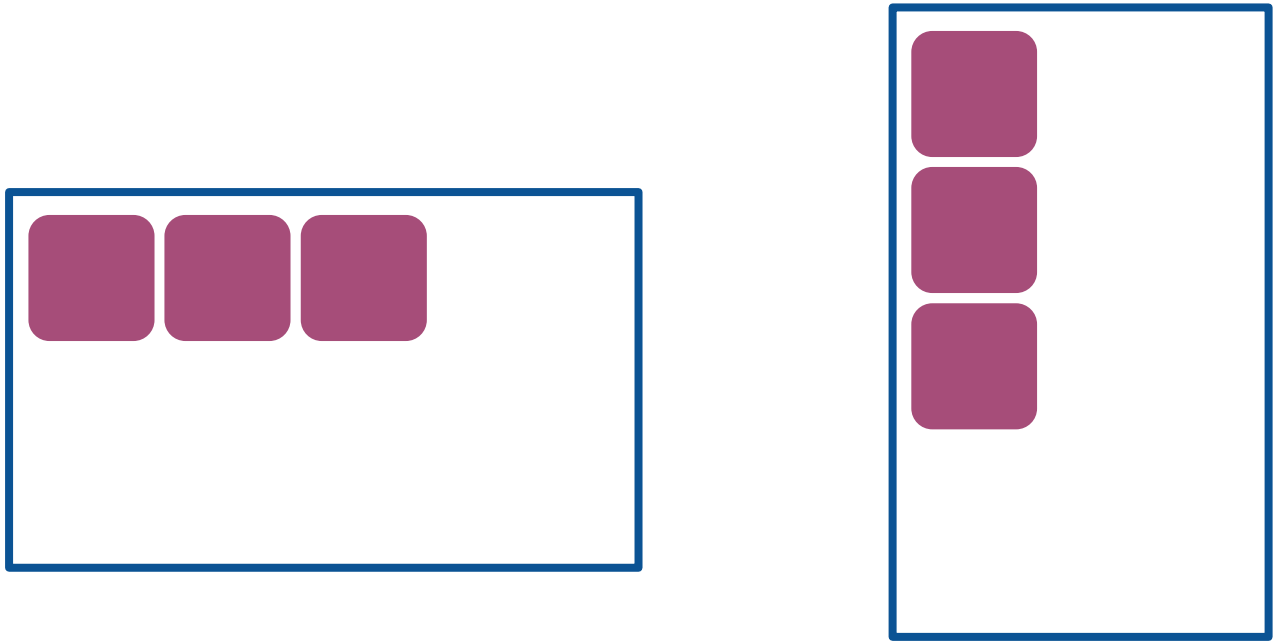
XKCD a webcomic of romance, sarcasm, math, and language...

property	description
overflow	specifies what to do if an element's content is too large; can be auto, visible, hidden, or scroll

Flex layout

To achieve more complicated layouts, we can enable a different kind of CSS layout rendering mode: **Flex layout.**

Flex layout defines a special set of rules for laying out items in rows or columns.

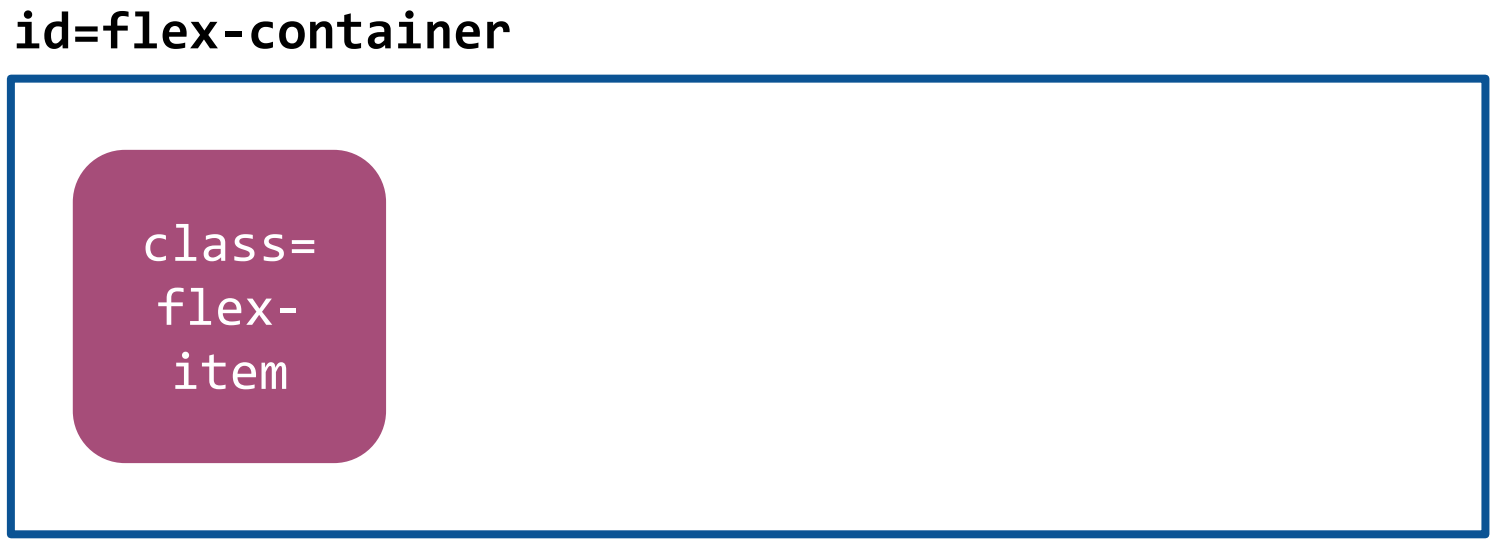


Flex basics

Flex layouts are composed of:

- A **Flex container**, which contains one or more:
  - **Flex item**(s)

You can then apply CSS properties on the **flex container** to dictate how the flex items are displayed.



Flex basics

To make an element a flex container, change `display`:

- Block container: `display: flex;` or
- Inline container: `display: inline-flex;`

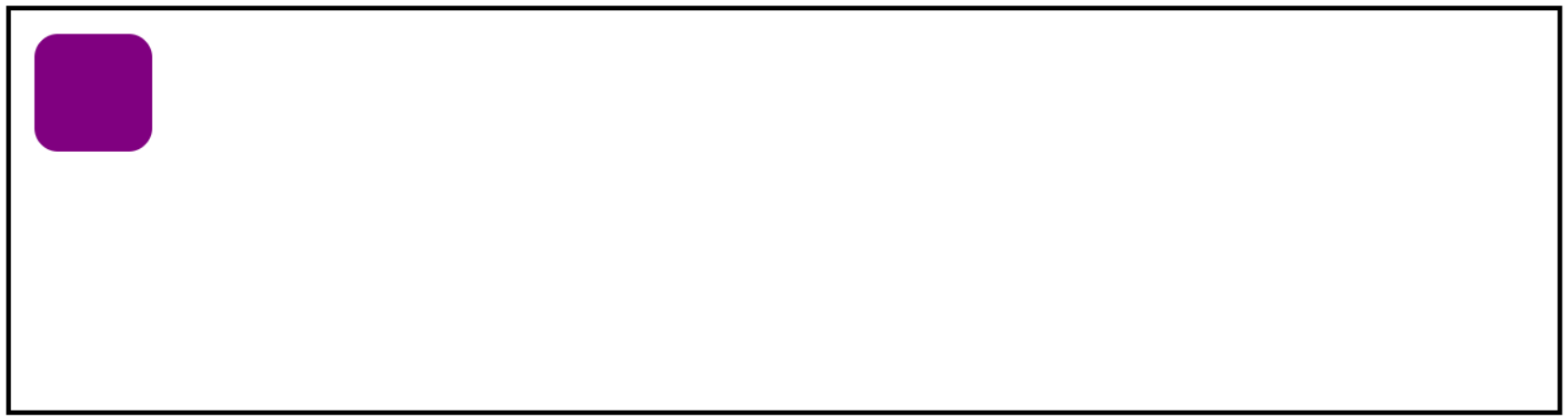
HTML

```html
<html>
  <head>
    <meta charset="utf-8">
    <title>Flexbox example</title>
  </head>
  <body>

    <div id="flex-container">
      <div class="flex-item"></div>
    </div>

  </body>
</html>
```

CSS

```css
#flex-container {
  display: flex;
  border: 2px solid black;
  padding: 10px;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  background-color: purple;
  height: 50px;
  width: 50px;
}
```

JS



Flex basics: `justify-content`

You can control where the item is horizontally* in the box by setting `justify-content` on the flex container:

```
#flex-container {
  display: flex;
  justify-content: flex-start;
}
```

*when flex direction is `row`. We'll get to what "flex direction" means soon.

Flex basics: `justify-content`

You can control where the item is horizontally* in the box by setting `justify-content` on the flex container:

```
#flex-container {
  display: flex;
  justify-content: flex-end;
}
```

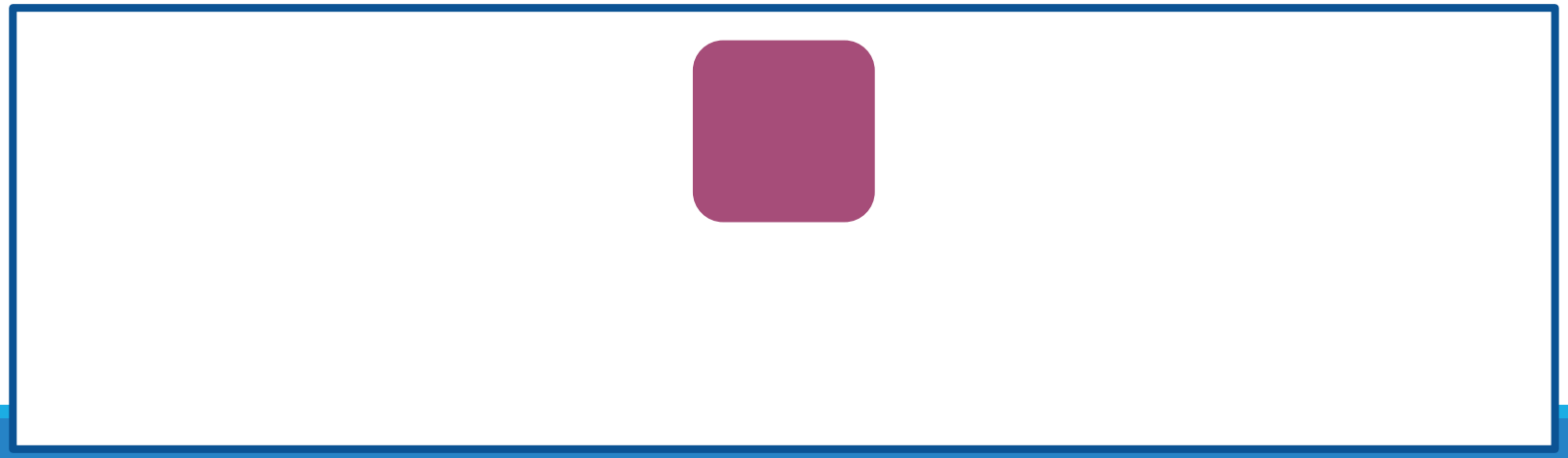
*when flex direction is `row`. We'll get to what "flex direction" means soon.

Flex basics: `justify-content`

You can control where the item is horizontally* in the box by setting `justify-content` on the flex container:

```
#flex-container {
  display: flex;
  justify-content: center;
}
```

*when flex direction is `row`. We'll get to what "flex direction" means soon.

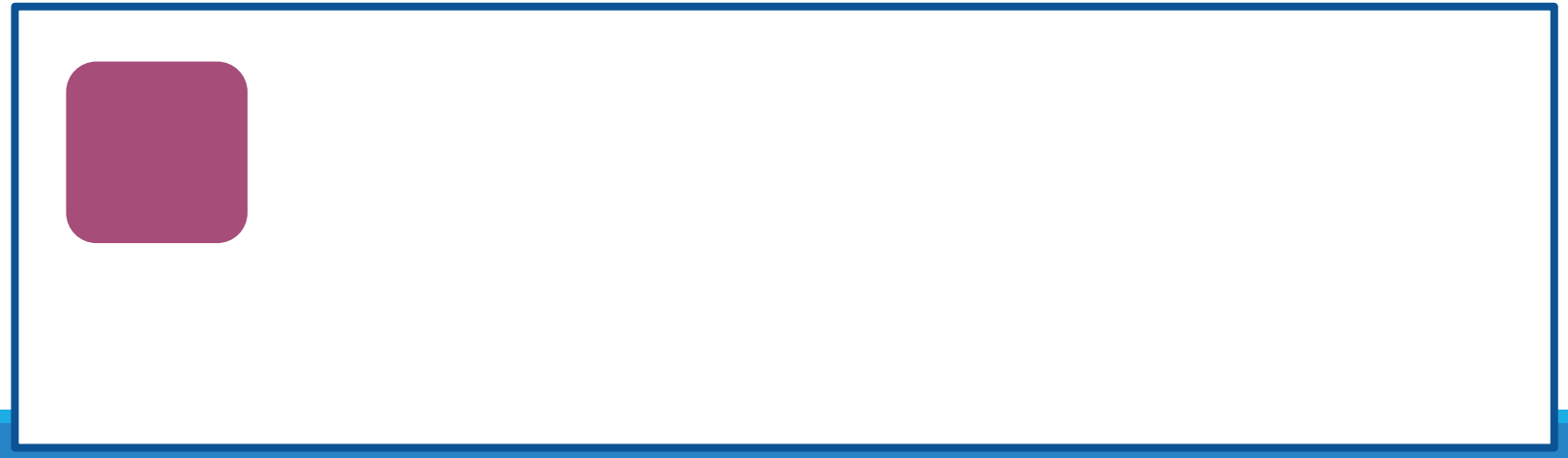


Flex basics: `align-items`

You can control where the item is vertically* in the box by setting `align-items` on the flex container:

```
#flex-container {
  display: flex;
  align-items: flex-start;
}
```

*when flex direction is `row`. We'll get to what "flex direction" means soon.

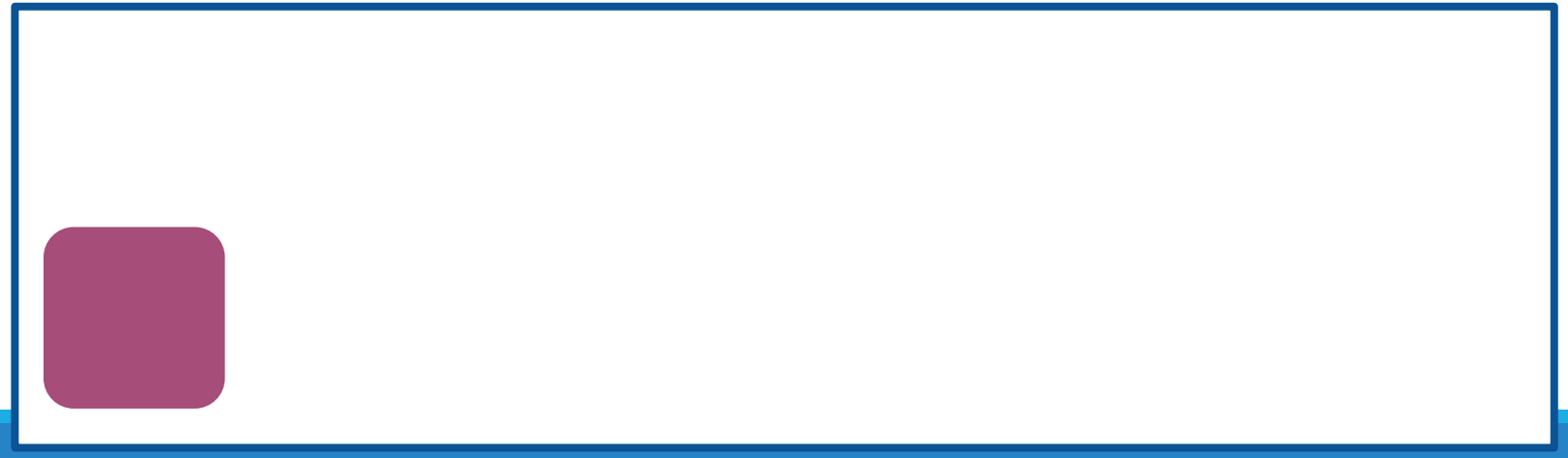


Flex basics: `align-items`

You can control where the item is vertically* in the box by setting `align-items` on the flex container:

```
#flex-container {
  display: flex;
  align-items: flex-end;
}
```

*when flex direction is `row`. We'll get to what "flex direction" means soon.

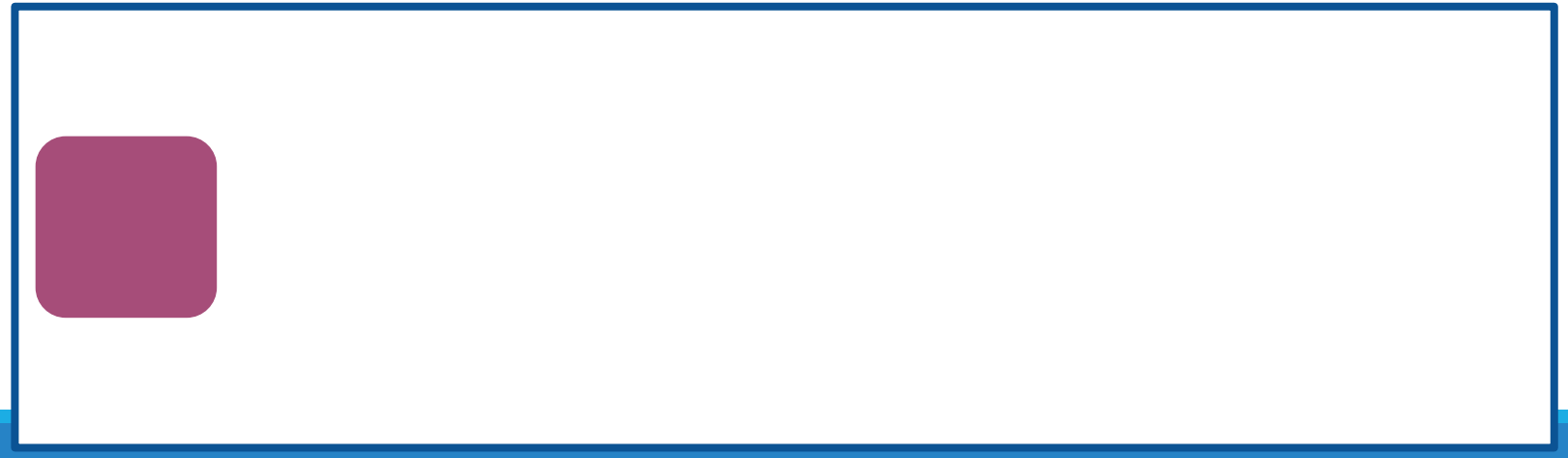


Flex basics: `align-items`

You can control where the item is vertically* in the box by setting `align-items` on the flex container:

```
#flex-container {
  display: flex;
  align-items: center;
}
```

*when flex direction is `row`. We'll get to what "flex direction" means soon.



Multiple items

Same rules apply with multiple flex items:

```
#flex-container {
  display: flex;
  justify-content: flex-start;
  align-items: center;
}
```

Multiple items

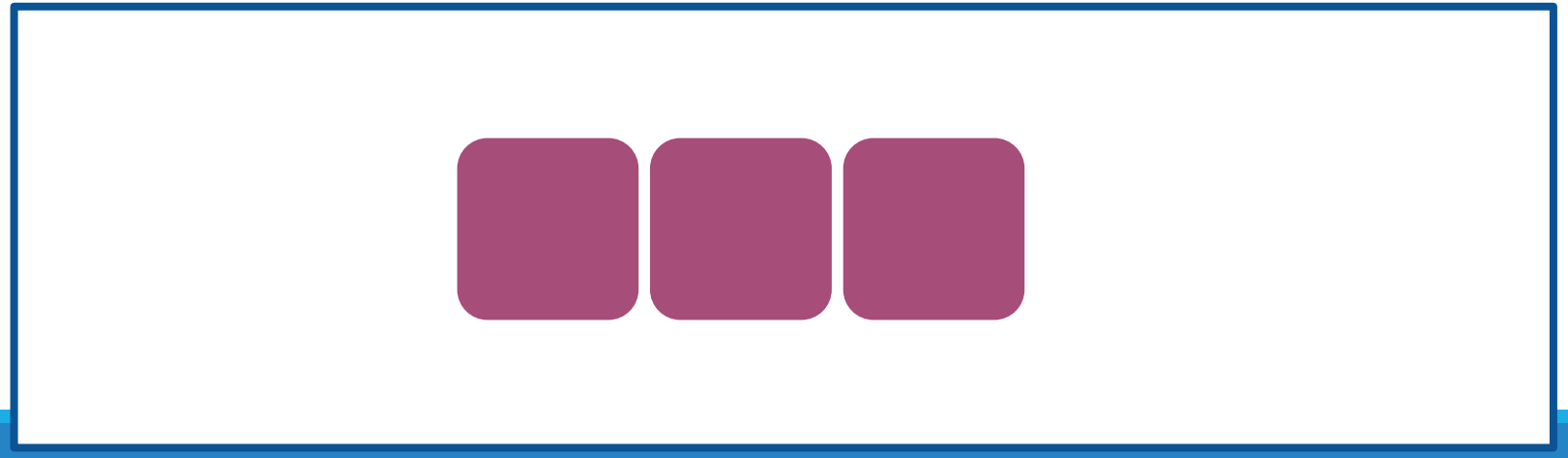
Same rules apply with multiple flex items:

```
#flex-container {
  display: flex;
  justify-content: flex-end;
  align-items: center;
}
```

Multiple items

Same rules apply with multiple flex items:

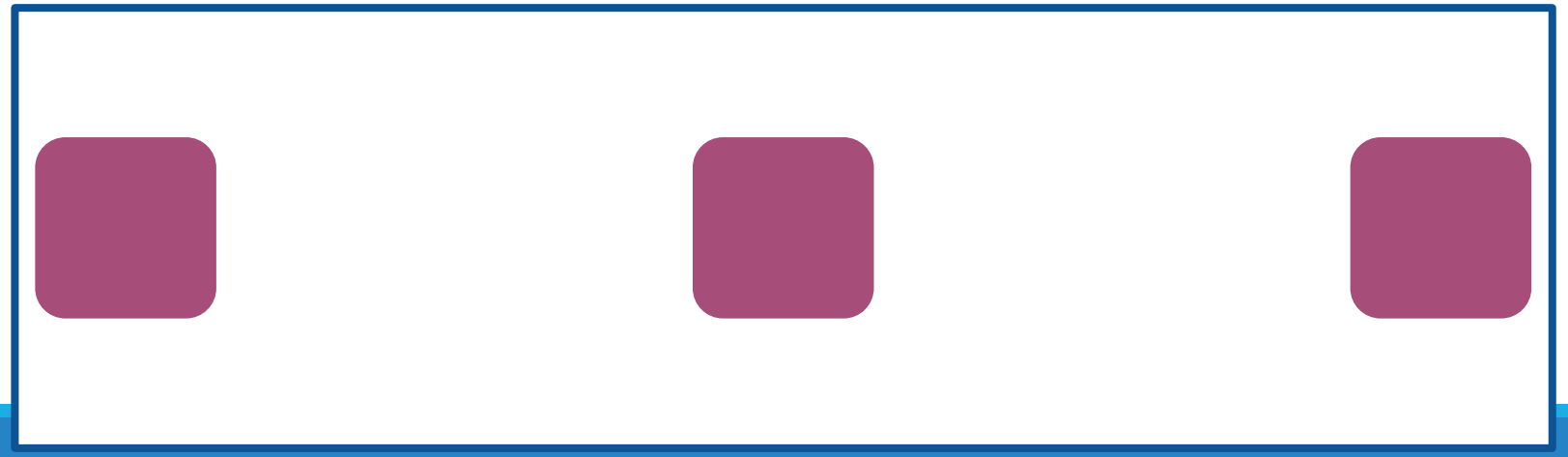
```
#flex-container {
  display: flex;
  justify-content: center;
  align-items: center;
}
```



Multiple items

And there is also **space-between** and **space-around**:

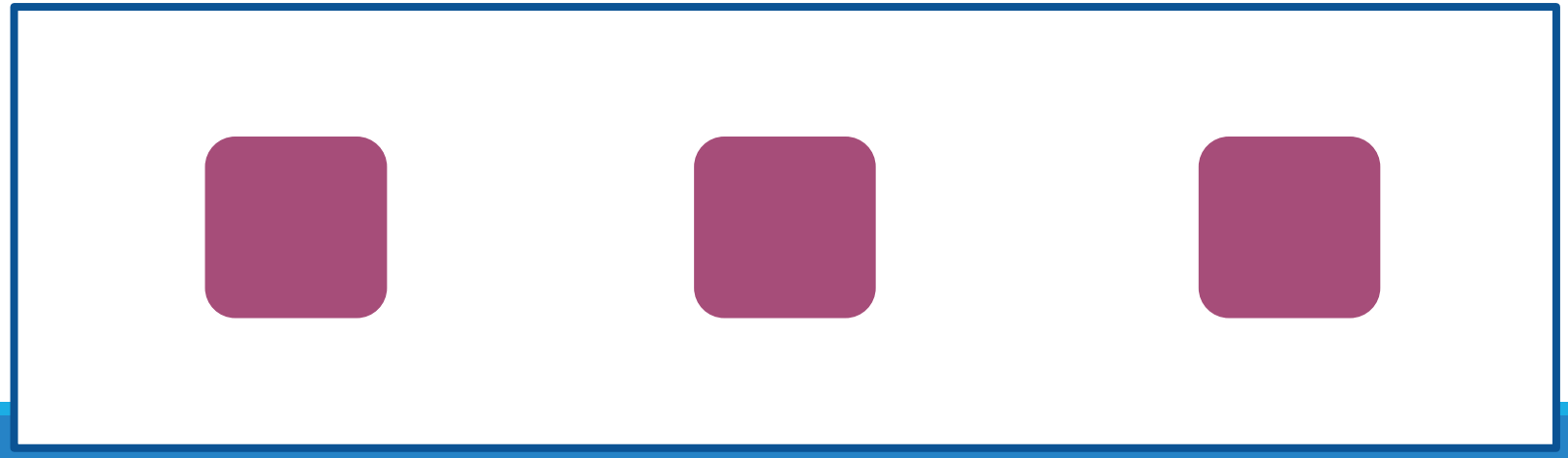
```
#flex-container {
  display: flex;
  Justify-content: space-between;
  align-items: center;
}
```



Multiple items

And there is also **space-between** and **space-around**:

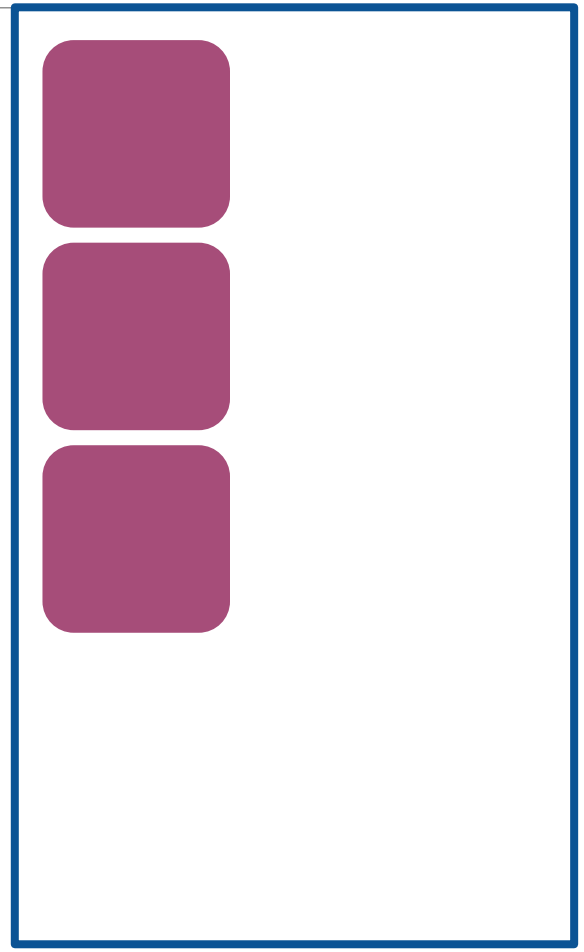
```
#flex-container {
  display: flex;
  Justify-content: space-around;
  align-items: center;
}
```



flex-direction

And you can also lay out columns instead of rows:

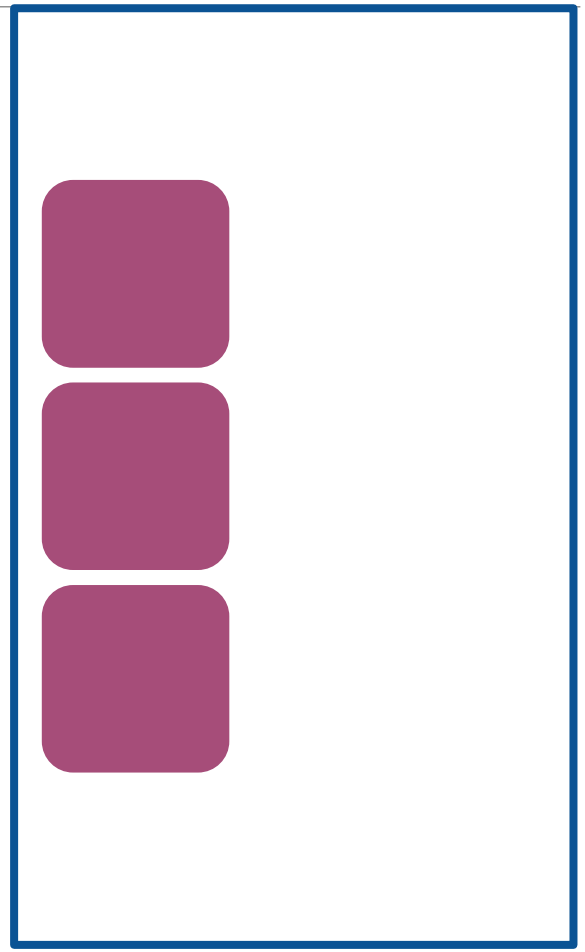
```
#flex-container {
  display: flex;
  flex-direction: column;
}
```



flex-direction

And you can also lay out columns instead of rows:

```
#flex-container {
  display: flex;
  flex-direction: column;
  justify-content: center;
}
```

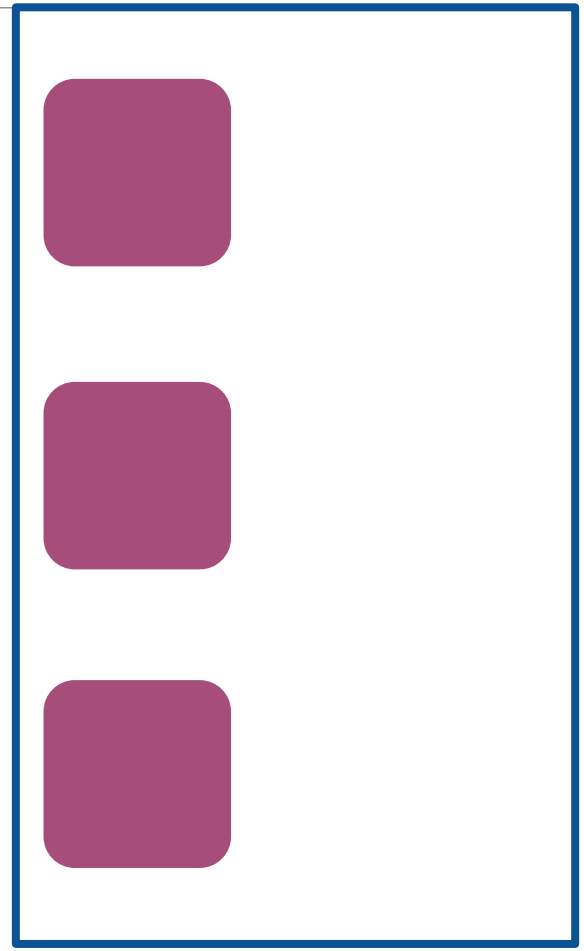


Now **justify-content** controls where the column is vertically in the box

flex-direction

And you can also lay out columns instead of rows:

```
#flex-container {
  display: flex;
  flex-direction: column;
  justify-content: space-around;
}
```

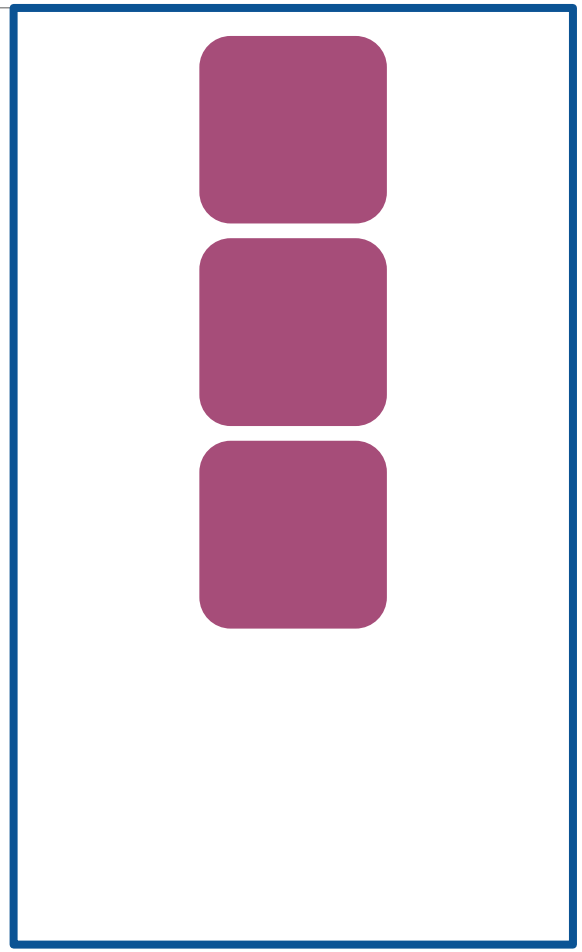


Now **`justify-content`** controls where the column is vertically in the box

flex-direction

And you can also lay out columns instead of rows:

```
#flex-container {
  display: flex;
  flex-direction: column;
  align-items: center;
}
```

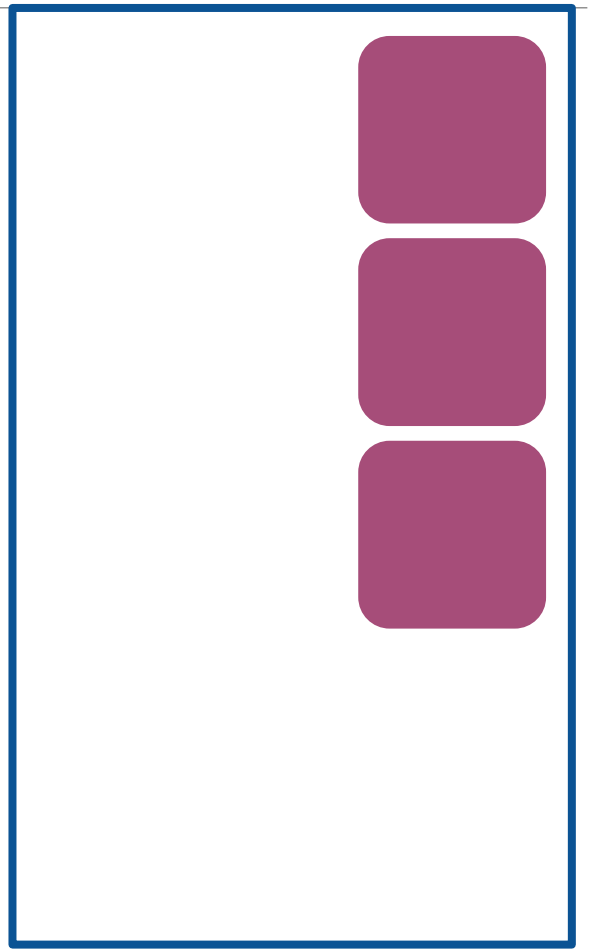


Now **`align-items`** controls where the column is horizontally in the box

flex-direction

And you can also lay out columns instead of rows:

```
#flex-container {
  display: flex;
  flex-direction: column;
  align-items: flex-end;
}
```



Now **`align-items`** controls where the column is horizontally in the box

Activity

- Write code to match the image on the right.
- Starter code available here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent enim enim, convallis a volutpat at, sollicitudin ac ligula. Sed diam urna, tempor at risus sit amet, molestie vestibulum ligula. In hac habitasse platea dictumst. Proin molestie augue lectus. Vestibulum rutrum libero et erat hendrerit semper. Vestibulum at dolor vel ligula sollicitudin lobortis. Nunc id tincidunt dolor. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Vestibulum a nulla vitae mauris semper cursus. Nulla aliquet, lorem nec semper hendrerit, odio purus ornare sapien, ac finibus nulla nulla non justo. Maecenas sapien nulla, dapibus vitae nunc eu, molestie mattis lorem. Nullam cursus efficitur finibus.

Fusce venenatis placerat augue, in tristique purus tempus sed. Pellentesque ac erat et turpis vulputate blandit id nec erat. Vestibulum sollicitudin ac dolor ut iaculis. Curabitur fringilla lacus risus, eu sollicitudin libero mollis vel. Nulla quis risus id nisi suscipit mattis. Maecenas non placerat risus, sagittis aliquet mi. Donec tristique elit nec turpis mollis, pulvinar eleifend massa egestas. Mauris luctus mattis ultrices. Vivamus tincidunt malesuada diam, in posuere ligula porttitor nec. Curabitur auctor nibh ut nulla aliquam, et mollis dui blandit.

Ut vehicula id quam ac venenatis. Fusce eget elit rhoncus, interdum leo quis, tempor neque. Proin condimentum accumsan lobortis. Nulla lacinia dignissim posuere. Nulla bibendum tristique eleifend. Aliquam commodo, nisl a interdum iaculis, metus arcu luctus diam, id lobortis arcu massa eget sem. Aliquam cursus hendrerit tellus sed feugiat.

Sed pretium eget dui et maximus. Vivamus fermentum mauris nunc, id consectetur diam congue et. Quisque scelerisque ex et metus ultrices, et lacinia quam posuere. Donec lectus eros, sodales et eleifend ut, dictum non arcu. Morbi venenatis viverra nunc eu tristique. Nunc dolor lectus, finibus vel pharetra vel, finibus sit amet justo. Integer lobortis massa justo, id bibendum neque convallis at. Morbi luctus interdum enim. Duis ornare a quam eu maximus. Quisque magna magna, hendrerit a elementum at, suscipit id sapien. Integer nisi est, mollis dignissim volutpat et, efficitur eu nunc. Phasellus sed lectus odio. Ut feugiat tempus justo eget dapibus. Aenean justo diam, euismod nec lobortis non, tempus at mauris.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vivamus vel arcu id dui pharetra commodo at dignissim risus. Fusce in ipsum tellus. Phasellus a volutpat ante. Nulla a metus dapibus magna rutrum porttitor sed id ante. Donec in quam sit amet ipsum dapibus hendrerit in eu metus. Nunc feugiat purus sit amet eros dictum rhoncus. Pellentesque commodo ligula neque, placerat accumsan orci fringilla ut. Sed facilisis, ligula in egestas lacinia, ligula nulla euismod ante, nec sagittis lorem augue et enim. Vestibulum sodales dolor eu ex lacinia, a molestie nisi fringilla. Etiam facilisis quam vel bibendum pharetra. Sed rutrum placerat lectus at ultrices. Sed blandit pretium mi, sit amet finibus turpis suscipit quis. Praesent ligula est, commodo non dolor a, tincidunt facilisis risus.

The position property

```
div#ad {
  position: fixed;
  right: 10%;
  top: 45%;
}
```

Here I am!

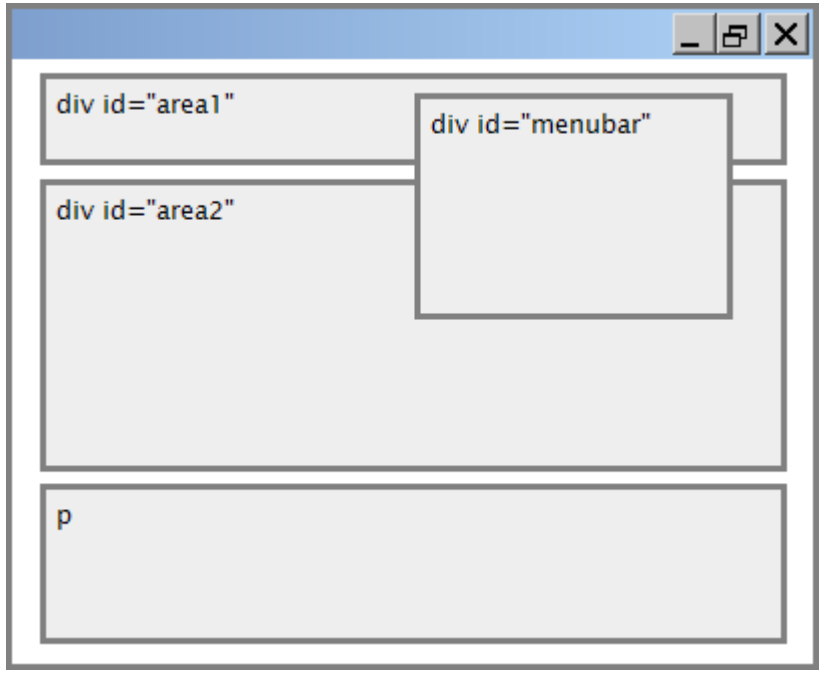
CSS

property	value	description
position	static	default position
	relative	offset from its normal static position
	absolute	a fixed position within its containing element
	fixed	a fixed position within the browser window
top, bottom, left, right	positions of box's corners	

Absolute positioning

```
#menubar {
      position: absolute;
      left: 400px;
      top: 50px;
}
```
CSS

- removed from normal flow (like floating ones)
- positioned relative to the block element containing them (assuming that block also uses `absolute` or `relative` positioning)
- actual position determined by `top`, `bottom`, `left`, `right` values
- should often specify a `width` property as well

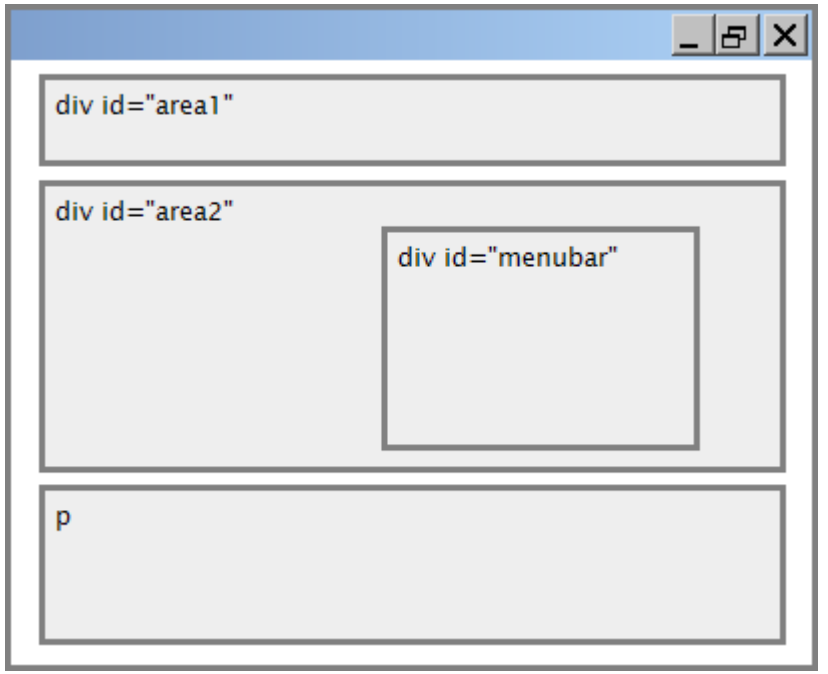


Relative positioning

```
#area2 { position: relative; }                                CSS
```

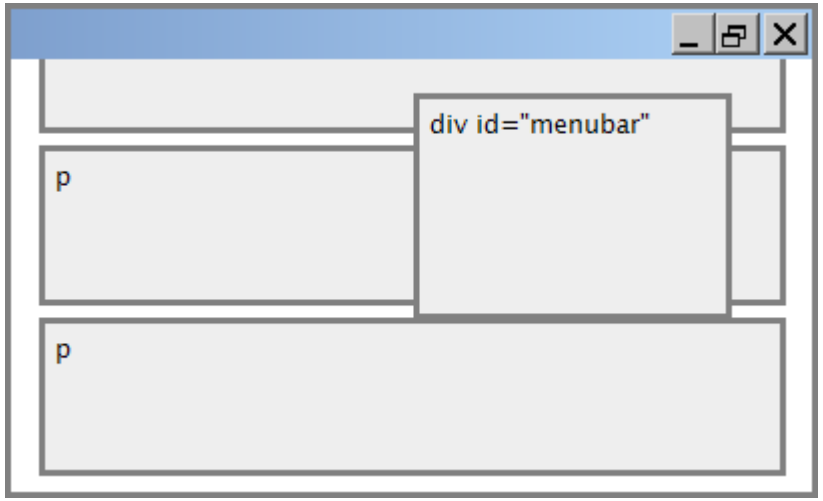
- absolute-positioned elements are normally positioned at an offset from the corner of the overall web page

- to instead cause the absolute element to position itself relative to some other element's corner, wrap the `absolute` element in an element whose `position` is `relative`



Fixed positioning

- removed from normal flow (like floating ones)
- positioned relative to the browser window
  - even when the user scrolls the window, element will remain in the same place



Alignment vs. float vs. position

1. if possible, lay out an element by *aligning* its content
    - horizontal alignment: `text-align`
        - set this on a block element; it aligns the content within it (not the block element itself)
    - vertical alignment: `vertical-align`
        - set this on an inline element, and it aligns it vertically within its containing element
2. if alignment won't work, try *floating* the element
3. if floating won't work, try *positioning* the element
    - absolute/fixed positioning are a last resort and should not be overused

Details about inline boxes

- size properties (`width`, `height`, `min-width`, etc.) are ignored for inline boxes
- `margin-top` and `margin-bottom` are ignored, but `margin-left` and `margin-right` are not
- the containing block box's `text-align` property controls horizontal position of inline boxes within it
    - text-align does not align block boxes within the page
- each inline box's `vertical-align` property aligns it vertically within its block box

The display property

```
h2 { display: inline; background-color: yellow; }
```
CSS

This is a heading **This is another heading**

output

property	description
display	sets the type of CSS box model an element is displayed with

- values: `none`, `inline`, `block`, `run-in`, `compact`, ...
- use sparingly, because it can radically alter the page layout

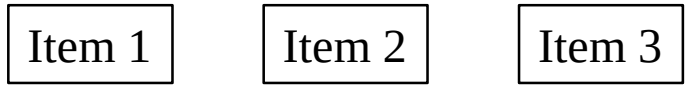
Displaying block elements as inline

```
<ul id="topmenu">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

HTML

```
#topmenu li {
  display: inline;
  border: 2px solid gray;
  margin-right: 1em;
}
```

CSS



output

- lists and other block elements can be displayed inline
  - flow left-to-right on same line
  - width is determined by content (block elements are 100% of page width)