Display and Positioning

Display is Key to Layout

- Every element is a box
- Display affects the layout with regard to neighboring elements

Common Display Values

- inline: sits next to other elements
  - takes up “just enough” width and height

- block: forces line break
  - default: take up all horizontal width and “just enough” height
  - rules can set height and width

Common Display Values cont

- inline-block:
  - same as inline, but accepts height and width
- none: removed from page
  - Still in DOM, but not visual (even to SRs)
- Examples:
  - Display Example
  - Display Example 2

Complementary Properties

- float – USE AS A LAST RESORT**
  - Reposition elements to the right or left.
  - Elements are aware of one another and will not overlap.
  - Values: left, right
- clear
  - Used to keep floating elements away
  - Values: left, right, both

**I won’t help people debug code that uses float.

Element Overflow

- What happens when you set a height/width and the content doesn’t fit any longer?
- Use overflow to determine access

Overflow

- visible: Can cause text to show up “on top” of other text
- hidden: Hides anything that goes beyond bounding box
  - This can cause problems since if the user increases font size, they may not be able to see content
- scroll: Gives horizontal and vertical scrollbars
- auto: Adds scrollbars as needed

Visibility

- Specifies whether or not element is visible
- Options include:
  - visible, hidden, collapse (only for table elements)
- Unlike display:none a hidden element is still part of the DOM and still takes up space

- Practice: Exercises 1 – 4

Box Model

- The default width of inline elements is the content
- Elements that are not inline can take width and height properties

Border

- Any element can have a border around it
- border property specifies *style*, *width*, and *color*
- The border style MUST be specified

```
footer{
        border: solid 1px #CC00AA;
}
```

Specifying Individual Sides

```
border-width: 3px;
```



```
border-width: 3px 10px;
```



```
border-width: 3px 10px 20px;
```



```
border-width: 3px 10px 20px 1px;
```


Margin

- Margin is additional space *outside* your border – between you and neighbor

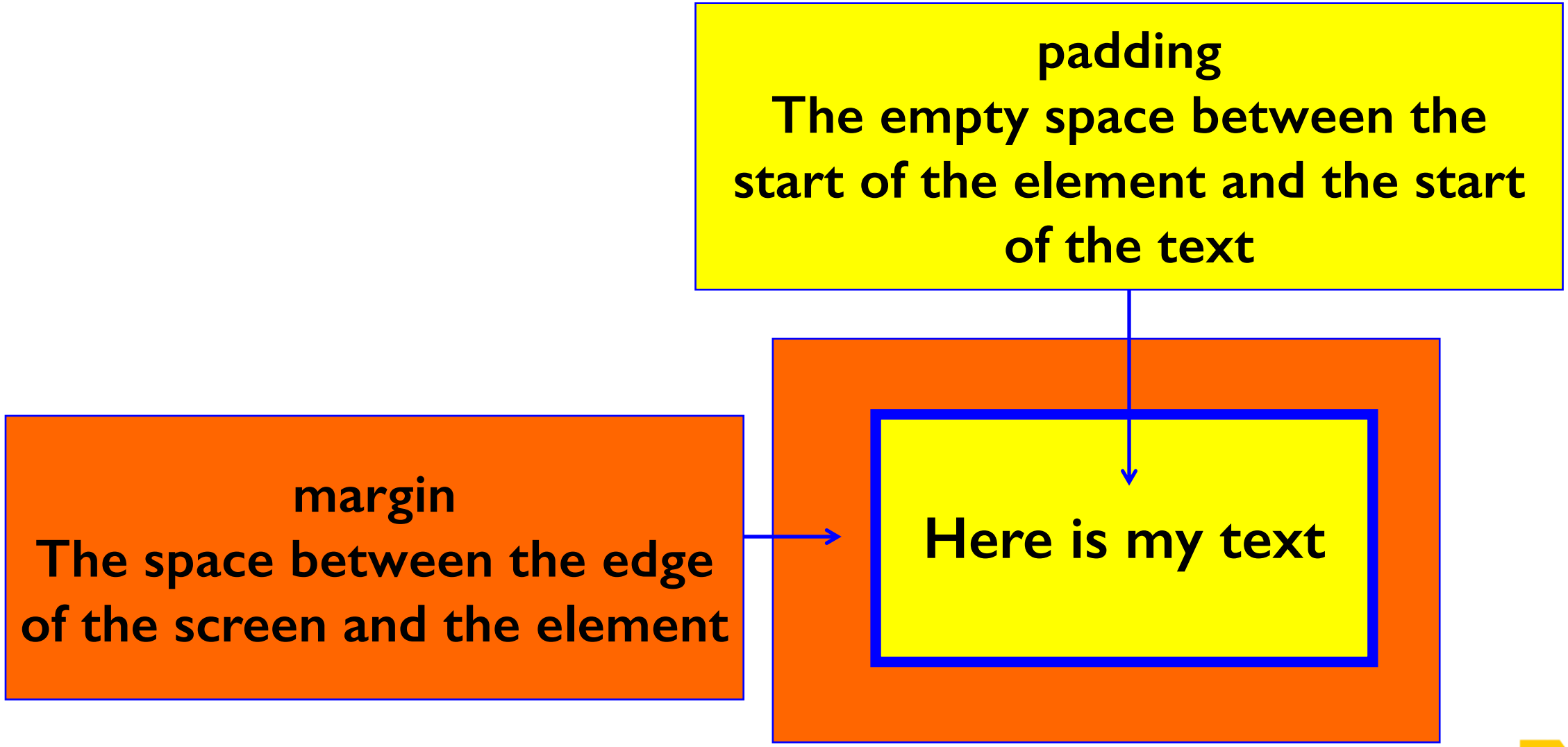
- Positive margin
  - element moves right/bottom

- Negative margin
  - element moves left/upward

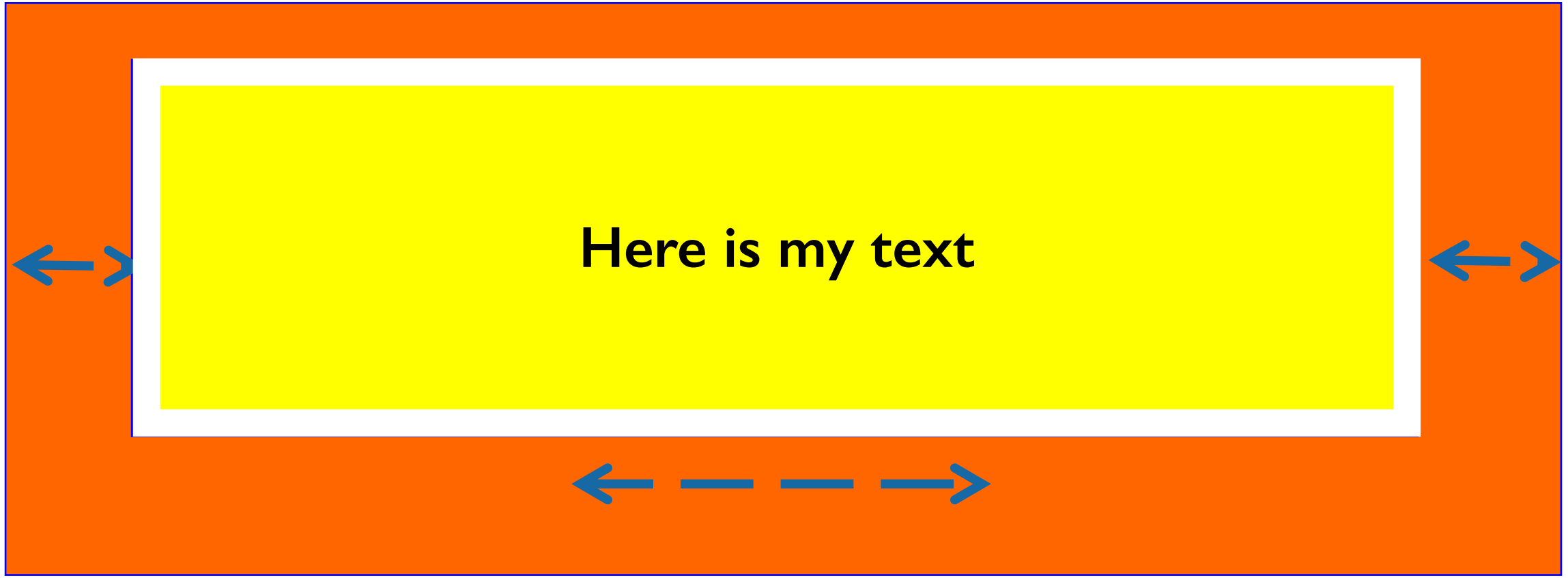
Padding

- Padding is additional space *between* the element and its border.
- Positive padding
  - border moves outward from element
- Neither takes a color (transparent) but both can be be defined with 1-4 values

Full box model



Additive Height and Width



margin + border + padding + width= actual width

What is the width and height?

```
footer{
        width: 100px;
        width: 50px;
        width: 10px;
        width: 5px;
        width: 1px solid black;
}
```

width = 132px:

height = 82:

box-sizing

- box-sizing takes some of the “math” out

- Options:
  - content-box: default additive
  - border-box: width takes content, padding, and border into consideration

Measurements

- Absolute – set to a specific size
  - px, mm, cm, pt, ….
- Fluid – sets size relative to surrounding elements
  - %, vw, vh
  - em (for font): 1em is current size, .75 is 75% of the current size
  - rem (for font): 1rem is current size of root element

Example

- Practice: Exercises 1 – 4

- Example: boxModel.html

  Can you?

  1. Put these divs next to each other? - Make the screen smaller, still working?
  2. Make them taller?
  3. After you do that, can you now center them under each other?

Review

- Design sketches should be done with box model (margin, border, padding, content) in mind.
- Use box-model to reduce complexity
- Margin must always be considered
- Use fluid sizes for best viewing

Positioning!

- Putting elements where you want them can be time-consuming and frustrating.

- Why not tables?

- Make sure to do the reading on positioning and play with the code samples!!!

Position Properties

- The four position properties are:
  - static
  - relative
  - absolute
  - fixed

- Position properties are modified by the properties: top, right, bottom, left

Static

- Default value for elements
- Place in the next available position
- Not affected by the top, bottom, left, and right properties.

Relative

- Positioned “relative to itself”
- Take the static position, but add offsets.
- The new positioning does NOT affect any other element. It is possible to move an element and leave a big hole where it would have been.
- Relatively positioned elements are often used as container blocks for absolutely positioned elements.

Absolute

- Element is removed from the document flow and positioned relative to it's ***nearest ancestor*** (or the root)
- Other elements behave as if element does not exist
- Can end up on top of another element

Fixed Position

- Positioned relative to the *browser window*
- Will not move, even if the window is scrolled
  - IE7 and IE8 support the fixed value only if a !DOCTYPE is specified
- Think of popup boxes that wont’ go away!!!
- Or a navigation bar that is always visible on the top

Positioning Example

- Practice: Exercises 1 – 5
- Example: positioning.html
  1. Open this page - the paragraphs are all in their default location. The top/left values are ignored.
  2. Using Inspect Element, uncomment the next line so the position is relative.
  3. Change the position to absolute.
  4. Change the position to fixed.

Z-index

- Multiple elements may be placed in the same position.
- z-index is a numeric value, positive or negative that dictates stacking order

Accessible Navigation

- Navigation is a critical aspect of accessibility
- Sighted users have tried and true visual cues to orient them on a page
  - Banner, Search box, Main navigation box, Content well
- Blind and low-vision users rely on proper coding of page for orientation

Proper <h1> heading

- Screen readers can find and list headings
- <h1> heading uniquely identifies the page in the website
- Should be placed directly in front of the main content of the page
- The <h1> header should also match at least a subset of the the page <title>

Proper heading hierarchy

- Headings need to be properly nested to convey organization of the page

```
<h1></h1>
    <h2></h2>
        <h3> </h3>
    <h2></h2>
    <h2></h2>
```

Skip To Content

- Allows SR users a navigational aid without cluttering presentation
- https://webaim.org/techniques/skipnav/

```
.offpage{
    position: absolute;
    left: -1000px;
}
```

```
.offpge:focus{
    position: absolute;
    left: 10px;
}
```

Meaningful link text

- Screen readers can find and list links
- Descriptions for the links must be meaningful out of context, via tabbing or presented in a list
- Don’t use “here”, “click here”, “read this”, and “more”
- Don’t use URL as a link description—will sound like gibberish, unless very short and intuitive

Avoiding visual inaccessibility

- Title of page lets you know what page you're on when page loads – every page should have a unique title.

- Proper heading placement and hierarchy conveys organization of page and allows SR users to skip navigation – every page should have "Skip to Content"

- Link descriptions convey content of page and organization of site

Accessibility Review

- How easy is it to navigate your page?
- What would happen if the colors weren’t there?
- What would happen if you couldn’t use a mouse?
- Plan for everyone

Acknowledgements

- Initial Development: Colleen van Lent , University of Michigan School of Information