Positioning

The CSS Box model has every element as a box

- Visually within the parent container box
- `position` css property can change 'within'

Positions

- `static`
- `relative`
- `absolute`
- `fixed`
- `sticky`

position: static

`static` is what you've been doing it all along

Elements are maintained within the document flow

`top` / `bottom` / `left` / `right` CSS properties do nothing

- To an element with `position: static;`

position: relative

- Maintains **space** for element in document flow
- Positions the element relative to that space
- Creates a new **stacking context**
    - That's for later

Rarely move elements using `position: relative;`

- Does occasionally happen

More often used to make element "**positioned**"

- Changes behavior of `position: absolute;` on descendants

`position: absolute`

Pulls the element out of the document flow

- No space is left for it
- Element now defaults to height/width of contents, even as block element
    - It isn't "within" any other element

You can place the element "over" other elements

- Using `top` / `bottom` / `left` / `right`

Placing an absolute element

`top` / `bottom` / `left` / `right` place that side of the element that distance from the listed side of **positioned container**

- Ex: `top: 5px;`
  - Top of element 5 pixels from top of **positioned container**
- Ex: `right: 10px;`
  - Right side of element 10 pixels from right
- Result: Don't need math about container size

But what is the "positioned container"?

Positioned container

By default, `absolute` will be relative to the document.

- Ex: `top: 0;` positions element at top of document
- Probably covering up the top of the document

When an ancestor element has a non-`static` position

- That ancestor element is **positioned**
- `absolute` directions relative to THAT ancestor
- Relative to "nearest positioned ancestor"
    - "nearest" means "closest relation"
    - "parent" is closer than grandparent

Uses of absolute positioning

Show "over" other content

- Use `absolute`
- Often have to position an ancestor element too

Examples:

- Overlay menus
    - Dropdown menus
    - Slide-in menus
- Tooltip-like effects
- Older way to do "modal windows"

`position: fixed`

`fixed` position elements are

- Pulled from the document flow
- No space is given for the element
- Placed relative to the document
    - Like `absolute` with no positioned container

Remain in position *relative to the viewport*

- e.g. a top menu always at top even if you scroll

Fixed position issues

There are issues with fixed positions

- Can get in the way of other elements
    - Ex: Hiding content because overlap
    - Collapsing can help, but complexity goes up
- Can stutter on heavy scroll (allegedly)

Example of Fixed position

https://css-tricks.com/

- The top of screen search
- The bottom of screen cookie
- The bottom of screen scroll-to-top

position: sticky

`sticky` elements start normal while "on screen"

- When normal position in viewport
    - `static` behavior
- When normal position out of viewport
    - And container is IN viewport
        - `fixed` behavior
    - And container OUT of viewport
        - `static` behavior (off screen)

Sticky business

- Position is relative to a "scrolling" ancestor
    - Different browsers = different behavior
- Ex: a big table wants header (or section header) always visible while part table is visible
    - Breaks if wrong part is horizontally scrollable

Example of Sticky positioning

https://dev.to/killjoy__02/position-relative-and-absolute-16gf

- left side menu
- ad at the bottom of the right side menu

Summary - Practical positioning

- `static` is normal
- `relative` to create **positioned container**
- `absolute` to put "over" other content
    - Often involves positioned ancestor
- `fixed` to keep on screen when scroll
    - Can cover content unexpectedly
    - Best for floating "footer"
- `sticky` to keep on screen when scroll
    - Best for section headers
    - Best for floating "header"
    - Has issues with horizontal

Summary - relative positioning

- Element is **positioned**
  - All non-`static` are positioned
  - Relative used if that is sole point
- Keeps space for element
- Allows offset
  - using `top/right/bottom/left` properties
- Offscreen content still IN document
  - impacts a11y

Summary - absolute positioning

- Space NOT reserved in document flow
- Content will visually overlap
- Position relative to **positioned container**

Summary - fixed positioning

- Space NOT reserved in document flow
- Placed relative to viewport
    - NOT positioned container
- Used for visible headings/menus on scroll
- Can cover elements unexpectedly

Summary - sticky positioning

- Space IS reserved in document flow
- Sometimes `static`, sometimes `fixed`
- Keeps section headers onscreen while scroll
- Keeps primary page header onscreen while scroll
- Based on container (parent) being on screen
- Can get confused with horizontal scrolling