CSS Rules and Conceptual Model:

Display property

- Block level vs inline elements:
    - Layout blocks by stacking them vertically (each one on new line), stretch horizontally.
    - Each element has a default type.
- Others:
    - None: doesn't take up space (contrast with visibility)
    - New modes added to make table-like, two-dimensional/complex layout easier to do without using <table> (which is wrong semantically)
    - Inline-block: A mix of inline and block layout
    - Flex: The flexible box model you typically find in traditional GUI system as well as book publishing

Box Model

- Details of how blocks are shown
- Margin - Border - Padding - Content
    - A box is bounded by the border / Margin is the space between model
    - But width only count the content (!)
    - Also, unintuitive behavior: margin collapsing
- Solution (other than custom math): box-sizing property
    - Set to border-box
    - Width now counts up to the border (i.e. turns to inset)
    - Need vendor prefix to work cross browser

Positioning mode

- One of the two fundamental mechanism to making complex layout
- Fixed: Relative to the **viewport**
- Static: Default. An element is *positioned* iff it is any mode other than this default.
- Relative: Displacement from the default position.
- Absolute (misleading term): Fixed positioning relative to the *nearest positioned ancestor*.

Floats

- The other basic mechanism (more tricky)
- Floated elements (to left or right) are plucked out of normal flow of layout
- Then re-inserted (one by one if more than one floated elements) while making texts wrap around it

    - If this wrapping is not desired, use clear to fully remove it from layout flow (i.e. other elements box layout around/after it)
- The Clearfix hack
    - Cause: Floated element may overflow outside the wrapping element if it is too large
    - Fix: Use overflow: auto on affected wrapping element - it will enlarge to make things fit
    - This hack is tricky if real backward compatibility is needed

Inline-Block

- The one dimensional-ness of modern CSS layout come from block layout being stacked vertically
- Stacking is like inline element: from left to right
- But unlike inline, allow:
    - Top and bottom margin and padding
    - Width and height to be set
    - (Beware of linebreaks and wrapping)

Flexbox

Other: Media Query and Responsive design

- Media Query qualify styling rule on device width or mode (e.g. printing)
- Switch what design to use on different range of device width

Layout Techniques:

Centered block

- Set max-width, left and right margin auto
- Explain: both side auto implies balanced margin. Use max-width due to need to be responsive (device width narrower than the box)

Fixed Navbar/Header/Footer

- Set the header/footer to position: fixed
- Then add appropriate margin to the body

Container based/Absolute layout

- Fixed element (such as sidebar) is done by applying position: absolute
- Need to have a container div with position set (in principle anything other than static)
- Regular element then adjust (to avoid overlap) by adding margin

Container based/Float layout

- Same as absolute layout
- Except that sidebar rely on floating
- Then apply clearfix to the container
    - No overflow problem when sidebar is too long

Container based/Inline-block layout

- Mix of usual block and inline-block elements
- With the appropriate nesting
- This results in a mix of horizontal and vertical stacking
- Also need *vertical align* property
- Main Cons: May need lots of extra div/divception

Principles of CSS frameworks:

Grid System
TODO

Fancy Styling: Input box

- Use box-sizing plus padding so that inputted text don't touch the edge
- Set the border (optionally with round corner via border-radius) as usual
- Add a shadow to make it look nice
    - box-shadow: inset 0 1px 3px #ddd;
- Finally, use :focus pseudo element to change border color
    - Remember to remove the default highlighting that happen during input by setting outline to 0

New References:
https://stackoverflow.com/questions/9189810/css-display-inline-vs-inline-block

http://necolas.github.io/normalize.css/
An alternative to reset

https://zellwk.com/blog/responsive-grid-system/

https://www.sitepoint.com/understanding-css-grid-systems/