# L3P2

## Objective:

1. Reset style sheet
2. Page layout designs
3. Center block element
4. floating element
5. Clear floating layout
6. Prevent container collapse
7. Grid-based layouts
8. Layout grid
9. Format grid
10. CSS grid styles
11. Positioning styles
12. Relative positioning
13. Absolute positioning
14. Overflow content

## Display style

HTML Elements:

1. Block elements
2. Inline elements

Defined using: `display: type;`

## Reset Style Sheet

Replace browser default styles, give consistent starting point

1. First style rule is for HTML5 to display all structure elements as blocks (hence display:block;). Adds insurance to ensure structure elements rendered correctly in ancient browsers.

## Page Layout Designs

1. Categories:
    1. Fixed layout: size fixed
    2. Fluid layout: size set as percentage of available screen
    3. Elastic layout: image/text sized in ratio to others
2. Responsive design
    - Layout & design change depending on screen resolution

## Width & Height

1. WxH set with:
    1. width: value;
    2. Height: value;
2. Value = CSS measurement unit / percentage of W or H

## Center block element

- Set both left and right margins to `auto`

## Vertical Centering

- Set `vertical-align` to `middle`

## Floating Page Content

- Floating element takes it out and place it along left/right side of parent element
- `float: position;`

## Clearing Float

- `clear: position;`
- `position`
    - Display element only when `position` margin is clear (except `none`)

## Refining Floated Layout

Use `box-sizing: type;`. Usually with reset style sheet. Some older browsers require `webkit` or `moz` box sizing to work,

- `content-box` model (default)
    - `width` property includes content only
    - total width = width + border + padding
- `border-box` model
    - `width` includes content, padding, border space
    - total width = width - border - padding
- `inherit` model
    - Inherit container's property

## Container collapse

- Empty container, element inside floated
- Use `after` to add placeholder element

General rule (AKA the "clearfix"):

```
container::after {
   clear: both;
}
```

## Grid-based Layouts

### Row & columns form grid

- Row based on page content
- Column based on number that provides most flexibility in laying out content

### Advantages:

- Add order
- Consistent logical design
- Easily accessible
- Increase development speed

## Fixed & Fluid Grids

- Fixed grids: Column specified in pixels
- Fluid grids: Column specified in percentage

## CSS Frameworks

- Software package, provides library of tools to design website
- Include style sheets & built in scripts

- Popular examples:
  - Bootstrap, YAML4

## Setting grid

- Based on rows of floating elements
- Common to use `div` to mark rows & columns
- Common to give columns class name indicating width. Ex: `col-num-den` = numerator/denominator fractional width

## Designing grid rows

- Grid rows contain floating columns
- Display when both margin clear
  - `clear both` to display after last row ended
  - 'clearfix' to prevent collapsing

### Setting column widths

```
div.col-1-1 {width: 100%;}
div.col-1-2 {width: 50%;}
...
```

### Styling column elements

```
div[class^="col-"] {
    float: left; /* left/right/none/inherit */
}
```

`div[class^="col-"]` = select all `div` elements with the attribute `class` which begins with `"col-"`

## Outlining grid

Outlines - Line drawn enclosing entire element

- `outline-width: value;`
- `outline-color: color;`
- `outline-style: style;`

## Defining CSS grid

Create grid display without `div`

```
selector {
    display: grid;
    grid-template-rows: track-list;
    grid-template-columns: track-list;
}
```

`track-list` can use `fr` unit: Fraction of available space left after other row/column reach max size

Everything else: Google

## Assigning Content to Grid Cells

To place element in different cell, use:

- `grid-row-start`
- `grid-row-end`
- `grid-column-start`
- `grid-column-end`

# CSS Positioning Styles

Place element at specific position in container

```
position: type;
top/right/bottom/left: value;
```

### Positionings

- Static (default): Placed at natural place
- Relative: Moved out of normal position
- Absolute: At specific coordinates
- Fixed: Avoid movement
- Inherited: Inherit parent element

More (such as sticky) at: W3School CSS Positioning

## Handing overflow

- `overflow: type;` control browser that handles excess content
- `type` = `visible` by default

### Types (if don't know then look them up)

- `visible`: show all
- `hidden`: cut off excess
- `scroll`: add scroll bars
- `auto`: add scroll bars when needed

CSS3: `overflow-x` and `overflow-y`

## Clipping element

Clip - Rectangular region where element can be seen, outside is hidden

`clip: rect(top, right, bot, left)`

## Stacking elements

Default: Elements loaded later display on top of elements loaded earlier

### Different stacking order

- Use `z-index: value;` property
  - Only works with `absolute` position
  - relative to others with common parent

More info: MDN z-index