CSS Layout Notes

1. Overview

- Focus: Element positioning, site layouts, designing for various screen sizes.
- Topics:
  - Positioning: normal flow, relative, absolute, fixed, floats.
  - Screen size/resolution considerations.
  - Fixed vs. liquid layouts.
  - Grids and CSS frameworks for professional designs.

2. Building Blocks

- HTML elements as boxes:
  - *Block-level*: New line (e.g., `<h1>`, `<p>`, `<ul>`, `<li>`).
  - *Inline*: Flow within text (e.g., `<img>`, `<b>`, `<i>`).
- Control size with `width`, `height`; separate with `borders`, `margins`, `padding`, `background-color`.
- **Containing Elements**: Block-level `<div>` groups content (e.g., header with logo, navigation).

3. Positioning Schemes

- **Normal Flow** (`position: static`):
  - Default; block-level elements stack vertically.
  - No CSS needed; elements take full width unless specified (e.g., `width: 450px`).
- **Relative Positioning** (`position: relative`):
  - Shifts from normal position using `top`, `bottom`, `left`, `right` (e.g., `top: 10px; left: 100px`).
  - No impact on surrounding elements.
- **Absolute Positioning** (`position: absolute`):
  - Removes from normal flow, positions relative to containing element.
  - Uses `top`, `bottom`, `left`, `right` (e.g., `top: 0px; left: 500px; width: 250px`).
  - Surrounding elements ignore its space.
- **Fixed Positioning** (`position: fixed`):
  - Absolute positioning relative to browser window.
  - Stays fixed during scrolling (e.g., `top: 0px; left: 50px`).
  - Use `margin-top` to avoid overlap (e.g., `margin-top: 100px`).
- **Floating Elements** (`float: left` or `float: right`):
  - Moves to left/right of container, becomes block-level.
  - Content flows around (e.g., `float: right; width: 275px`).
  - Requires `width`.

- **Z-Index** (`z-index`):
  - Controls stacking order (higher value = front, e.g., `z-index: 10`).
  - Used with relative, absolute, fixed positioning.

4. Screen Sizes and Resolutions

- Devices vary in size/resolution, requiring responsive design.
- Layouts typically 960–1000px wide; key content in top 600px.

5. Fixed Width vs. Liquid Layouts

- **Fixed Width**:
  - Pixel-based (e.g., `body { width: 960px; margin: 0 auto }`).
  - Columns use `float: left` (e.g., `.column1, .column2, .column3 { width: 300px; float: left }`).
  - Consistent across window sizes.
- **Liquid Layout**:
  - Percentage-based (e.g., `body { width: 90%; margin: 0 auto }`).
  - Columns adjust (e.g., `.column1, .column2, .column3 { width: 31.3%; float: left }`).
  - Use `min-width`, `max-width` for boundaries (IE7+ support).
- Both use `overflow: auto` for floated content.

6. Layout Grids

- Grids ensure consistent proportions/spacing.
- **960 Pixel Grid**:
  - 960px wide, 12 columns (60px each, 10px margins).
  - Supports varied layouts (e.g., two 460px, three 300px, six 140px columns).
  - Benefits: Continuity, predictability, easier content addition, collaboration.

7. CSS Frameworks

- Pre-written CSS for grids, forms, etc.
- **960.gs Grid System**:
  - Uses $\texttt{container}_1 2 for 12 - columngrid(960px).Classes : \texttt{grid}_3, \texttt{grid}_4, \texttt{grid}_1 2 forcolumnwidths.\texttt{clearfi}$
- **Advantages**: Saves time, browser-tested.
- **Disadvantages**: Non-semantic classes, code bloat.
- Other frameworks: Blueprint, YUI Grids, Less Framework.

8. Multiple Style Sheets

- Split CSS into files (e.g., layout, typography, tables).
- Methods:
  - *@import*: One `<link>` imports master CSS, which uses `@import url("file.css")`.
  - *Multiple <link>*: Separate `<link>` for each CSS file.
- `@import` rules precede other CSS rules.

9. Example Application

- Magazine-style layout with 960.gs:
  - Fixed header (`position: fixed; top: 0px; z-index: 50`).
  - $\texttt{container}_12, \texttt{grid}_12(header/footer), \texttt{grid}_6, \texttt{grid}_3(content).Backgroundimages(e.g., \texttt{background:}$

10. Summary

- `<div>` groups content as containers.
- Positioning: `static`, `relative`, `absolute`, `fixed`, `float`.
- `z-index` manages overlap.
- Layouts: Fixed (pixels) or liquid (percentages).
- Grids (e.g., 960.gs) ensure professional designs.
- CSS frameworks simplify tasks, may add bloat.
- Multiple style sheets via `@import` or `<link>`.