DOMS | Page No.
Date / /

Advance CSS

Quick Overview

Core topics CSS display block /inline/ inline-block /etc) float how / when to use & why its mostly legacy for layout), making responsive websites, & media queries - plus a small project to apply those skills

Learning goals

1. Explain the different CSS display modes & when to use each
2. Use floats correctly (& know its limitation & modern alternatives)
3. Build a responsive layout using the viewport meta tag & media queries (mobile first approach)
4. Apply a simple responsive site pattern (change layout / typography across breakpoint
5. Debug layout issues using power browser Dev tools (inspect computed styles, box models, & media query breakpoints)

DOMS | Page No.
Date / /

1) Css (Display)

Key concepts

display determines how an elements participantes in the flow & how much box-space it takes

Main

block occupies full available width, starts on a new line <p>
inline sits inline with text only as wide as it content <a>
inline-block like inline but can have width/height (very useful for small components)
none elements removed from layout flow (not visible, not accessible)
flex, grid modern layout display types (covered later in course but important to mention here as alternatives

visual / mental model

Imagine each elements is a box boxes stack virtically like bricks, inline boxes sit like beads on a string inline-block are beads that can be resized, flex & grid are advanced toolkits for arranging groups of boxes

```
<!-- block : will take full width -->
<div class = "card" > card content </div>
<!-- inline : sits with text -->
<p> some text <span class "tag" > inline tag </span> more text . </p>
<!-- inline-block : sits inline but can be sized -->
<span class = "chip" > 42 </span>
<styles>
.card { display : block , background : #F6F6F6 ; padding : 12px }
.tag { display : inline ; background : yellow ; }
.chip { display : inline-block ; width : 40px ; height : 24px ; line-height : 24 px }
```

DOMS | Page No.
Date / /

Common Students Pitfall

- Expecting inline elements to accept width/height: they don't use (inline-block)
- using display: none to "hide" things you still need accessible - it removes the elements from layout & screen reader flow
- Confusing visibility: hidden (keeps space but hides content) vs display: none (removes space)

2) CSS float

What float does
originally intended to wrap text around images (like in print)
it takes an element out of the normal document flow.
pushes inline content around it. It was later hacked for entire page layout (not ideal)

How to use
- float an image to the left, let text wrap to the right. when floating containers, it's common to clear the float afterward so parent container size correctly

```
<img src="pic.jpg" class="left" />
<p> Long para that will wrap around the image... </p>
<style>
.left { float: left; margin-left: 12px; width: 200px; }
.clearfix::after { content: ""; display: table; clear: both; }
</style>
```

The Clearfix pattern
when children are floated the parent collapses - fix it by adding a clearfix (pseudo - elements that clears floats)

above uses ::after with 'display: table; clear: both;'

why floats are less used today

Flexbox Grid provides robust, predictable layout systems - prefer these for overall layout. use float only for text wrapping or legacy code maintainence (Good to point this out - as a best practice)

3) Responsive Web Design

viewport meta tag
Always include in <head> for proper mobile scaling

<meta name='viewport' content="width=device-width, initial-scale=1"/>

- Explain students with otherwise see a zoomed out desktop view on phones

Mobile first vs desktop first

Mobile first write base style for small screens then add min-width media queries to expand layout for larger screens.
This encourage performant, mobile-optimized CSS.
Explain advantages: smaller CSS for mobile, progressive enhancement, better maintainability.

DOMS | Page No.
Date / /

Breakpoints
- use semantic breakpoints tied to design not device names.

small (mobile): default (no media query)
tablet @media (min-width: 768 px) {...}
desktop @media (min-width: 1024 px {...}

Encourage: test on real devices or responsive dev tools

4) Media Queries

/* mobile-first : apply on >=768 px */
@ media (min-width: 768 px) {
  .container { max-width: 720 px }
}

Example

- Stack to grid patterns: single column on small screens → two colums on tablets → three columns on wide screen

- Rsponsive typography: increase font-size at larger breakpoints for readability

.card { padding: 12px; font-size: 16px; }
@ media { min-width: 768 px) {
  .card { font-size: 18 px; }
}

DOMS | Page No.
Date / /

Debugging media queries in Chrome

- Use dev tools "Toggle device toolbar" & resize to confirm CSS changes at the intended breakpoints
- Show computed styles panel to demonstrate which rule is active & which media query matched.

5) Practical Project
Project "Small Agency landing page" responsive layout

- Req:
  - Header with logo + nav (hamburger on small screens)
  - Hero section with centered headline & CTA
  - Services grid: 1 column (mobile), 2 column (tablet) 3 column (desktop)
  - footer with links stacked on mobile inline on larger screens

Teaching steps

1) Scaffold HTML: Semantic elements <header>, <main>, <section> <footer>
2) Style base (mobile first)
3) Use inline-block or simple flex for nav items; demonstrate switch to hamburger via media query
4) Add breakpoint to change grid layout
5) Test in DevTools; show how to debug collapsed parents & float issues with DevTools highlighting

Page No.
Date / /

6) Debugging tips & Dev tools walkthrough

- inspect element → check computed panel for display, width, height, margin, padding
- use the box model visualization to show content / padding / border / margin.
- use Elements panel to toggle classes & see live effect
- use the responsive device toolbar to simulate breakpoints, add throttled network / CPU for mobile testing

7) Common mistakes & how to fix them

- Parent collapse after floating children → add clearfix or avoid floats by using flex / grid
- using fixed pixel widths for layout → switch to relative units (%, vw, rem) for responsive
- Media queries with max-width Vs min-width confusion - pick a strategy (mobile-first) min-width & stick to it
- forgetting viewport meta tag → site looks scaled on mobile

DOMS | Page No.
Date / /

8) Assessment

1) what is the difference between display: inline & inline-block?

2) why is the clearfix needed? Show the CSS for it

3) write a media query that changes a 1-column grid to 3 column on screens 1024px & wider

4) Explain why float is not typically used for full page layout today

5) what does display: none do to accessibility & layout?

9) Suggested in class exercise

1) Mini task Convert simple two-column layout written with floats into a flexbox layout. Discuss advantages

2) Homework Build the agency landing page (from the project spec above) Provide different breakpoints to achieve responsive design)

3) Extension Add a responsive image (use srcset / sizes) & discuss performance trade offs

DOMS | Page No.
Date / /

10) Cheatsheet

- display : block new line, full width
- display : inline no width / height, inline
- display : inline-block inline but sized
- float ~~text wrap~~ use clear / clearfix prefer flex / Grid for layout
- Always include <meta name="viewport" content="width=device-width, initial-scale=1">
- Mobile first @media (min-width : Xpx) is preffered
- Use Dev Tools to inspect which media query is active & what rules override others