Mi-4 Cheatsheet:

Recap Tailwind:

🅰️ BASICS

Concept	Utility	Example
Font Size	`text-xs` → `text-9xl`	`text-lg`
Font Weight	`font-thin` → `font-black`	`font-bold`
Text Align	`text-left`, `text-center`, `text-right`, `text-justify`	`text-center`
Text Color	`text-red-500`, `text-blue-700`, etc.	`text-green-600`
Background Color	`bg-red-500`, `bg-gray-100`, etc.	`bg-blue-200`
Border	`border`, `border-2`, `border-red-400`	`border` `border-gray-300`
Border Radius	`rounded`, `rounded-md`, `rounded-full`	`rounded-lg`
Display	`block`, `inline-block`, `flex`, `grid`, `hidden`	`flex`

🅱️ BOX MODEL

Property	Utility	Example
Padding	`p-0` → `p-96`, `px-4`, `py-2`, `pt-3`	`px-4` `py-2`
Margin	`m-0` → `m-96`, `mx-2`, `my-5`, `mt-3`	`mt-5` `ml-4`
Width	`w-0`, `w-px`, `w-1/2`, `w-full`, `w-screen`	`w-1/3`

Height	`h-0`, `h-px`, `h-1/2`, `h-full`, `h-screen`	`h-64`
Max Width	`max-w-xs` → `max-w-7xl`	`max-w-md`

C FLEX & GRID

Concept	Utility	Example
Flexbox	`flex`, `inline-flex`	`flex`
Flex Direction	`flex-row`, `flex-col`, `flex-wrap`	`flex-col`
Justify Content	`justify-start`, `justify-center`, `justify-between`	`justify-around`
Align Items	`items-start`, `items-center`, `items-end`	`items-center`
Grid Layout	`grid`, `grid-cols-1` → `grid-cols-12`	`grid grid-cols-3`
Gap	`gap-1` → `gap-10`	`gap-4`

D POSITION & Z-INDEX

Concept	Utility	Example
Position	`static`, `relative`, `absolute`, `fixed`, `sticky`	`absolute`
Top/Right/Bottom/Left	`top-0`, `left-1/2`, etc.	`top-4 left-4`
Z-Index	`z-0`, `z-10`, `z-50`, `z-auto`	`z-30`

E EFFECTS

Concept	Utility	Example
Shadow	`shadow`, `shadow-md`, `shadow-lg`, `shadow-xl`	`shadow-md`
Opacity	`opacity-0` → `opacity-100`	`opacity-75`
Mix Blend	`mix-blend-normal`, `mix-blend-multiply`	`mix-blend-screen`
Blur	`blur-sm`, `blur`, `blur-md`, `blur-lg`	`blur-md`
Transition	`transition`, `transition-all`, `duration-300`	`transition duration-200`
Hover	`hover:bg-blue-500`, `hover:text-white`	`hover:shadow-lg`

🅵 RESPONSIVE DESIGN

Breakpoint	Prefix
`sm:`	≥640px
`md:`	≥768px
`lg:`	≥1024px
`xl:`	≥1280px
`2xl:`	≥1536px

Example: `md:text-xl`, `lg:flex`, `sm:hidden`

🅶 TYPOGRAPHY

Concept	Utility	Example
Font Style	`italic`, `not-italic`	`italic`

Text Transform	`uppercase`, `lowercase`, `capitalize`, `normal-case`	`uppercase`
Line Height	`leading-none`, `leading-snug`, `leading-relaxed`	`leading-tight`
Letter Spacing	`tracking-tight`, `tracking-widest`	`tracking-wide`

🅷 INTERACTIVITY

Concept	Utility	Example
Cursor	`cursor-pointer`, `cursor-wait`, `cursor-not-allowed`	`cursor-pointer`
Pointer Events	`pointer-events-none`, `pointer-events-auto`	`pointer-events-none`
User Select	`select-none`, `select-text`, `select-all`	`select-none`

🅸 LIST & TABLE

Concept	Utility	Example
List	`list-none`, `list-disc`, `list-decimal`, `list-inside`	`list-disc list-inside`
Table	`table`, `table-auto`, `table-fixed`, `table-row`, `table-cell`	`table-auto`

🅹 BACKGROUND

Concept	Utility	Example

Background Size	`bg-cover`, `bg-contain`, `bg-auto`	`bg-cover`
Background Position	`bg-center`, `bg-top`, `bg-bottom`	`bg-center`
Background Repeat	`bg-no-repeat`, `bg-repeat-x`	`bg-no-repeat`

🅺 MISCELLANEOUS

Concept	Utility	Example
Overflow	`overflow-auto`, `overflow-hidden`, `overflow-scroll`	`overflow-x-hidden`
Object Fit	`object-cover`, `object-contain`, `object-fill`	`object-cover`
Visibility	`visible`, `invisible`	`invisible`
Aspect Ratio	`aspect-auto`, `aspect-square`, `aspect-video`	`aspect-video`

🅻 FORMS

Concept	Utility	Example
Input Style	`form-input`, `form-select`, `form-textarea`, `form-checkbox`	`form-input`
Focus States	`focus:outline-none`, `focus:ring-2`	`focus:ring-blue-500`

🔚 ADVANCED/PLUGIN-BASED

Concept	Plugin Required	Example
Line Clamp	`line-clamp-3`	Needs `@tailwindcss/line-clamp`
Aspect Ratio	`aspect-[16/9]`	Needs `@tailwindcss/aspect-ratio`
Typography	`prose`, `prose-lg`, `prose-indigo`	Needs `@tailwindcss/typography`

🧭 Tour of DOM (Document Object Model)

🔹 What is DOM?

- **DOM (Document Object Model)** is a programming interface for HTML and XML documents.
- It represents the page as a tree structure where each node is an object.
- JS uses DOM to manipulate content, structure, and styles.

🔹 `getElementsByTagName(tag)`

- Returns a **live HTMLCollection** of elements with the given tag name.
- Example:

```
const allDivs = document.getElementsByTagName('div');
```

🔹 `getElementsByClassName`, `getElementById`

- `getElementsByClassName('className')`: Returns **live HTMLCollection**.
- `getElementById('id')`: Returns a **single element** with the specific ID.

```
const items = document.getElementsByClassName('item');
const header = document.getElementById('main-header');
```

🔹 `querySelector`, `querySelectorAll`

- `querySelector(selector)`: Returns the **first matching** element.
- `querySelectorAll(selector)`: Returns a **static NodeList** of all matches.

```
const firstItem = document.querySelector('.item');
const allItems = document.querySelectorAll('.item');
```

🔹 Dynamic Style, `getAttribute`, `setAttribute`, `innerText`, `innerHTML`

Property/Method	Description	Example
`element.style.property`	Change inline style	`div.style.color = "red";`
`getAttribute()`	Get value of an attribute	`img.getAttribute('src')`
`setAttribute()`	Set an attribute	`img.setAttribute('alt', 'image')`
`innerText`	Gets/sets **visible** text	`para.innerText = 'Hello';`
`innerHTML`	Gets/sets **HTML content**	`div.innerHTML = '<b>Hi</b>'`

🔹 Add/Remove CSS Classes

```
element.classList.add('highlight');
element.classList.remove('hidden');
```

```
element.classList.toggle('active');
```

🔹 Node vs Collection

Concept	Description
`HTMLCollection`	Live, returned by `getElementsBy...`
`NodeList`	Static, returned by `querySelectorAll`
`parentNode`	Returns parent of node
`childNodes`	Returns NodeList of child nodes (including text)

🔹 Create and Append Elements

```
const newDiv = document.createElement('div');
newDiv.innerText = 'Hello DOM';
document.body.appendChild(newDiv);
```

🧩 Events and Event Handling

🔹 What is an Event?

- **Events** are actions users perform (click, input, hover).
- JS can listen to these events using **event handlers**.

🔹 Add `onclick` Handler

Inline:

```
<button onclick="handleClick()">Click</button>
```

JS File:

```
document.getElementById('btn').onclick = function() {
  alert('Clicked');
};
```

🔹 `addEventListener()`

- **Modern and flexible** way to attach event listeners.

```
button.addEventListener('click', () => {
  alert('Event Listener Clicked');
});
```

🔹 Common Event Handlers

Event	Triggered When
`click`	Click on element
`mouseover` / `mouseout`	Mouse enters/leaves
`keydown` / `keyup`	Key is pressed/released
`submit`	Form submission
`input`	Input field changes
`change`	Input value changes (after blur)

🔹 Event Bubbling & `stopPropagation()`

📌 What is Event Bubbling?

- Events **bubble up** from the target element to its ancestors.

```
<div id="parent">
  <button id="child">Click Me</button>
</div>

document.getElementById('parent').addEventListener('click', () => {
  console.log('Parent clicked');
});
document.getElementById('child').addEventListener('click', (e) => {
  console.log('Child clicked');
  // e.stopPropagation();  // stops bubbling
});
```

Output if bubbling is not stopped:

Child clicked
Parent clicked

🔹 Event Delegation

✅ What is Event Delegation?

- A technique where a parent handles events of child elements using bubbling.
- Useful for dynamic content.

Example:

```
<ul id="list">
  <li>Item 1</li>
  <li>Item 2</li>
</ul>

document.getElementById('list').addEventListener('click', (e) => {
  if (e.target.tagName === 'LI') {
    alert('Clicked on: ' + e.target.innerText);
  }
});
```

⭐ Benefits:

- Reduces number of event listeners.
- Handles dynamically added elements.

- Learn multiple ways to handle events.
- Use addEventListener for flexibility.
- Understand bubbling & delegation to manage events cleanly.
