# BASIC KNOWLEDGE

## CSS

- Discuss with your partner how the DOM is created and displayed on your screen.

The Document Object Model (DOM) is created and displayed on the screen through a series of steps involving the browser's rendering engine. This series of steps—HTML parsing, DOM tree construction, CSS styling, layout and rendering, painting, and event handling—ultimately allows the browser to create and display the web page based on the provided HTML, CSS, and JavaScript.

- What are the best CSS creation practices?

1. **Use a CSS Methodology**: CSS methodologies like BEM (Block Element Modifier), SMACSS (Scalable and Modular Architecture for CSS), or CSS-in-JS can provide structure and organization to your CSS codebase. They encourage component-based thinking and make styles more modular and reusable.

2. **Keep Selectors Specific**: Avoid using overly broad or generic selectors that may unintentionally target multiple elements. Instead, write selectors that are as specific as needed to target only the desired elements. This helps avoid unintended side effects and makes styles easier to maintain.

3. **Follow a Naming Convention**: Adopt a consistent and meaningful naming convention for CSS classes and IDs. Choose descriptive and semantic names that reflect the purpose or function of the elements they represent. This promotes clarity and makes your code more readable and understandable.

4. **Keep Styles DRY (Don't Repeat Yourself)**: Avoid duplicating styles across different CSS rules. Instead, extract common styles into reusable classes or placeholders and apply them wherever necessary. This reduces code redundancy, improves maintainability, and makes it easier to make global style changes.

5. **Avoid Style Coupling**: Try to keep your styles decoupled from the HTML structure as much as possible. Avoid relying on specific HTML element hierarchies or nesting patterns in your CSS. This allows for greater flexibility in modifying the HTML structure without affecting the styles.

6. **Use Flexbox and Grid for Layouts**: CSS Flexbox and CSS Grid are powerful layout tools that provide flexible and responsive layout capabilities. Use

them to create complex and responsive page layouts without relying heavily on floats or positioning hacks.

7. **Minimize the Use of !important**: The !important declaration should be used sparingly, as it overrides other styles and can lead to specificity issues. It is generally best to avoid using !important unless absolutely necessary. Instead, rely on specificity, order of rules, or refactoring to solve styling conflicts.

8. **Optimize CSS Performance**: Write efficient CSS by minimizing the use of unnecessary or redundant styles. Remove unused styles, consolidate and combine selectors, and consider using CSS preprocessors or post-processors for optimization techniques like minification, autoprefixing, and bundling.

9. **Comment and Document Your CSS**: Add comments to your CSS code to explain complex styles, provide context, or document any workarounds or hacks. Documenting your CSS helps other developers understand your code and facilitates future maintenance and updates.

10. **Regularly Refactor and Consolidate**: As your CSS codebase grows, periodically review and refactor your stylesheets. Identify opportunities to consolidate styles, remove unused or unnecessary code, and improve organization. This helps keep your codebase clean, efficient, and maintainable.

- What is specificity and how can you use it to select the elements you want?

Specificity in CSS determines which CSS rules are applied to an element when multiple rules target the same element. It is a way to assign a weight or value to CSS selectors, allowing the browser to determine which styles should take precedence.
When multiple rules target the same element, the rule with the highest specificity value will be applied. However, if two rules have the same specificity, the rule that appears later in the CSS file will take precedence.

- What types of CSS selectors exist; can you name examples?

1. Type Selectors:
    - Target elements based on their tag names.
    - Example: h1, p, div

2. Class Selectors:
    - Target elements based on their class attribute.
    - Example: .my-class, .btn, .container

3. ID Selectors:
    - Target a specific element based on its ID attribute.
    - Example: #my-id, #header, #nav

4. Universal Selector:
    - Targets all elements in the document.
    - Example: *

5. Attribute Selectors:
    - Target elements based on their attribute values.
    - Example: [type="text"], [href^="https://"], [data-tooltip]

- How can we apply CSS code to a HTML page?

1. **Incline CSS**: You can apply CSS directly to HTML elements using the “style” attribute. In this approach, the CSS rules are defined within the HTML tags themselves. Inline styles have high specificity and affect only the targeted elements.

2. **Internal CSS**: You can embed CSS code within the HTML document using the “<style>” tags within the “<head>” section. In this approach, the CSS rules are defined within the “<style>” tags, and they apply to all HTML elements that match the specified selectors. Internal CSS affects the entire HTML document.

3. **External CSS**: You can create a separate CSS file and link it to your HTML document using the “<link>” tag. In this approach, the CSS rules are stored in an external file (e.g., “styles.css”) and linked to the HTML document using the “href” attribute to the “<link>” tag. The CSS file contains all the styles, and any changes made to it will automatically reflect across all HTML pages linked to it.