Advanced Client-Side Development

Advanced CSS Selectors and Specificity

Transform, Transition and Animation.

Responsive Web Design, Responsive Layouts and Media Queries

Flex and Grid

By Ebad Majeed

Attribute Selector

- The [attribute] selector is used to target elements that have a specific attribute.
- Examples:
    - a[target] selects all the anchor elements with target attribute.
    - [type="text"] selects all the inputs where type is text.
    - [href^="https"] selects all the elements where href's value starts with https.
    - [class*="button"] selects all the elements with a class that contains button

UI Pseudo-Class Selector

- These selectors respond to user interactions and the current state of elements in a web page, providing a way to enhance the user experience.
- Examples:
  - **:focus** Applies styles to an element when it receives focus, such as when a user clicks on it or navigates to it using the keyboard (e.g., tabbing).
  - **:active** Styles an element when it is being activated, typically during the time it is being clicked or tapped.
  - **:disabled** Applies styles to elements (like buttons or form inputs) that are disabled and cannot be interacted with.
  - **:checked** Targets input elements like checkboxes or radio buttons that are checked or selected.

Structure Selectors

- Selects elements based on their position in the document.
- Examples:
  - First of type
  - First child
  - Last of type
  - Only child

Negation Pseudo class

- Selecting elements that do not match a certain selector.
- Example

Pseudo Elements

- Pseudo-elements are special CSS selectors that allow you to style specific parts of an element, such as the first line, first letter, or content before or after an element, without altering the HTML structure.
- Examples:
  - Before content
  - After content
  - First line
  - First letter

Specificity

- Specificity is a way of determining which CSS rule applies to an element when there are multiple rules that could apply.

Specificity

- Specificity Calculation:
  - Specificity is calculated based on the types of selectors used in a rule. It's usually represented as a four-part value (a, b, c, d):
  - a: Inline styles (style attribute directly on the HTML element). This is the highest specificity and counts as 1, 0, 0, 0.
  - b: IDs. Each ID counts as 0, 1, 0, 0.
  - c: Classes, attributes, and pseudo-classes. Each of these counts as 0, 0, 1, 0.
  - d: Elements and pseudo-elements. Each of these counts as 0, 0, 0, 1.

Transform

- Allows you to modify the coordinate space of the CSS visual formatting model.
- It can be used to apply various transformations like translation, rotation, scaling, and skewing to an element.
- Examples

Transition

- CSS transitions allow you to change property values smoothly (over a given duration) instead of having them change abruptly.
- You can specify which properties should transition, the duration of the transition, the timing function (ease, linear, etc.), and any delay before the transition starts.
- Examples

Animations

- CSS animations allow you to animate the transition of properties over time using keyframes.
- Unlike transitions, which are triggered by specific events (like hover), animations run automatically when the page loads or when an element is added to the DOM.
- You can define multiple keyframes to create complex animations.
- Examples

RWD

- Responsive Web Design (RWD) is an approach to web design that allows web pages to render well on a variety of devices and window or screen sizes.
- It involves using flexible layouts, images, and CSS media queries to ensure that users have an optimal viewing experience, regardless of the device they use (desktop, tablet, or mobile).

Key Concepts (RWD)

- Flexible Layouts: The layout of a web page adjusts based on the size of the viewport.
- Responsive Images: Images scale based on the viewport size to avoid overflow.
- Media Queries: CSS techniques that apply styles based on the device characteristics (e.g., screen width, height, orientation).
- Example

Flex box

- Flexbox (Flexible Box Layout) is a CSS layout model that allows you to design complex layouts easily and efficiently.
- It provides a more efficient way to align and distribute space among items in a container, even when their size is unknown or dynamic.
- Flexbox makes it easier to design responsive layouts without the need for float or positioning.

Key Concepts

- Flex Container: The parent element that has display: flex;. It enables flex properties for its child elements.
- Flex Items: The child elements inside the flex container that can be manipulated using flex properties.
- Direction: You can set the direction of the flex items (row or column).
- Justification: Aligns items along the main axis (e.g., left, center, right).
- Alignment: Aligns items along the cross axis (e.g., top, bottom, center).

Grid

- CSS Grid Layout is a two-dimensional layout system for the web that allows you to create complex and responsive layouts with ease.
- It enables you to define rows and columns in a grid format, making it simple to place items in specific areas of the layout.

Key Concepts

- Grid Container: The parent element that has display: grid;. This enables grid properties for its child elements.
- Grid Items: The child elements inside the grid container that can be manipulated using grid properties.
- Rows and Columns: You can define the number of rows and columns in the grid and how they should behave in terms of size and spacing.
- Grid Areas: You can assign specific areas in the grid for grid items, allowing for more control over layout positioning.