CONTAINER QUERIES

Container Queries

Definition:

Container queries are a CSS feature that allows you to apply styles to an element based on the size of its container rather than the size of the viewport. This enables more flexible and modular design patterns, particularly useful for creating responsive components that adapt to different layout contexts.

Key Concepts

1. Container Units:

   Definition: Container units are similar to viewport units but are relative to the size of a container instead of the viewport.

   Types:

   - cqw: 1% of the container's width.
   - cqh: 1% of the container's height.
   - cqi: 1% of the container's inline size.
   - cqb: 1% of the container's block size.
   - cqmin: The smaller value of cqw or cqh.
   - cqmax: The larger value of cqw or cqh.

2. Containment:

   Purpose: To mark an element as a containment context, which is a prerequisite for applying container queries.

   Properties:

   - container-type: Defines the type of containment.
     - size: Contains both inline and block size.
     - inline-size: Contains only the inline size.
     - block-size: Contains only the block size.

3. Container Query Syntax:

Structure: Container queries use the @container at-rule, similar to how media queries use the @media at-rule.

Example:

```css
.container {
  container-type: inline-size;
}

@container (min-width: 400px) {
  .child {
    background-color: lightblue;
  }
}
```

4. Container Name:

Purpose: Naming containers allows for more specific targeting of nested containers.

Usage:

```css
.container {
  container-type: inline-size;
  container-name: parent-container;
}

@container parent-container (min-width: 500px) {
  .child {
    background-color: lightcoral;
  }
}
```

5. Nested Containers:

Concept: Containers can be nested within each other, and each nested container can have its own set of queries.

Example:

```css
.outer-container {
  container-type: size;
}

.inner-container {
  container-type: inline-size;
}

@container (min-width: 600px) {
  .outer-container .item {
    font-size: 2em;
  }
}

@container (min-width: 300px) {
  .inner-container .item {
    color: blue;
  }
}
```

6. Supported Properties:

   Properties: Most CSS properties can be used within container queries, similar to media queries. These include width, height, padding, margin, font-size, color, etc

Difference Between Media Queries and Container Queries

Media Queries:

1. Scope:
   - Global Context: Based on the viewport or user device.
   - Applicability: Affects the entire page or specific sections based on the viewport size.
2. Usage:
   - Responsive Design: Adapt the layout for different screen sizes.
   - Syntax Example:

```
@media (max-width: 600px) {
  .box {
    background-color: lightgreen;
  }
}
```

3. Units:
   - Viewport Units: vw, vh, vmin, vmax.

Container Queries:

1. Scope:
   - Local Context: Based on the size of an element’s container.
   - Applicability: More granular control over individual components or sections.
2. Usage:
   - Modular Design: Create components that adapt to their container’s size.
   - Syntax Example:

```
.container {
  container-type: inline-size;
}

@container (min-width: 300px) {
  .item {
    padding: 20px;
  }
}
```

3. Units:
   - Container Units: cqw, cqh.

Feature	Media Queries	Container Queries
Targets	Viewport size and characteristics	Size of a container element
Use case	Responsive layouts based on screen size, device type, etc.	Responsive layouts within a specific container

Practical Use Cases

1. Media Queries:
   - Responsive Layouts: Adjust the overall page layout for different devices.
   - Typography Adjustments: Change font sizes for better readability on smaller screens.
   - Conditional Display: Show or hide elements based on screen size.
2. Container Queries:
   - Component Flexibility: Make components like cards, buttons, or form elements responsive to their container.
   - Nested Components: Ensure nested components adapt within different contexts.
   - Design Systems: Facilitate the creation of consistent and reusable design patterns.

Examples and Advanced Usage

Media Query Example:

```css
/* Responsive typography for different devices */
@media (max-width: 768px) {
  body {
    font-size: 14px;
  }
}

@media (min-width: 769px) and (max-width: 1200px) {
  body {
    font-size: 16px;
  }
}

@media (min-width: 1201px) {
  body {
    font-size: 18px;
  }
}
```

Container Query Example:

```
/* Responsive card component */
.card {
  container-type: inline-size;
  padding: 10px;
  border: 1px solid #ccc;
}

@container (min-width: 400px) {
  .card {
    padding: 20px;
    border: 2px solid #666;
  }
}

@container (min-width: 600px) {
  .card {
    padding: 30px;
    border: 3px solid #000;
  }
}
```

Benefits of Container Queries

- Modular Design: Facilitates the creation of modular and reusable components.
- Context-Specific Styling: Allows components to adapt based on their immediate context rather than the overall viewport.
- Improved Maintainability: By making components self-contained, it simplifies the maintenance and updating of styles.

- Enhanced Flexibility: Provides more granular control over layout and styling changes, making it easier to design complex, responsive layouts.

Conclusion

Container queries represent a significant advancement in CSS, providing developers with powerful tools to create responsive, modular, and maintainable web designs. By allowing styles to be applied based on container size, they complement media queries and open up new possibilities for component-based design.

References

- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_containment/Container_queries
- https://medium.com/gravel-engineering/container-queries-do-we-still-need-media-queries-2440aabd3383#:~:text=Container%20queries%20allow%20us%20to,their%20position%20within%20the%20UI.