Fundamentals of CSS III

CSS Combinators

CSS (Cascading Style Sheets) combinators are powerful tools that allow you to select and style HTML elements based on their relationships with other elements. Combinators help you create more specific and targeted styles, making your web design more efficient and organized.

Types of Combinators:

1. Descendant Combinator (space):
   Definition: Selects all elements that are descendants of a specified element, regardless of how deep they are nested.
   Example:

```
div p {
  color: blue;
}
```

   Use Case: Applying styles to all <p> elements that are descendants of a .

2. Child Combinator (>):
   Definition: Select all direct children of a specified element.
   Example:

```
ul > li {
  list-style-type: square;
}
```

   Use Case: Styling only the immediate <li> children of a <ul>.

3. Adjacent Sibling Combinator (+):
   Definition: Selects an element that is immediately preceded by a specified element.
   Example:

```
h2 + p {
  font-style: italic;
}
```

   Use Case: Applying styles to an <p> element that directly follows an <h2>.

4. General Sibling Combinator (~):
   Definition: Selects all sibling elements that share the same parent with a specified element.
   Example:

```
h3 ~ p {
  margin-left: 20px;
}
```

   Use Case: Styling all <p> elements that are siblings of an <h3>.

CSS Attribute Selectors:

CSS attribute selectors provide a powerful way to select and style HTML elements based on their attributes and attribute values. This flexibility allows for more targeted styling, improving the precision and customisation of your web designs.

1. Basic Attribute Selector:

Definition: Selects elements based on the presence of a specified attribute, regardless of its value.

Example:

```
[target] {
  color: blue;
}
```

Use Case: Applying styles to all elements with a target attribute.

2. Attribute and Value Selector:

Definition: Selects elements with a specific attribute value.
Example:

```
input[type="text"] {
  border: 1px solid #ccc;
}
```

Use Case: Styling text input elements specifically.

3. Partial Value Matching:

Definition: Selects elements with attributes containing a specific value or a value that starts, ends, or includes a given substring.
Examples:

```
a[href^="https"] {
  color: green;
}
```

```
img[alt$="icon"] {
  width: 20px;
}
```

```
input[name*="user"] {
  background-color: #f9f9f9;
}
```

Use Cases:

- Selecting links with HTTPS URLs.
- Styling images with alt attributes ending in "icon".
- Applying styles to form elements with names containing "user".

4. Negation Attribute Selector:

Definition: Selects elements that do not match a specified attribute or attribute value.
Example:

```
p:not([class="important"]) {
  color: gray;
}
```

Use Case: Styling paragraphs that do not have the class "important".

CSS Pseudo-classes:

CSS pseudo-classes are selectors that allow you to style elements based on their state or position within the document. These classes provide dynamic styling, enabling developers to create interactive and responsive web designs.

1. :hover Pseudo-class:

Definition: Applies styles to an element when the user hovers over it.
HTML:

```
<a href="https://example.com">Hover me!</a>
```

CSS:

```
a:hover {
  color: #ff0000;
  text-decoration: underline;
}
```

2. :active Pseudo-class:

Definition: Applies styles to an element while it is being activated (clicked).
HTML:

```
<button>Click me!</button>
```

CSS:

```
button:active {
  background-color: #4CAF50;
  color: white;
}
```

3. :focus Pseudo-class:

Definition: Applies styles to an element that has keyboard focus.
HTML:

```
<input type="text" placeholder="Type something" id="myInput">
```

CSS:

```
#myInput:focus {
  border: 2px solid #4CAF50;
}
```

4. :nth-child() Pseudo-class:

Definition: Select elements based on their position within a parent.

- Select Even and Odd Rows:
  HTML:

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

CSS:

```
li:nth-child(even) {
  background-color: #f2f2f2;
}
```

- Select Every Third List Item:
  HTML:

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
</ul>
```

CSS:

```
li:nth-child(3n) {
  color: #008080;
}
```

5. :not() Pseudo-class:

Definition: Selects elements that do not match a specified selector.
HTML:

```
<p class="normal">Normal paragraph</p>
<p class="important">Important paragraph</p>
```

CSS:

```
p:not(.important) {
  color: gray;
}
```

CSS Pseudo-elements

CSS pseudo-elements allow developers to style specific parts of an element's content or generate additional content dynamically. These elements provide fine-grained control over the appearance of a document, enhancing both structure and style.

1. ::before Pseudo-element:

Definition: Inserts content before the actual content of an element.
HTML:

```
<div class="quote">This is a quote.</div>
```

CSS:

```
.quote::before {
  content: """;
  color: #888;
  font-size: 1.5em;
}
```

2. ::after Pseudo-element:

Definition: Inserts content after the actual content of an element.
HTML:

```
<p>This is a paragraph.</p>
```

CSS:

```
p::after {
  content: " - Author";
  font-style: italic;
  color: #555;
}
```

3. ::first-line Pseudo-element:

Definition: Styles the first line of text within an element.
HTML:

```
<p>This is the first line of a paragraph that extends to
the next lines.</p>
```

CSS:

```
p::first-line {
  font-weight: bold;
  font-size: 1.2em;
}
```

4. ::first-letter Pseudo-element:

Definition: Styles the first letter of text within an element.
HTML:

```
<p>This is a paragraph.</p>
```

CSS:

```
p::first-letter {
  font-size: 2em;
  color: #4CAF50;
}
```

5. :: selection Pseudo-element:

Definition: Styles the portion of text that is selected by the user.
CSS:

```
::selection {
  background-color: #ffcc00;
  color: #333;
}
```

CSS Filter, Transparency, and Opacity

Filter Property:

The filter property in CSS allows you to apply visual effects to elements, creating various artistic and functional styles. Filters can enhance images, adjust colors, and blur or sharpen content.

Common Filter Functions:

- Grayscale:
  CSS:

```
.grayscale {
  filter: grayscale(100%);
}
```

  Use Case: Creating a grayscale version of an image.

- Blur:
  CSS:

```
.blurred {
  filter: blur(5px);
}
```

  Use Case: Applying a blur effect to an element.

- Brightness:
  CSS:

```
.bright {
  filter: brightness(150%);
}
```

  Use Case: Increasing the brightness of an image.

- Contrast:
  CSS:

```
.high-contrast {
  filter: contrast(200%);
}
```

  Use Case: Enhancing the contrast of an element.

- Hue-rotate:
  CSS:

```
.hue-rotated {
  filter: hue-rotate(90deg);
}
```

  Use Case: Rotating the hues of an image.

- Invert:
  CSS:

```
.inverted {
  filter: invert(100%);
}
```

  Use Case: Inverting the colors of an element.

- Saturate:
  CSS:

```
.saturated {
  filter: saturate(200%);
}
```

  Use Case: Increasing the saturation of an image.

- Sepia:
  CSS:

```
.sepia {
  filter: sepia(100%);
}
```

Use Case: Applying a sepia tone to an element.

Transparency and Opacity:

Transparency (rgba()):

Transparency in CSS is achieved using the rgba() function, where the 'a' stands for alpha, representing the level of opacity. The alpha value ranges from 0 (completely transparent) to 1 (completely opaque).

CSS:

```
.transparent-background {
  background-color: rgba(255, 0, 0, 0.5); /* Semi-transparent red background */
}
```

Opacity Property:

The opacity property sets the transparency level of an element and its children. It takes a value between 0 (completely transparent) and 1 (completely opaque).

CSS:

```
.half-opacity {
  opacity: 0.5;
}
```

CSS Gradients

CSS gradients allow you to create smooth transitions between two or more colors, enhancing the visual appeal of elements on your webpage. Gradients are versatile and can be applied to backgrounds, borders, and text, providing flexibility in design.

1. Linear Gradients:

Basic Linear Gradient:
CSS:

```
.linear-gradient {
  background: linear-gradient(to right, #ff8c00, #ffd700);
}
```

Use Case: Creating a linear gradient from orange to gold.

2. Diagonal Linear Gradient:

CSS:

```
.diagonal-gradient {
  background: linear-gradient(to bottom right, #4caf50, #2196f3);
}
```

Use Case: Applying a diagonal gradient from green to blue.

3. Radial Gradients:

Basic Radial Gradient:
CSS:

```
.radial-gradient {
  background: radial-gradient(circle, #ff4081, #7b1fa2);
}
```

Use Case: Creating a radial gradient from pink to purple.

4. Elliptical Radial Gradient:

CSS:

```
.elliptical-gradient {
  background: radial-gradient(ellipse at center, #ff5722,
#e91e63);
}
```

Use Case: Applying an elliptical gradient from deep orange to pink.

Conclusion

In conclusion, mastering the fundamentals of CSS, including combinators, attribute selectors, pseudo-classes, pseudo-elements, filters, transparency, opacity, and gradients, equips web developers with a robust set of tools for creating visually appealing and interactive websites.

CSS combinators offer precise control over styling by selecting elements based on their relationships, making web design more efficient and organized. Attribute selectors enhance customization by allowing styles to be applied based on specific attribute values, contributing to a more targeted design.

Pseudo-classes and pseudo-elements add dynamic styling options, enabling developers to create interactive and responsive web designs. These features, such as :hover and ::before, provide fine-grained control over the appearance of elements, enhancing both structure and style.

Filters, transparency, and opacity properties introduce visual effects, from grayscale images to blurred backgrounds, providing developers with creative tools for enhancing aesthetics. Finally, gradients add depth and dimension to web design, allowing for smooth transitions between colors in various elements.

By combining and utilizing these CSS features effectively, developers can create engaging, visually appealing, and user-friendly websites that align with modern design trends and user expectations.

References

1. https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors/Combinators
2. https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors/Attribute_selectors
3. https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors/Pseudo-classes_and_pseudo-elements
4. https://developer.mozilla.org/en-US/docs/Web/CSS/filter-function/opacity
5. https://developer.mozilla.org/en-US/docs/Web/CSS/opacity
6. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_images/Using_CSS_gradients