

Selects all elements that are siblings of a specified element.

html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>General Sibling Selector Example</title>
  <style>
    h1 ~ p {
      color: orange; /* All <p> elements that are siblings of an <h1> will have
orange text */
    }
  </style>
</head>
<body>
  <h1>Heading 1</h1>
  <p>This paragraph is a sibling of the heading and will be orange.</p>
  <p>This paragraph is also a sibling of the heading and will be orange.</p>
</body>
</html>
```

These examples demonstrate how to use various CSS selectors to style specific elements or groups of elements on a webpage.



values, colors, keyword colors, RGB, HSL, absolute length, relative length, percentages, EM, inline display



Sure, let's dive deeper into each of these CSS concepts.

Values

In CSS, values are the specifications used for properties. Values can be numbers, lengths, colors, strings, URLs, etc. For example, `width`, `color`, and `font-size` are CSS properties that accept different types of values:

CSS

```
width: 100px; /* Length value */
color: red; /* Color value */
font-size: 16px; /* Font size value */
```

Colors

Colors in CSS can be defined in different ways:

Keyword Colors

These are predefined color names. CSS supports 140 standard color names.

CSS

```
color: red;  
background-color: black;  
border-color: green;
```

HEX Colors

Colors are defined using hexadecimal values. It consists of a `#` followed by either three or six hexadecimal digits.

CSS

```
color: #ff0000; /* Red */  
background-color: #00ff00; /* Green */  
border-color: #0000ff; /* Blue */
```

RGB Colors

Colors are defined using the RGB model (Red, Green, Blue).

CSS

```
color: rgb(255, 0, 0); /* Red */  
background-color: rgb(0, 255, 0); /* Green */  
border-color: rgb(0, 0, 255); /* Blue */
```

RGBA Colors

Similar to RGB but includes an alpha channel (opacity).

CSS

```
color: rgba(255, 0, 0, 0.5); /* Red with 50% opacity */  
background-color: rgba(0, 255, 0, 0.3); /* Green with 30% opacity */  
border-color: rgba(0, 0, 255, 0.7); /* Blue with 70% opacity */
```

HSL Colors

Colors are defined using the HSL model (Hue, Saturation, Lightness).

CSS

```
color: hsl(0, 100%, 50%); /* Red */  
background-color: hsl(120, 100%, 50%); /* Green */
```

```
border-color: hsl(240, 100%, 50%); /* Blue */
```

HSLA Colors

Similar to HSL but includes an alpha channel (opacity).

CSS

```
color: hsla(0, 100%, 50%, 0.5); /* Red with 50% opacity */
background-color: hsla(120, 100%, 50%, 0.3); /* Green with 30% opacity */
border-color: hsla(240, 100%, 50%, 0.7); /* Blue with 70% opacity */
```

Absolute Length Units

Absolute length units are fixed and do not change according to the context. Common absolute units:

- ``px`` (pixels): A pixel is a dot on the screen (1px = 1/96th of an inch).
- ``pt`` (points): 1 point = 1/72 of an inch.
- ``cm`` (centimeters): Actual centimeters.
- ``mm`` (millimeters): Actual millimeters.
- ``in`` (inches): Actual inches.
- ``pc`` (picas): 1 pica = 12 points.

Example:

CSS

```
div {
  width: 100px; /* 100 pixels */
  height: 2in; /* 2 inches */
  font-size: 12pt; /* 12 points */
}
```

Relative Length Units

Relative length units are relative to other lengths:

- ``em``: Relative to the font-size of the element (or its parent if used for other properties).
- ``rem``: Relative to the font-size of the root element.
- ``%``: Percentage relative to the parent element.
- ``vw``: Relative to 1% of the viewport's width.
- ``vh``: Relative to 1% of the viewport's height.
- ``vmin``: Relative to 1% of the viewport's smaller dimension.
- ``vmax``: Relative to 1% of the viewport's larger dimension.

Example:

CSS

```
body {
  font-size: 16px;
}

div {
  width: 50%; /* 50% of the parent element's width */
  height: 10em; /* 10 times the font-size of the parent */
  margin: 2rem; /* 2 times the root element's font-size */
  padding: 5vw; /* 5% of the viewport's width */
}
```

Percentages

Percentages are used to create fluid, responsive layouts that adjust to the size of the parent element.

CSS

```
.container {
  width: 80%; /* 80% of the parent element's width */
  height: 50%; /* 50% of the parent element's height */
}
```

`em` and `rem` Units

- ``em``: Relative to the font-size of the element itself or its parent element.
- ``rem``: Relative to the font-size of the root element.

Example:

CSS

```
body {
  font-size: 16px;
}

.container {
  font-size: 1.5em; /* 1.5 times the font-size of the parent element */
}

.text {
  font-size: 2rem; /* 2 times the root element's font-size (32px) */
}
```

Inline Display

The ``display`` property specifies the display behavior of an element.

- ``display: inline;``: The element does not start on a new line and only takes up as much width as necessary.
- ``display: block;``: The element starts on a new line and takes up the full width available.
- ``display: inline-block;``: The element is inline like inline elements but respects width and height properties.

- `display: none;`: The element is not displayed and does not take up any space.

Example:

CSS

```
.inline {
  display: inline;
}

.block {
  display: block;
}

.inline-block {
  display: inline-block;
}

.hidden {
  display: none;
}
```

Example Putting It All Together

Here's a comprehensive example combining many of these concepts:

html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Example</title>
<style>
  body {
    font-size: 16px; /* Base font size */
    color: #333; /* Default text color */
  }

  .container {
    width: 80%;
    margin: 0 auto;
  }

  .header {
    background-color: #f8f9fa;
    padding: 1em;
    text-align: center;
    border-bottom: 1px solid #ccc;
  }

  .content {
    display: flex;
    justify-content: space-between;
    margin-top: 1em;
  }

  .content .box {
    width: 30%;
    padding: 1em;
    background-color: hsl(210, 50%, 90%);
    border: 1px solid #ddd;
    box-sizing: border-box;
  }
</style>
</html>
```

```

    .footer {
      text-align: center;
      padding: 1em;
      font-size: 0.9rem;
      color: #666;
    }

    .highlight {
      color: red;
    }

    .hidden {
      display: none;
    }

    .big-text {
      font-size: 2em; /* 2 times the size of the parent font */
    }

    .root-relative {
      font-size: 2rem; /* 2 times the size of the root font */
    }

    .percentage-width {
      width: 50%; /* 50% of the parent element's width */
    }
  </style>
</head>
<body>

<div class="container">
  <div class="header">
    <h1 class="big-text">Welcome to My Website</h1>
  </div>
  <div class="content">
    <div class="box">Content Box 1</div>
    <div class="box highlight">Content Box 2</div>
    <div class="box">Content Box 3</div>
  </div>
  <div class="footer root-relative">
    &copy; 2024 My Website
  </div>
</div>

<p class="percentage-width">This paragraph takes up 50% of the container's width.</p>
<p class="hidden">This paragraph is hidden.</p>

</body>
</html>

```

Explanation of the Example

- **Base Font Size:** The base font size for the body is set to 16px.
- **Container:** A container with 80% width, centered horizontally.
- **Header:** A header with a light background color, padding, centered text, and a bottom border.
- **Content:** A flex container with three boxes, each taking up 30% of the width, with padding and a light background color.
- **Footer:** A footer with centered text, smaller font size, and gray color.
- **Highlight Class:** Changes the text color to red.
- **Hidden Class:** Hides the element.
- **Big Text Class:** Doubles the font size.
- **Root Relative Class:** Doubles the root element's font size.
- **Percentage Width Class:** Takes up 50% of the container's width.

This example uses various CSS properties, selectors, units, and colors to create a simple but structured layout.

ChatGPT can make mistakes. Check important info.