* **how to add comments on css?**

A CSS comment is placed inside the <style> element, and starts with /\* and ends with \*/:

/\* This is  
a multi-line  
comment \*/  
  
p {  
  color: red;  
}

From the HTML tutorial, you learned that you can add comments to your HTML source by using the <!--...--> syntax.

<!-- These paragraphs will be red -->  
<p>Hello World!</p>

* **Why do we use pseudo-class?**

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

* Style an element when a user mouses over it
* Style visited and unvisited links differently
* Style an element when it gets focus

The syntax of pseudo-classes:

* selector:pseudo-class {  
    property: value;  
  }

a:link {  
  color: #FF0000;  
}  
  
/\* visited link \*/  
a:visited {  
  color: #00FF00;  
}  
  
/\* mouse over link \*/  
a:hover {  
  color: #FF00FF;  
}  
  
/\* selected link \*/  
a:active {  
  color: #0000FF;  
}

* **How is specificity applied?**
* If there are two or more CSS rules that point to the same element, the selector with the highest specificity value will "win", and its style declaration will be applied to that HTML element.
* Think of specificity as a score/rank that determines which style declaration are ultimately applied to an element.

There are four categories which define the specificity level of a selector:

* **Inline styles** - Example: <h1 style="color: pink;">
* **IDs** - Example: #navbar
* **Classes, pseudo-classes, attribute selectors** - Example: .test, :hover, [href]
* **Elements and pseudo-elements** - Example: h1, :before

|  |  |  |
| --- | --- | --- |
| **Selector** | **Specificity Value** | **Calculation** |
| p | 1 | 1 |
| p.test | 11 | 1 + 10 |
| p#demo | 101 | 1 + 100 |
| <p style="color: pink;"> | 1000 | 1000 |
| #demo | 100 | 100 |
| .test | 10 | 10 |
| p.test1.test2 | 21 | 1 + 10 + 10 |
| #navbar p#demo | 201 | 100 + 1 + 100 |
| \* | 0 | 0 (the universal selector is ignored) |

* What method allows an element to be moved from its current position?

The **translate()** [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) [function](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Functions) repositions an element in the horizontal and/or vertical directions. Its result is a [<transform-function>](https://developer.mozilla.org/en-US/docs/Web/CSS/transform-function) data type.

* what properties does flex model have?
  + The flex property sets the flexible length on flexible items.
  + The flex-basis property specifies the initial length of a flexible item.
  + The flex-direction property specifies the direction of the flexible items.

The flex-flow property is a shorthand property for:

* [flex-direction](https://www.w3schools.com/cssref/css3_pr_flex-direction.asp)
* [flex-wrap](https://www.w3schools.com/cssref/css3_pr_flex-wrap.asp)
  + The flex-grow property specifies how much the item will grow relative to the rest of the flexible items inside the same container.
  + The flex-shrink property specifies how the item will shrink relative to the rest of the flexible items inside the same container.
  + The flex-wrap property specifies whether the flexible items should wrap or not.
  + **What is the difference between flex and grids?**

The basic difference between CSS Grid Layout and [CSS Flexbox Layout](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout) is that flexbox was designed for layout in one dimension - either a row or a column. Grid was designed for two-dimensional layout - rows, and columns at the same time.

**Alignment:**

* Flex Direction allows developers to align elements vertically or horizontally, which is used when developers create and reverse rows or columns.
* CSS Grid deploys fractional measure units for grid fluidity and auto-keyword functionality to automatically adjust columns or rows.
* **Give an example where we have to use grids and where you have to use flexbox?**

Not possible.

* **Give an example where you cannot use flexbox, and you can only use grids?**

<!DOCTYPE html>

<html>

<head>

<style>

.grid-container {

display: grid;

grid-template-columns: auto auto auto;

background-color: #2196F3;

padding: 10px;

}

.grid-item {

background-color: rgba(255, 255, 255, 0.8);

border: 1px solid rgba(0, 0, 0, 0.8);

padding: 20px;

font-size: 30px;

text-align: center;

}

</style>

</head>

<body>

<h1>display: grid</h1>

<p>Use display: grid; to make a block-level grid container:</p>

<div class="grid-container">

<div class="grid-item">1</div>

<div class="grid-item">2</div>

<div class="grid-item">3</div>

<div class="grid-item">4</div>

<div class="grid-item">5</div>

<div class="grid-item">6</div>

<div class="grid-item">7</div>

<div class="grid-item">8</div>

<div class="grid-item">9</div>

</div>

</body>

</html>

* **What are combinators? give examples of how you can use them**

A combinator is something that explains the relationship between the selectors.

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS:

* descendant selector (space)
* child selector (>)
* adjacent sibling selector (+)
* general sibling selector (~)
* The descendant selector matches all elements that are descendants of a specified element.
* The following example selects all <p> elements inside <div> elements:

div p {  
  background-color: yellow;  
}

The child selector selects all elements that are the children of a specified element.

The following example selects all <p> elements that are children of a <div> element:

div > p {  
  background-color: yellow;  
}

The adjacent sibling selector is used to select an element that is directly after another specific element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

The following example selects the first <p> element that are placed immediately after <div> elements:

div + p {  
  background-color: yellow;  
}

The general sibling selector selects all elements that are next siblings of a specified element.

The following example selects all <p> elements that are next siblings of <div> elements:

div ~ p {  
  background-color: yellow;  
}

* What does object-fit do?

The object-fit property is used to specify how an <img> or <video> should be resized to fit its container.

* What does rotate do?
* The **rotate()** [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) [function](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Functions) defines a transformation that rotates an element around a fixed point on the 2D plane, without deforming it. Its result is a [<transform-function>](https://developer.mozilla.org/en-US/docs/Web/CSS/transform-function) data type.
* The fixed point that the element rotates around — mentioned above — is also known as the **transform origin**. This defaults to the center of the element, but you can set your own custom transform origin using the [transform-origin](https://developer.mozilla.org/en-US/docs/Web/CSS/transform-origin) property.

The amount of rotation created by rotate() is specified by an [<angle>](https://developer.mozilla.org/en-US/docs/Web/CSS/angle). If positive, the movement will be clockwise; if negative, it will be counter-clockwise. A rotation by 180° is called point reflection.

* **What rule can be used to define animations**
* The @keyframes rule specifies the animation code.
* The animation is created by gradually changing from one set of CSS styles to another.
* During the animation, you can change the set of CSS styles many times.
* [animation-name](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-name)
* Specifies the name of the [@keyframes](https://developer.mozilla.org/en-US/docs/Web/CSS/@keyframes) at-rule describing the animation's keyframes.
* [animation-duration](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-duration)
* Configures the length of time that an animation should take to complete one cycle.
* [animation-timing-function](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-timing-function)
* Configures the timing of the animation; that is, how the animation transitions through keyframes, by establishing acceleration curves.
* [animation-delay](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-delay)
* Configures the delay between the time the element is loaded and the beginning of the animation sequence.
* [animation-iteration-count](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-iteration-count)
* Configures the number of times the animation should repeat; you can specify infinite to repeat the animation indefinitely.
* [animation-direction](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-direction)
* Configures whether or not the animation should alternate direction on each run through the sequence or reset to the start point and repeat itself.
* [animation-fill-mode](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-fill-mode)
* Configures what values are applied by the animation before and after it is executing.
* [animation-play-state](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-play-state)
* Lets you pause and resume the animation sequence.
* When working with attribute selectors, how can you select elements which contain a particular attribute value?

The CSS **attribute selector** matches elements based on the presence or value of a given attribute.

[**Links**](https://developer.mozilla.org/en-US/docs/Web/CSS/Attribute_selectors#links)

**CSS**

a {

color: blue;

}

/\* Internal links, beginning with "#" \*/

a[href^="#"] {

background-color: gold;

}

/\* Links with "example" anywhere in the URL \*/

a[href\*="example"] {

background-color: silver;

}

#### HTML

<ul>

<li><a href="#internal">Internal link</a></li>

<li><a href="http://example.com">Example link</a></li>

<li><a href="#InSensitive">Insensitive internal link</a></li>

<li><a href="http://example.org">Example org link</a></li>

<li><a href="https://example.org">Example https org link</a></li>

</ul>

<https://developer.mozilla.org/en-US/docs/Web/CSS/Attribute_selectors>

### [Languages](https://developer.mozilla.org/en-US/docs/Web/CSS/Attribute_selectors#languages)

#### CSS

/\* All divs with a `lang` attribute are bold. \*/

div[lang] {

font-weight: bold;

}

/\* All divs without a `lang` attribute are italicized. \*/

div:not([lang]) {

font-style: italic;

}

[**HTML ordered lists**](https://developer.mozilla.org/en-US/docs/Web/CSS/Attribute_selectors#html_ordered_lists)

The HTML specification requires the [type](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#attr-type) attribute to be matched case-insensitively because it is primarily used in the [<input>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input) element. Note that if the modifiers are not supported by the user agent, then the selector will not match.

**CSS**

/\* Case-sensitivity depends on document language \*/

ol[type="a"] {

list-style-type: lower-alpha;

background: red;

}

ol[type="b" s] {

list-style-type: lower-alpha;

background: lime;

}

ol[type="B" s] {

list-style-type: upper-alpha;

background: grey;

}

ol[type="c" i] {

list-style-type: upper-alpha;

background: green;

}

Copy to Clipboard

**HTML**

<ol type="A">

<li>Red background for case-insensitive matching (default for the type selector)</li>

</ol>

<ol type="b">

<li>Lime background if `s` modifier is supported (case-sensitive match)</li>

</ol>

<ol type="B">

<li>Grey background if `s` modifier is supported (case-sensitive match)</li>

</ol>

<ol type="C">

<li>Green background if `i` modifier is supported (case-insensitive match)</li>

</ol>

* What does @media do?

The @media rule is used in media queries to **apply different styles for different media types/devices**. Media queries can be used to check many things, such as: width and height of the viewport. width and height of the device.

@media screen and (max-width: 600px) {  
  div.example {  
    display: none;  
  }  
}

* **What can be used to override properties of an element**

To override the CSS properties of a class using another class, we can use the **!** **important directive**. In CSS, ! important means “this is important”, and the property:value pair that has this directive is always applied even if the other element has higher specificity

* **How can you select every alternate elements in a list of elements using css?**

li:nth-child(even) { /\* Selects only even elements \*/

color: green;

}

* **What is the ranking of selectors with respect to specificity**

**Inline styles** - Example: <h1 style="color: pink;">

**IDs** - Example: #navbar

**Classes, pseudo-classes, attribute selectors** - Example: .test, :hover, [href]

**Elements and pseudo-elements** - Example: h1, :before

* **how can we apply same styles to multiple selectors**?

CSS allows you to group multiple selectors that share the same declaration. This optimization technique allows you to apply the same style to multiple elements to save space.

So this:

div#main {border:1px solid red;}

div#sidebar {border:1px solid red;}

Becomes this:

dive#main, div#sidebar {

border:1px solid red;

}

You can combine grouped selectors with contextual and other selectors to create powerful yet compact rules for your style sheets. The body id/class method used to highlight current tabs is one example of this technique.

* **What are the differences between relative and absolute in CSS?**

1. An element with position: relative; is positioned relative to its normal position.
2. Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

* An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
* However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.