**Pseudo-Classes**

Pseudo-classes are powerful in CSS, allowing you to style elements based on their state or position. Here’s an introduction to some commonly used pseudo-classes:

**1. Basic Interaction States**

* **:hover**: Applies styles when the user hovers over an element.

|  |
| --- |
| .  button*:hover* {  **background-color**: *lightblue*;  }  . |

* **:active**: Applies styles when an element is actively being clicked.
* **:focus**: Targets elements (like inputs) when they’re focused, typically by clicking or tabbing.

**2. Structural Pseudo-Classes**

* **:first-child**: Targets the first child of a parent.
* **:last-child**: Targets the last child.
* **:nth-child(n)**: Targets elements based on their position in the parent, where n can be a number, odd, even, or a formula (e.g., 2n+1).

|  |
| --- |
| .  li*:nth-child*(2n) {  **color**: *blue*;  }  . |

**3. Form-Related Pseudo-Classes**

* **:checked**: Targets checked checkboxes or radio buttons.
* **:disabled**: Targets disabled form fields.
* **:not(selector)**: Targets elements that do not match a given selector.
* **:placeholder-shown**: Targets inputs and textareas with a visible placeholder.

**4. Content-Based Pseudo-Classes**

* **:empty**: Selects elements with no children (including text nodes).
* **:first-of-type** / **:last-of-type**: Targets the first/last child of a specific type within a parent.

**Example Summary**

|  |
| --- |
| .  input*:focus* {  **border-color**: *green*;  }  p*:nth-child*(*odd*) {  **color**: *red*;  }  input*:not*(:placeholder-shown) {  **border**: 2px *solid* *blue*;  }  . |

These pseudo-classes allow for dynamic and conditional styling. Let me know if you’d like to explore any of these in more detail or try some examples!

CSS pseudo-classes are more extensive, especially with modern CSS advancements. Here’s a more comprehensive list to cover different pseudo-classes and their uses:

### Common Pseudo-Classes by Category

#### ****1. User Interaction****

* **:hover**: Applies styles when an element is hovered over.
* **:focus**: Applies styles when an element is focused.
* **:active**: Styles an element while it’s being clicked.
* **:visited**: Styles links that have been visited.
* **:link**: Styles links that have not been visited.
* **:focus-within**: Applies styles to an element when any child element has focus.
* **:focus-visible**: Styles elements when they’re focused with keyboard navigation, enhancing accessibility.

#### ****2. Structural and Positional****

* **:first-child** / **:last-child**: Targets the first or last child of a parent element.
* **:nth-child(n)**: Styles elements based on position (e.g., nth-child(2n) for even, 2n+1 for odd).
* **:nth-of-type(n)**: Like nth-child, but only counts elements of the same type.
* **:first-of-type** / **:last-of-type**: Targets the first or last occurrence of an element of a particular type.
* **:only-child**: Applies styles to elements that are the only child of their parent.
* **:only-of-type**: Applies styles if it’s the only element of its type within a parent.

#### ****3. Form and Input-Related****

* **:checked**: Targets checkboxes or radio buttons that are checked.
* **:disabled** / **:enabled**: Styles form elements based on whether they’re disabled or enabled.
* **:required** / **:optional**: Applies styles to inputs based on the required attribute.
* **:valid** / **:invalid**: Styles form elements based on validity.
* **:in-range** / **:out-of-range**: Styles input elements with a defined range.
* **:placeholder-shown**: Styles inputs when placeholder text is visible.
* **:read-only** / **:read-write**: Applies styles to form elements based on whether they’re read-only or writable.

#### ****4. Content and State****

* **:empty**: Targets elements that contain no child elements or text.
* **:not(selector)**: Applies styles to elements that do not match a particular selector.
* **:root**: Targets the root element of the document (<html>), useful for setting CSS variables.
* **:target**: Styles elements that match a URL fragment identifier (e.g., when a specific section of a page is targeted by an anchor link).

#### ****5. UI State and Accessibility****

* **:checked** / **:indeterminate**: Styles elements in a checkbox that are checked or in an indeterminate state.
* **:has()**: Selects elements that contain elements matching a specific selector. This powerful pseudo-class is supported only in newer browsers.
* **:is() and :where()**: Applies a single style rule to multiple selectors, enhancing readability and simplifying complex selectors. :where() has no specificity.

css

Copy code

button:is(:hover, :focus) {

background-color: lightblue;

}

* **:lang(language)**: Styles elements based on their language attribute.

### Summary of Popular CSS Pseudo-Classes

This list breaks down pseudo-classes by their primary use cases: **user interaction** (:hover, :focus), **structural positioning** (:first-child, :nth-of-type), **form control** (:checked, :disabled), **content state** (:empty, :not), and advanced UI and accessibility controls (:has, :lang).

Let me know if you’d like examples on any specific pseudo-class!

CSS pseudo-elements allow you to style specific parts of an element without adding extra HTML. They are written with :: before the name (e.g., ::before), although some browsers still support the single : syntax. Here are some of the most commonly used pseudo-elements:

### 1. ::before ****and**** ::after

* Used to insert content before or after an element’s content. Often used with content property to add extra styling, icons, or decorative elements.
* Example:

|  |
| --- |
| .  p*::before* {  **content**: "→ ";  **color**: *blue*;  }  p*::after* {  **content**: " ←";  **color**: *red*;  }  . |

### 2. ::first-line

* Styles only the first line of text within an element. Useful for making the first line of a paragraph bold, a different color, etc.
* Example:

|  |
| --- |
| .  p*::first-line* {  **font-weight**: *bold*;  **color**: *green*;  }  . |

### 3. ::first-letter

* Styles only the first letter of a block of text. Often used to create drop-cap effects (like in traditional print).
* Example:

|  |
| --- |
| .  p*::first-letter* {  **font-size**: 2*em*;  **color**: *orange*;  }  . |

### 4. ::selection

* Applies styles to the portion of text highlighted by the user (e.g., when they click and drag to select text).
* Example:

|  |
| --- |
| .  *::selection* {  **background-color**: *yellow*;  **color**: *black*;  }  . |

### 5. ::marker

* Styles the bullet or number marker of a list item. It’s useful when customizing list styles without affecting the content.
* Example:

|  |
| --- |
| .  li*::marker* {  **color**: *purple*;  **font-size**: 1.5*em*;  }  . |

### Usage Example Summary

|  |
| --- |
| .  */\**  *html*  *<p>Learning CSS pseudo-elements is fun!</p>*  *\*/*  p*::before* {  **content**: "💡 ";  }  p*::first-letter* {  **font-size**: 2*em*;  **color**: *coral*;  }  p*::selection* {  **background**: *pink*;  }  . |

### Quick Summary

* **::before** and **::after**: Insert content around elements.
* **::first-line** and **::first-letter**: Style specific parts of text.
* **::selection**: Style text selection.
* **::marker**: Style list markers.

Let me know if you'd like to dive deeper into any of these or see more examples!

These are the most widely used CSS pseudo-elements, but there are a few more specialized ones that can be useful in certain scenarios. Here are some additional pseudo-elements:

### Additional CSS Pseudo-Elements

1. **::backdrop**:
   * Used with elements like <dialog> or fullscreen elements. It styles the background (or "backdrop") behind an element, such as a modal dialog, when it’s displayed.
   * Example:

|  |
| --- |
| .  dialog*::backdrop* {  **background-color**: rgba(0, 0, 0, 0.8);  }  . |

1. **::placeholder**:
   * Styles the placeholder text inside input fields (works on <input> and <textarea> elements).
   * Example:

|  |
| --- |
| .  input*::placeholder* {  **color**: *gray*;  **font-style**: *italic*;  }  . |

1. **::file-selector-button**:
   * Targets the button in <input type="file"> elements, allowing you to style the button that triggers the file upload dialog.
   * Example:

|  |
| --- |
| .  input[*type*="file"]::file-selector-button {  **background-color**: #4CAF50;  **color**: *white*;  **border**: *none*;  **padding**: 5px 10px;  **cursor**: *pointer*;  }  . |

1. **::cue (for WebVTT captions)**:
   * Used in styling WebVTT subtitles or captions in media elements like <video>.
   * Example:

|  |
| --- |
| .  ::cue {  **color**: *yellow*;  **font-size**: 1.2*em*;  }  . |

1. **::part() and ::slotted()** (for Web Components):
   * **::part()**: Used in styling shadow DOM elements that are exposed through the part attribute in Web Components.
   * **::slotted()**: Used to style elements slotted into a shadow DOM slot.
   * Example:

|  |
| --- |
| .  */\* Styling a part exposed in shadow DOM \*/*  custom-element::part(button) {  **color**: *blue*;  }  */\* Styling a slotted element \*/*  ::slotted(span) {  **color**: *green*;  }  . |

### Summary

To recap, here’s a categorized list:

* **Text styling**: ::first-line, ::first-letter, ::selection
* **Content addition**: ::before, ::after
* **Interactive elements**: ::backdrop, ::file-selector-button, ::placeholder
* **Lists**: ::marker
* **Media**: ::cue
* **Web Components**: ::part(), ::slotted()

These cover most styling needs across text, content, forms, interactive elements, and custom components. Let me know if you’d like to explore any of these in detail!