

THE ULTIMATE WEB DEV JOURNEY

STEP 5

CSS Selectors & Specificity

Becoming a CSS Targeting Ninja!

Precision Styling Mastery

Your Progress:

Step 1: HTML Basics *(Complete)*

Step 2: Links, Images & Lists *(Complete)*

Step 3: CSS Basics *(Complete)*

Step 4: CSS Box Model *(Complete)*

Step 5: CSS Selectors & Specificity *(Current)*

Step 6: CSS Positioning *(Locked)*

What You'll Master Today

By the end of this step, you'll be able to:

- Target ANY element with surgical precision
- Use advanced selectors like a pro
- Master pseudo-classes for interactive effects
- Understand CSS specificity wars
- Create hover effects and dynamic styling

What Are CSS Selectors?

The Targeting System

Imagine you're a sniper in a video game

You have different ways to target enemies:

- **Basic selector** = "Shoot the guy in the red shirt" (element)
- **Class selector** = "Shoot all soldiers" (.class)
- **ID selector** = "Shoot the commander" (#id)
- **Advanced selector** = "Shoot the soldier standing next to the tank" (combinator)

CSS selectors are how you **TARGET** which HTML elements to style!

Types of Selectors

1. Basic Selectors You Already Know

Quick Recap

```
1 /* Element Selector */
2 p {
3     color: blue;
4 }
5
6 /* Class Selector */
7 .highlight {
8     background: yellow;
9 }
10
11 /* ID Selector */
12 #header {
13     font-size: 24px;
14 }
```

You already learned these in Step 3! Now let's go DEEPER!

2. Universal Selector

The "Select Everything" Button

The **asterisk (*)** selects EVERY element on the page!

```
1 /* Targets EVERY element */
2 * {
3     margin: 0;
4     padding: 0;
5     box-sizing: border-box;
6 }
```

When to Use It

Use sparingly! It's powerful but can slow down your page.

Best use: Resetting default browser styles at the start of your CSS.

3. Combinator Selectors (The Power Moves!)

Relationship-Based Targeting

Combinators let you select elements based on their **relationship** with other elements!

A) Descendant Selector (Space)

Select Children

Selects ALL elements inside another element (at any level).

```
1 <div class="container">
2   <p>I will be styled</p>
3   <div>
4     <p>I will also be styled!</p>
5   </div>
6 </div>
```

```
1 /* Targets ALL <p> inside .container (any level deep) */
2 .container p {
3   color: purple;
4   font-weight: bold;
5 }
```

Think: "All paragraphs ANYWHERE inside .container"

B) Child Selector (>)

Direct Children Only

Selects only **DIRECT** children, not grandchildren!

```
1 <div class="parent">
2   <p>I'm a direct child - I'll be styled</p>
3   <div>
4     <p>I'm a grandchild - I WON'T be styled</p>
5   </div>
6 </div>
```

```
1 /* Only direct <p> children of .parent */
2 .parent > p {
3   color: red;
4 }
```

Think: "Only my kids, not my grandkids"

C) Adjacent Sibling Selector (+)

The Element Right Next Door

Selects the element that comes **immediately after** another element.

```
1 <h2>Main Heading</h2>
2 <p>This paragraph will be styled</p>
3 <p>This one won't</p>
```

```

1  /* The <p> that comes RIGHT AFTER <h2> */
2  h2 + p {
3      font-size: 20px;
4      color: orange;
5  }

```

Think: "My immediate younger sibling"

D) General Sibling Selector (~)

All Siblings After

Selects ALL sibling elements that come after.

```

1  <h2>Heading</h2>
2  <p>Styled!</p>
3  <p>Also styled!</p>
4  <div>Not styled (different element)</div>
5  <p>Still styled!</p>

```

```

1  /* ALL <p> elements that come after <h2> */
2  h2 ~ p {
3      color: green;
4  }

```

Think: "All my younger siblings of the same type"

Attribute Selectors (Super Powers!)

Target Elements by Their Attributes

You can select elements based on their **attributes** (like href, type, class, etc.)!

Attribute Selector Syntax

```

1  /* Has the attribute */
2  [type] {
3      border: 2px solid blue;
4  }
5
6  /* Exact match */
7  [type="text"] {
8      background: lightyellow;
9  }
10
11 /* Starts with */
12 [href^="https"] {
13     color: green; /* Secure links in green */
14 }
15

```

```

16 /* Ends with */
17 [href$=".pdf"] {
18     color: red; /* PDF links in red */
19 }
20
21 /* Contains */
22 [class*="btn"] {
23     padding: 10px; /* Any class with "btn" in it */
24 }

```

Real-World Example

```

1 /* Style all external links differently */
2 a[href^="http"] {
3     color: blue;
4 }
5
6 /* Style email links */
7 a[href^="mailto"] {
8     color: purple;
9 }
10
11 /* Style file download links */
12 a[href$=".zip"],
13 a[href$=".pdf"] {
14     font-weight: bold;
15 }

```

Pseudo-Classes (Interactive Magic!)

Special States of Elements

Pseudo-classes select elements in a **specific state** (like hovering, focused, first child, etc.)

User Action Pseudo-Classes

```

1 /* When you hover over an element */
2 button:hover {
3     background: blue;
4     transform: scale(1.1);
5     cursor: pointer;
6 }
7
8 /* When an element is clicked/focused */
9 input:focus {
10     border: 3px solid orange;
11     outline: none;
12 }

```

```

13
14 /* When a link has been visited */
15 a:visited {
16     color: purple;
17 }
18
19 /* When a link is being clicked */
20 a:active {
21     color: red;
22 }

```

Try This Cool Button Effect!

```

1 .cool-button {
2     background: linear-gradient(45deg, #667eea, #764ba2);
3     color: white;
4     padding: 15px 30px;
5     border: none;
6     border-radius: 25px;
7     transition: all 0.3s;
8 }
9
10 .cool-button:hover {
11     transform: translateY(-5px);
12     box-shadow: 0 10px 20px rgba(0,0,0,0.3);
13 }
14
15 .cool-button:active {
16     transform: translateY(-2px);
17 }

```

Structural Pseudo-Classes (Position-Based)

Target Based on Position

Select elements based on their position among siblings!

```

1 /* First child */
2 li:first-child {
3     font-weight: bold;
4     color: green;
5 }
6
7 /* Last child */
8 li:last-child {
9     color: red;
10 }
11
12 /* Every odd item (1st, 3rd, 5th...) */
13 li:nth-child(odd) {
14     background: lightgray;

```

```

15 }
16
17 /* Every even item (2nd, 4th, 6th...) */
18 li:nth-child(even) {
19     background: white;
20 }
21
22 /* Every 3rd item */
23 li:nth-child(3n) {
24     color: blue;
25 }
26
27 /* The 5th item specifically */
28 li:nth-child(5) {
29     font-size: 20px;
30 }

```

Create Zebra-Striped Tables!

```

1  /* Alternating row colors */
2  tr:nth-child(odd) {
3      background: #f9f9f9;
4  }
5
6  tr:nth-child(even) {
7      background: #ffffff;
8  }
9
10 tr:hover {
11     background: #e0f7ff;
12     transition: 0.3s;
13 }

```

Other Useful Pseudo-Classes

```

1  /* Empty elements */
2  p:empty {
3      display: none;
4  }
5
6  /* Disabled form inputs */
7  input:disabled {
8      background: lightgray;
9      cursor: not-allowed;
10 }
11
12 /* Checked checkboxes/radios */
13 input:checked {
14     transform: scale(1.2);
15 }
16

```

```

17 /* Elements with no children */
18 div:not(.special) {
19     opacity: 0.5;
20 }

```

Pseudo-Elements (Create Virtual Elements!)

Add Content Without HTML

Pseudo-elements let you style **parts** of elements or create **virtual elements**!

::before and ::after

The Most Powerful Pseudo-Elements

These insert content BEFORE or AFTER an element's content!

```

1  /* Add emoji before every heading */
2  h2::before {
3      content: " ";
4      color: orange;
5  }
6
7  /* Add arrow after links */
8  a::after {
9      content: " ";
10     color: blue;
11 }
12
13 /* Create decorative elements */
14 .quote::before {
15     content: '"';
16     font-size: 60px;
17     color: lightgray;
18     position: absolute;
19     left: -20px;
20     top: -10px;
21 }

```

Create a Beautiful Quote Card

```
1 .quote-card {
2   position: relative;
3   padding: 30px;
4   background: linear-gradient(135deg, #667eea, #764ba2);
5   color: white;
6   border-radius: 10px;
7 }
8
9 .quote-card::before {
10  content: '';
11  font-size: 80px;
12  position: absolute;
13  top: -10px;
14  left: 10px;
15  opacity: 0.3;
16 }
17
18 .quote-card::after {
19  content: '';
20  font-size: 80px;
21  position: absolute;
22  bottom: -40px;
23  right: 10px;
24  opacity: 0.3;
25 }
```

Other Pseudo-Elements

```
1 /* Style first letter */
2 p::first-letter {
3   font-size: 40px;
4   color: red;
5   float: left;
6   line-height: 1;
7 }
8
9 /* Style first line */
10 p::first-line {
11   font-weight: bold;
12   color: blue;
13 }
14
15 /* Style selected text */
16 ::selection {
17   background: yellow;
18   color: black;
19 }
```

CSS Specificity (The Battle Royale!)

CRITICAL CONCEPT

When multiple CSS rules target the same element, WHO WINS?

Answer: **SPECIFICITY!** The most specific rule wins!

The Specificity Hierarchy

The Point System

CSS assigns points to selectors:

Selector Type	Points	Example
Inline styles	1000	style="color: red"
IDs	100	#header
Classes, attributes, pseudo-classes	10	.button, [type], :hover
Elements, pseudo-elements	1	p, div, ::before
Universal selector	0	*

Calculating Specificity

```
1  /* Specificity: 1 (one element) */
2  p {
3      color: blue;
4  }
5
6  /* Specificity: 10 (one class) */
7  .text {
8      color: green;
9  }
10
11 /* Specificity: 100 (one ID) */
12 #main {
13     color: red;
14 }
15
16 /* Specificity: 11 (one class + one element) */
17 .container p {
18     color: purple;
19 }
20
21 /* Specificity: 101 (one ID + one element) */
22 #main p {
23     color: orange;
24 }
25
26 /* Specificity: 111 (one ID + one class + one element) */
27 #main .text p {
28     color: yellow;
```

The Specificity Wars

```

1  /* This paragraph will be ORANGE, not blue or green */
2  <p id="special" class="highlight">What color am I?</p>
3
4  p {
5      color: blue;          /* Specificity: 1 - LOSES */
6  }
7
8  .highlight {
9      color: green;         /* Specificity: 10 - LOSES */
10 }
11
12 #special {
13     color: orange;         /* Specificity: 100 - WINS! */
14 }

```

The Nuclear Option: !important

Use With Extreme Caution

!important overrides **ALL** specificity (except other !important rules)

```

1  /* This will be red, no matter what! */
2  p {
3      color: red !important;
4  }
5
6  #super-specific {
7      color: blue; /* LOSES to !important above */
8  }

```

DON'T ABUSE !important

Bad practice!

Using !important is like using duct tape instead of fixing the real problem.

Only use when:

- Overriding external CSS you can't edit
- Utility classes that should always win

MEGA PROJECT: Advanced Product Card

Let's Build Something Amazing!

Create an interactive product card using EVERYTHING you learned!

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Product Card with Advanced Selectors</title>
5   <style>
6     /* Reset */
7     * {
8       margin: 0;
9       padding: 0;
10      box-sizing: border-box;
11    }
12
13    body {
14      font-family: Arial, sans-serif;
15      background: linear-gradient(135deg, #667eea, #764ba2);
16      min-height: 100vh;
17      display: flex;
18      justify-content: center;
19      align-items: center;
20    }
21
22    /* Product Card */
23    .product-card {
24      background: white;
25      width: 350px;
26      border-radius: 20px;
27      overflow: hidden;
28      box-shadow: 0 20px 60px rgba(0,0,0,0.3);
29      transition: all 0.3s;
30      position: relative;
31    }
32
33    /* Hover effect on entire card */
34    .product-card:hover {
35      transform: translateY(-10px);
36      box-shadow: 0 30px 80px rgba(0,0,0,0.4);
37    }
38
39    /* Sale badge using ::before */
40    .product-card::before {
41      content: "SALE";
42      position: absolute;
43      top: 20px;
44      right: -35px;
45      background: red;
46      color: white;
47      padding: 5px 40px;
48      transform: rotate(45deg);
49      font-weight: bold;
50      z-index: 10;
51    }
52
53    /* Image container */

```

```

54     .product-card > img {
55         width: 100%;
56         height: 250px;
57         object-fit: cover;
58     }
59
60     /* Content area */
61     .product-card > .content {
62         padding: 25px;
63     }
64
65     /* Direct child heading */
66     .content > h2 {
67         font-size: 24px;
68         margin-bottom: 10px;
69         color: #333;
70     }
71
72     /* First line special styling */
73     .content > p:first-of-type::first-line {
74         font-weight: bold;
75         color: #667eea;
76     }
77
78     /* All paragraphs inside content */
79     .content p {
80         color: #666;
81         line-height: 1.6;
82         margin-bottom: 15px;
83     }
84
85     /* Price section */
86     .price-section {
87         display: flex;
88         align-items: center;
89         gap: 10px;
90         margin: 20px 0;
91     }
92
93     /* Old price */
94     .price-section > .old-price {
95         text-decoration: line-through;
96         color: #999;
97         font-size: 18px;
98     }
99
100    /* New price - adjacent sibling */
101    .old-price + .new-price {
102        font-size: 28px;
103        color: #e74c3c;
104        font-weight: bold;
105    }
106

```

```

107  /* Features list */
108  .features {
109      list-style: none;
110      margin: 20px 0;
111  }
112
113  /* All list items */
114  .features li {
115      padding: 8px 0;
116      border-bottom: 1px solid #eee;
117  }
118
119  /* Add checkmark before each feature */
120  .features li::before {
121      content: "☐";
122      color: #27ae60;
123      font-weight: bold;
124      margin-right: 8px;
125  }
126
127  /* First feature special */
128  .features li:first-child {
129      border-top: 1px solid #eee;
130  }
131
132  /* Last feature special */
133  .features li:last-child {
134      border-bottom: 2px solid #667eea;
135  }
136
137  /* Odd features background */
138  .features li:nth-child(odd) {
139      background: #f9f9f9;
140  }
141
142  /* Hover effect on list items */
143  .features li:hover {
144      background: #667eea;
145      color: white;
146      transform: translateX(5px);
147      transition: all 0.3s;
148  }
149
150  .features li:hover::before {
151      color: white;
152  }
153
154  /* Buttons container */
155  .buttons {
156      display: flex;
157      gap: 10px;
158      margin-top: 20px;
159  }

```

```

160
161     /* All buttons */
162     .buttons button {
163         flex: 1;
164         padding: 12px;
165         border: none;
166         border-radius: 8px;
167         font-weight: bold;
168         cursor: pointer;
169         transition: all 0.3s;
170     }
171
172     /* First button (Add to Cart) */
173     .buttons button:first-child {
174         background: #667eea;
175         color: white;
176     }
177
178     .buttons button:first-child:hover {
179         background: #5568d3;
180         transform: scale(1.05);
181     }
182
183     .buttons button:first-child:active {
184         transform: scale(0.98);
185     }
186
187     /* Last button (Wishlist) */
188     .buttons button:last-child {
189         background: white;
190         border: 2px solid #667eea;
191         color: #667eea;
192     }
193
194     .buttons button:last-child:hover {
195         background: #667eea;
196         color: white;
197     }
198
199     /* Disabled button state */
200     .buttons button:disabled {
201         background: #ccc;
202         cursor: not-allowed;
203         transform: none !important;
204     }
205
206     /* Selection color */
207     ::selection {
208         background: #667eea;
209         color: white;
210     }
211
212     /* Rating stars */

```

```

213     .rating {
214         margin: 15px 0;
215     }
216
217     .rating span {
218         color: #ddd;
219         font-size: 24px;
220     }
221
222     /* Filled stars (first 4) */
223     .rating span:nth-child(-n+4) {
224         color: #f1c40f;
225     }
226
227     /* Hover effect on stars */
228     .rating span:hover,
229     .rating span:hover ~ span {
230         color: #f39c12;
231     }
232 </style>
233 </head>
234 <body>
235     <div class="product-card">
236         
238
239         <div class="content">
240             <h2>Premium Wireless Headphones</h2>
241
242             <p>Experience crystal-clear audio quality with our
243                 flagship wireless headphones. Perfect for music lovers
244                 and professionals alike.</p>
245
246             <div class="rating">
247                 <span></span>
248                 <span></span>
249                 <span></span>
250                 <span></span>
251                 <span></span>
252             </div>
253
254             <div class="price-section">
255                 <span class="old-price">$199</span>
256                 <span class="new-price">$149</span>
257             </div>
258
259             <ul class="features">
260                 <li>Active Noise Cancellation</li>
261                 <li>40-Hour Battery Life</li>
262                 <li>Premium Leather Cushions</li>
263                 <li>Bluetooth 5.0 Connectivity</li>
264                 <li>Built-in Microphone</li>
265             </ul>

```

```

263
264         <div class="buttons">
265             <button>Add to Cart</button>
266             <button>Wishlist</button>
267         </div>
268     </div>
269 </div>
270 </body>
271 </html>

```

What's Happening Here?

Let's break down the advanced selectors used:

1. * - Universal reset
2. .product-card > img - Direct child selector
3. .content > h2 - Direct child heading
4. ::before and ::after - Sale badge and checkmarks
5. :first-child, :last-child - Special list items
6. :nth-child(odd) - Zebra striping
7. :hover, :active - Interactive states
8. .old-price + .new-price - Adjacent sibling
9. ::first-line - Special paragraph styling
10. ::selection - Custom text selection color

Common Mistakes & How to Avoid Them

Mistake #1: Forgetting the Space

```

1  /* WRONG - No space, means element with BOTH classes */
2  .container.box {
3      /* Targets <div class="container box"> */
4  }
5
6  /* RIGHT - With space, means descendant */
7  .container .box {
8      /* Targets .box INSIDE .container */
9  }

```

Mistake #2: Single vs Double Colon

```
1 /* OLD syntax (still works) */
2 p:before {
3     content: "x";
4 }
5
6 /* NEW syntax (preferred) */
7 p::before {
8     content: "x";
9 }
10
11 /* RULE: Use :: for pseudo-ELEMENTS, : for pseudo-CLASSES */
```

Mistake #3: Over-Specific Selectors

```
1 /* TOO SPECIFIC - Hard to override later */
2 body div.container ul.menu li.item a.link {
3     color: blue;
4 }
5
6 /* BETTER - Just enough specificity */
7 .menu-link {
8     color: blue;
9 }
```

Keep selectors as simple as possible! You'll thank yourself later.

Mistake #4: Forgetting content Property

```
1 /* WRONG - ::before won't appear */
2 p::before {
3     color: red;
4 }
5
6 /* RIGHT - Must have content property */
7 p::before {
8     content: " ";
9     color: red;
10 }
11
12 /* Even if empty, you need content */
13 p::before {
14     content: "";
15     display: block;
16 }
```

Pro Tips & Tricks

Tip #1: Combine Pseudo-Classes

```
1 /* First item when hovered */
2 li:first-child:hover {
3     font-size: 20px;
4 }
5
6 /* Disabled buttons when focused */
7 button:disabled:focus {
8     outline: none;
9 }
10
11 /* Even items in a specific class */
12 .gallery img:nth-child(even) {
13     border: 2px solid blue;
14 }
```

Tip #2: :not() Selector (Exclusion)

```
1 /* All buttons EXCEPT the primary one */
2 button:not(.primary) {
3     background: gray;
4 }
5
6 /* All links that don't have a class */
7 a:not([class]) {
8     color: blue;
9 }
10
11 /* All paragraphs except the first one */
12 p:not(:first-child) {
13     margin-top: 20px;
14 }
```

Tip #3: Group Selectors for DRY Code

```
1 /* Instead of repeating styles... */
2 h1 { font-family: 'Arial', sans-serif; }
3 h2 { font-family: 'Arial', sans-serif; }
4 h3 { font-family: 'Arial', sans-serif; }
5
6 /* Group them! */
7 h1, h2, h3 {
8     font-family: 'Arial', sans-serif;
9 }
10
11 /* Complex grouping */
12 .button:hover,
13 .link:hover,
14 input[type="submit"]:hover {
15     opacity: 0.8;
16 }
```

Tip #4: Use :is() for Cleaner Code

```
1 /* Old way - repetitive */
2 header a:hover,
3 footer a:hover,
4 nav a:hover {
5     color: red;
6 }
7
8 /* New way - using :is() */
9 :is(header, footer, nav) a:hover {
10     color: red;
11 }
```

Hands-On Experiments

Experiment #1: Interactive Navigation Menu

Create a nav menu where:

- First item has a different color
- Last item has a special "Contact" badge
- Every odd item has a subtle background
- Links change color on hover

```
1 <!DOCTYPE html>
2 <html>
3 <head>
```

```

4      <style>
5          .nav {
6              display: flex;
7              list-style: none;
8              background: #333;
9              padding: 0;
10         }
11
12         .nav li {
13             padding: 15px 20px;
14         }
15
16         .nav li:first-child {
17             background: #667eea;
18         }
19
20         .nav li:last-child::after {
21             content: "□";
22         }
23
24         .nav li:nth-child(odd) {
25             background: rgba(255,255,255,0.05);
26         }
27
28         .nav li a {
29             color: white;
30             text-decoration: none;
31             transition: 0.3s;
32         }
33
34         .nav li:hover {
35             background: #667eea;
36         }
37
38         .nav li:hover a {
39             color: yellow;
40             font-size: 18px;
41         }
42     </style>
43 </head>
44 <body>
45     <ul class="nav">
46         <li><a href="#">Home</a></li>
47         <li><a href="#">About</a></li>
48         <li><a href="#">Services</a></li>
49         <li><a href="#">Portfolio</a></li>
50         <li><a href="#">Contact</a></li>
51     </ul>
52 </body>
53 </html>

```

Experiment #2: Fancy Input Forms

Style inputs with pseudo-classes:

```
1 <style>
2   input[type="text"],
3   input[type="email"] {
4     padding: 10px;
5     border: 2px solid #ddd;
6     border-radius: 5px;
7     transition: all 0.3s;
8   }
9
10  input:focus {
11    border-color: #667eea;
12    box-shadow: 0 0 10px rgba(102, 126, 234, 0.3);
13    outline: none;
14  }
15
16  input:valid {
17    border-color: green;
18  }
19
20  input:invalid:not(:focus) {
21    border-color: red;
22  }
23
24  input::placeholder {
25    color: #999;
26    font-style: italic;
27  }
28 </style>
29
30 <input type="email" placeholder="Enter your email" required>
```

Experiment #3: Advanced Card Grid

Create a responsive card grid with hover effects:

```
1 <style>
2   .grid {
3     display: grid;
4     grid-template-columns: repeat(3, 1fr);
5     gap: 20px;
6     padding: 20px;
7   }
8
9   .card {
10    background: white;
11    padding: 20px;
12    border-radius: 10px;
13    transition: all 0.3s;
14    position: relative;
```

```

15     }
16
17     .card:hover {
18         transform: translateY(-10px);
19         box-shadow: 0 10px 30px rgba(0,0,0,0.2);
20     }
21
22     .card:first-child {
23         grid-column: span 2;
24         background: linear-gradient(135deg, #667eea, #764ba2);
25         color: white;
26     }
27
28     .card:nth-child(3n) {
29         border-left: 5px solid #667eea;
30     }
31
32     .card::before {
33         content: attr(data-category);
34         position: absolute;
35         top: 10px;
36         right: 10px;
37         background: orange;
38         color: white;
39         padding: 5px 10px;
40         border-radius: 5px;
41         font-size: 12px;
42     }
43 </style>
44
45 <div class="grid">
46     <div class="card" data-category="Featured">Card 1</div>
47     <div class="card" data-category="New">Card 2</div>
48     <div class="card" data-category="Popular">Card 3</div>
49     <div class="card" data-category="Sale">Card 4</div>
50 </div>

```

Your Challenge Mission!

Mini Project: Build a Pricing Table

Your mission: Create a 3-column pricing table where:

1. The middle column is "featured" (bigger, different color)
2. Add badges using ::before
3. Style every other feature item differently
4. Add hover effects to buttons
5. Use :first-child and :last-child for special styling
6. Add a "Popular" ribbon on the featured plan

Starter code:

```
1 <div class="pricing">
2   <div class="plan">
3     <h3>Basic</h3>
4     <p class="price">$9/mo</p>
5     <ul class="features">
6       <li>10 Projects</li>
7       <li>5GB Storage</li>
8       <li>Email Support</li>
9     </ul>
10    <button>Choose Plan</button>
11  </div>
12
13  <div class="plan_featured">
14    <h3>Pro</h3>
15    <p class="price">$29/mo</p>
16    <ul class="features">
17      <li>Unlimited Projects</li>
18      <li>50GB Storage</li>
19      <li>Priority Support</li>
20      <li>Advanced Analytics</li>
21    </ul>
22    <button>Choose Plan</button>
23  </div>
24
25  <div class="plan">
26    <h3>Enterprise</h3>
27    <p class="price">$99/mo</p>
28    <ul class="features">
29      <li>Everything in Pro</li>
30      <li>500GB Storage</li>
31      <li>24/7 Phone Support</li>
32      <li>Dedicated Manager</li>
33    </ul>
34    <button>Choose Plan</button>
35  </div>
36 </div>
```

Try to use:

24

- .pricing > .plan (direct child)

What You've Mastered!

Step 5 Complete! You Now Know:

- **Basic Selectors** - Elements, classes, IDs
- **Universal Selector** - The mighty asterisk
- **Combinator Selectors** - Descendant, child, sibling
- **Attribute Selectors** - Targeting by attributes
- **Pseudo-Classes** - :hover, :focus, :nth-child, and more
- **Pseudo-Elements** - ::before, ::after, ::first-line
- **CSS Specificity** - Understanding the cascade
- **Advanced Techniques** - Combining selectors like a pro

What's Next?

Coming Up in Step 6

CSS Positioning - Making Elements Float and Stick!

You'll learn:

- Static, Relative, Absolute, Fixed, Sticky positioning
- Z-index and stacking contexts
- Creating overlays and tooltips
- Building sticky navigation bars
- Floating elements and clearing floats

CONGRATULATIONS!

You're becoming a CSS Selector Ninja!

Type "next step" when you're ready for Step 6!

Keep up the amazing work!