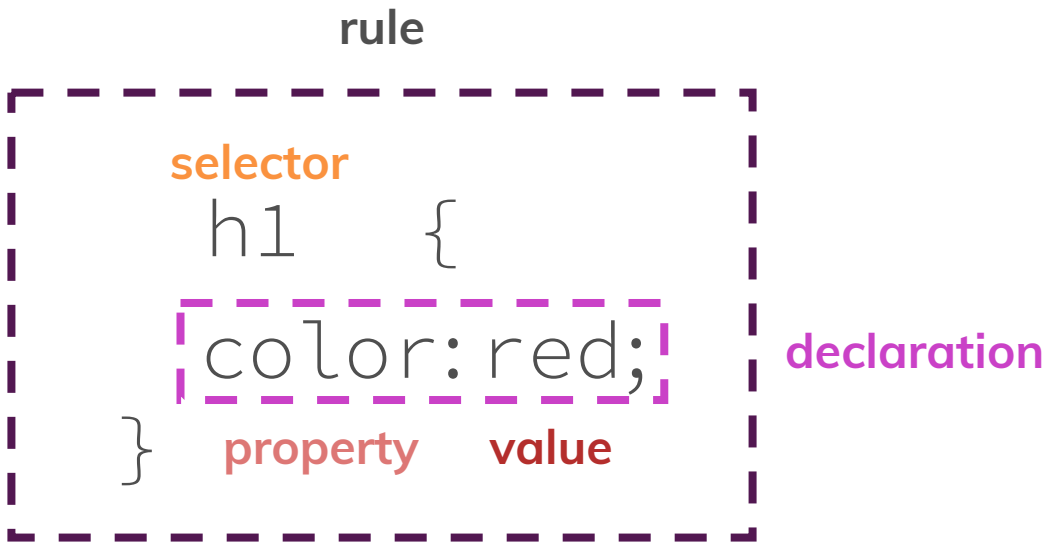
Understanding the CSS Syntax



More about Selectors

Elements

Set equal style for these elements

```html
<h1>Our header</h1>
<p>The Blog Post</p>
<div>More Info</div>
```

```css
h1 {
   color: red;
}
```

Classes

Set equal style for elements within the same class

```html
<h1 class="blog-post">
Our header</h1>
<p class="blog-post">
The blog post</p>
<div class="blog-post">
More info</div>
```

```css
.blog-post {
   color: red;
}
```

Universal

```html
<h1>Our header</h1>
<p class="blog-post">The
blog post</p>
```

```css
* {
   color: red
}
```

Rarely use this one!

More about Selectors

IDs

Set style to one specific element

```html
<h1 id="main-title">Our header</h1>
```

```css
#main-title {
    color: red;
}
```

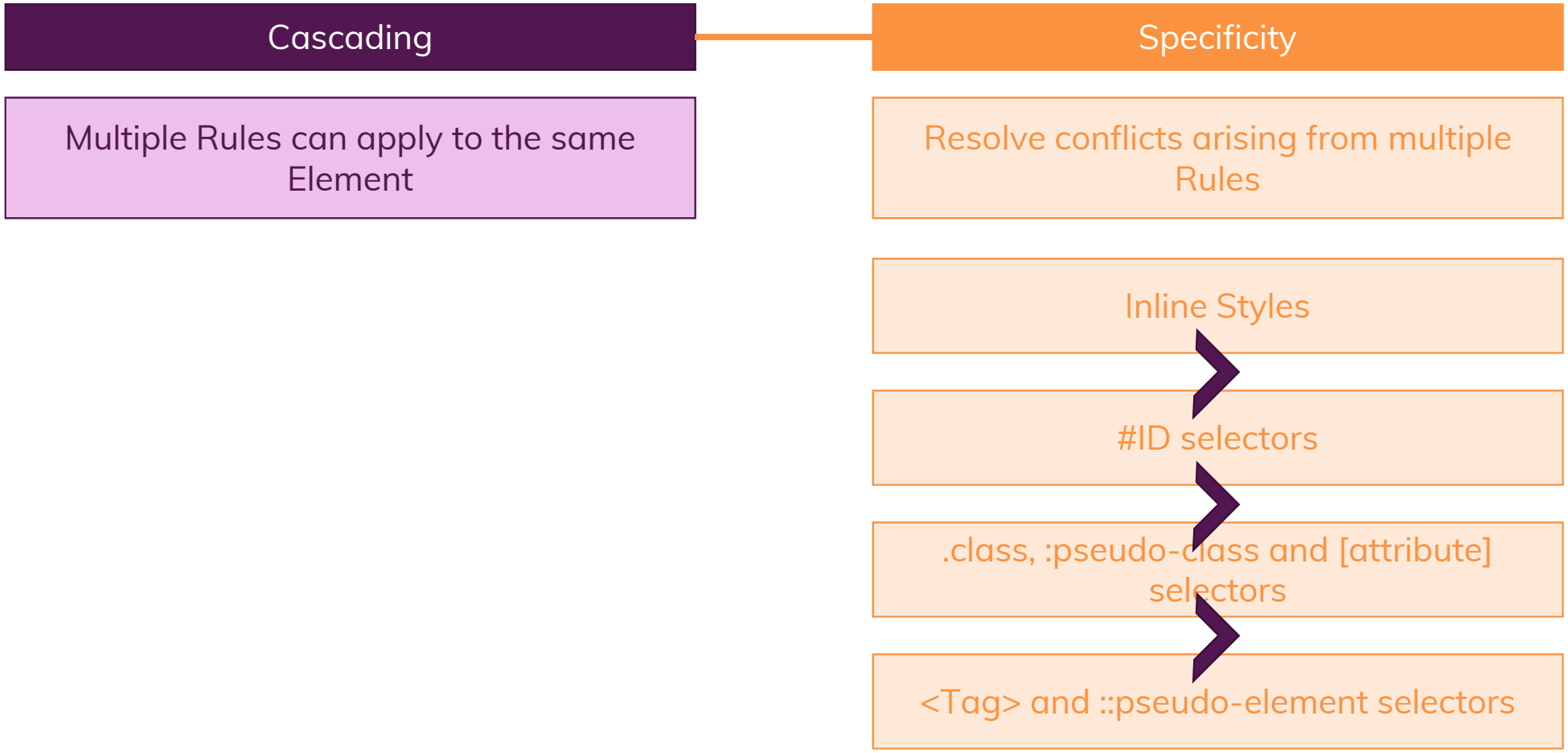
Attributes

Set equal styles to all elements with attribute(s)

```html
<button disabled>
    Click
</button>
```

```css
[disabled] {
    color: red;
}
```

Cascading Style Sheets & Specificity



Cascading Style Sheets & Specificity

Cascading

Multiple Rules can apply to the same Element

Specificity

Resolve conflicts arising from multiple Rules

Selector Hierarchy

Directly applied Styles win over Inheritance

`p {…} > div {…}` for `<div><p>Hi!</p></div>`

More specific Selector wins over less specific one

`p.some-class {…} > p {…}` for `<p class="some-class">Hi!</p>`

Inheritance

```css
div {
    color: red;
}
```

```css
p {
    color: green;
}
```

Parent styles are inherited by child elements if not overwritten!

```html
<div>
    <div>
        <h1>Inherited!</h1>
    </div>
    <p>Overwritten</p>
    <div>Inherited!</div>
    <article>
        <p>Overwritten</p>
    </article>
    <p>Overwritten</p>
</div>
```

Understanding Combinators

+ Adjacent Sibling

```
div + p {
}
```

~ General Sibling

```
div ~ p {
}
```

> Child

```
div > p {
}
```

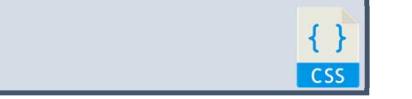
Descendant

```
div p {
}
```

Combinators – Adjacent Sibling

+ Adjacent Sibling

```css
h2 + p {
   color: red;
}
```



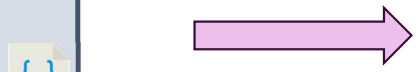
```html
<div>
   <h2>Not applied</h2>
   <p>CSS applied</p>
   <h2>Not applied</h2>
   <h3>Not applied</h3>
   <p>Not applied</p>
   <h2>Not applied</h2>
   <p>CSS applied</p>
</div>
```

- Elements share the same parent
- Second element comes **immediately** after first element

Combinators – General Sibling

~ General Sibling

```css
h2 ~ p {
   color: red;
}
```



```html
<div>
   <h2>Not applied</h2>
   <p>CSS applied</p>
   <h2>Not applied</h2>
   <h3>Not applied</h3>
   <p>CSS applied</p>
</div>
```

- Element share the same parent
- Second element comes after first element

Combinators – Child

Child

```css
div > p {
   color: red;
}
```

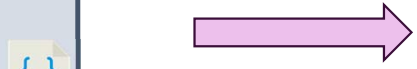
```html
<div>
   <div>Not applied</div>
   <p>CSS applied</p>
   <div>Not applied</div>
   <article>
      <p>Not applied</p>
   </article>
   <p>CSS applied</p>
</div>
```

Second element is a direct child of first element

Combinators – Descendant

Descendant

```css
div p {
   color: red;
}
```



```html
<div>
   <div>Not applied</div>
   <p>CSS applied</p>
   <div>Not applied</div>
   <article>
      <p>CSS applied</p>
   </article>
   <p>CSS applied</p>
</div>
```

Second element is a descendant of the first element

Value Types

Values are tightly coupled to specific property!

Pre-defined Options	Colors	Length, Sizes & Numbers	Functions
`display: block;`	`background: red;`	`height: 100px;`	`background: url(…);`
`overflow: auto;`	`color: #fa923f;`	`width: 20%;`	`transform: scale(…);`
	`color: #ccc;`	`order: 1;`	

Possible Values can be found in CSS References (e.g. MDN)!

Summary

CSS works with Rules

```
h1 {
    color: red;
}
p {
    color: red;
}
```

Different Types of Selectors

```
h1 {…}
.some-class {…}
[disabled] {…}
#some-id {…}
* {…}
```

Selectors with Combinators

```
div + p {
    color: red;
}
div ~ p {
    color: red;
}
div > p {
    color: red;
}
div p {
    color: red;
}
```

Properties & Values

- Long list of available Properties and Values
- Check MDN or comparable References
- Different Type of Values, depending on Properties

Inheritance & Specifity

- Parent styles are generally inherited
- Multiple rules can apply to one element
- Specifity resolves “multiple rules” conflicts
- Inheritance defaults can be changed