

Web Programming

CSS Part II.

Part II

Selectors

Recap

```
selector [p] {  
    font-family: Arial;  
    color: blue;  
    text-align: right;  
}
```

declaration

- **Selectors** indicate which element(s) the rule applies to
- **Declarations** describe the styling
 - List of property: value pairs separated by a semicolon

Element selector

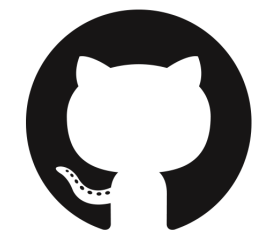
- Using single element as a selector:

```
body {  
    background-color: #f0f8ff;  
}
```

- Multiple elements can be listed by commas.
 - The individual elements can also have their own styles (like **p** below)

```
h1, h2, h3, p {  
    font-family: Verdana, Arial, sans-serif;  
}  
p {  
    margin: 1em;  
    padding: 0.5em;  
}
```

Exercise #1 (#1b)



[https://github.com/kbalog/web-programming/tree/master/](https://github.com/kbalog/web-programming/tree/master/exercises/css/selectors)
exercises/css/selectors

IDs and classes

- **ID** specifies a single unique element
 - HTML: **id** attribute with a unique value
 - CSS: id value prefixed by #

HTML `<p id="firstpar">...</p>`

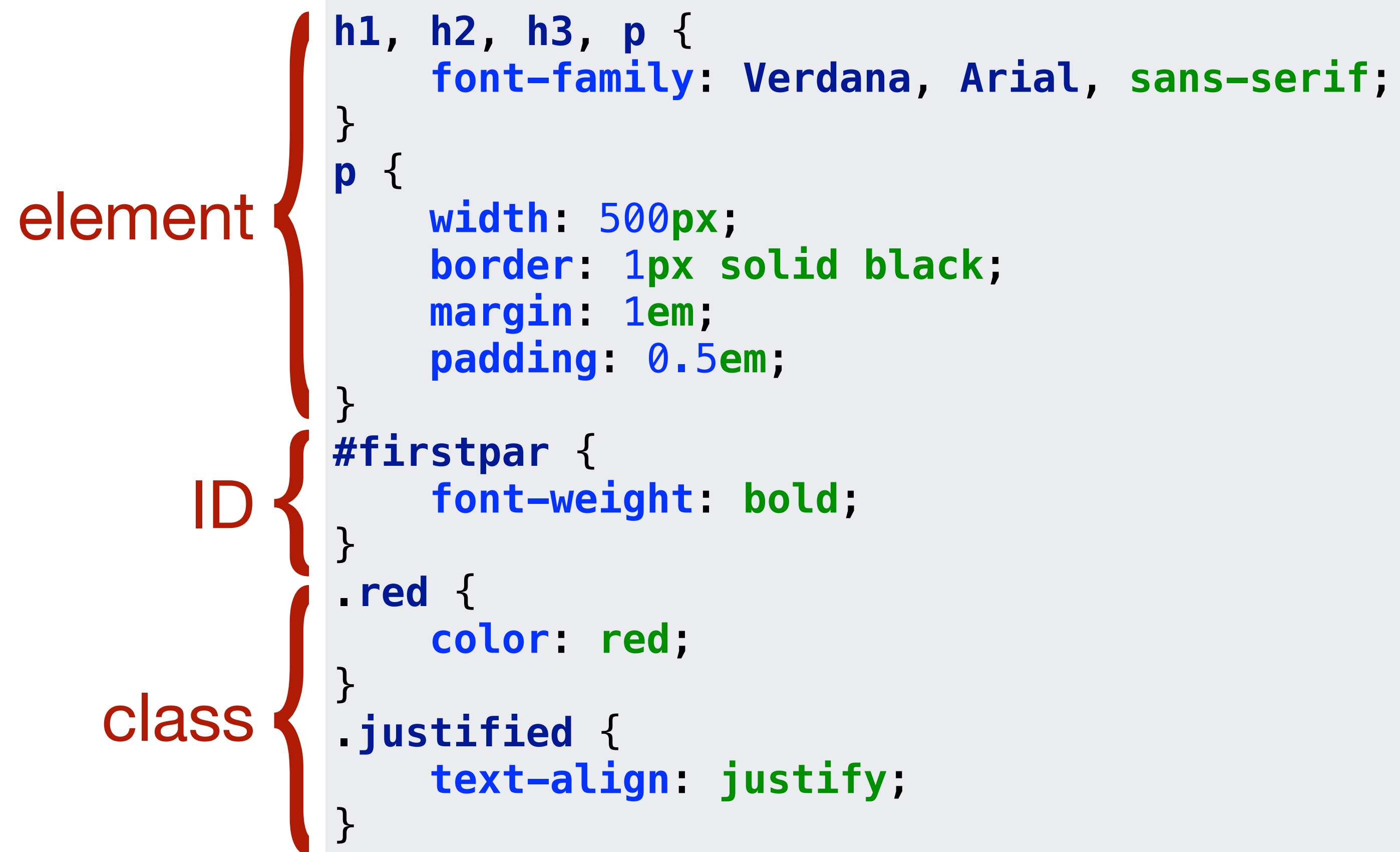
CSS `#firstpar {...}`

- **Class** can be assigned to a number of elements.
- An element can have multiple classes assigned to it.
 - HTML: **class** attribute with one or more values separated by space
 - CSS: class value prefixed by .

HTML `<p class="red">...</p>`
`<p class="red justified">...</p>`

CSS `.red {...}`
`.justified {...}`

Selectors so far



The diagram illustrates three types of CSS selectors: element, ID, and class. Each type is represented by a red bracket on the left, with its name written in red text. The corresponding CSS code is shown to the right of each bracket, with the selector part in blue and the property-value pairs in green.

```
h1, h2, h3, p {  
    font-family: Verdana, Arial, sans-serif;  
}  
p {  
    width: 500px;  
    border: 1px solid black;  
    margin: 1em;  
    padding: 0.5em;  
}  
#firstpar {  
    font-weight: bold;  
}  
.red {  
    color: red;  
}  
.justified {  
    text-align: justify;  
}
```

element {

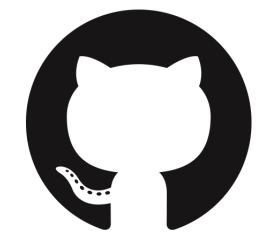
ID {

class {

ID selector vs. inline CSS

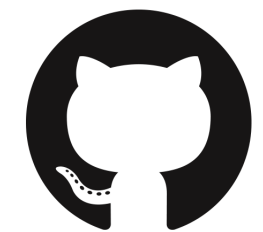
- With the ID selector inline CSS can be avoided
- That also means that it is possible from now on to move all style sheets to an external CSS file
- Best practice: **avoid inline CSS**
 - style sheets provide more maintainability
 - better separation of HTML data/structure and style/layout

Exercise #2



[https://github.com/kbalog/web-programming/tree/master/](https://github.com/kbalog/web-programming/tree/master/exercises/css/selectors)
exercises/css/selectors

Exercise #3



[https://github.com/kbalog/web-programming/tree/master/](https://github.com/kbalog/web-programming/tree/master/exercises/css/selectors)
exercises/css/selectors

Selectors

	Selector	Meaning	Example
★	Universal	Matches all elements in the document	* {} All elements on the page
★	Type	Matches element name	h1, h2, h3 {} <h1>, <h2>, <h3> elements
★	Class	Matches element class	.note {} Any elements whose class attribute has a value of note p.note {} Only <p> elements whose class attribute has a value of note
★	ID	Matches element ID	#introduction {} Element with an id attribute that has the value introduction

Selectors (2)

Selectors combinators



Descendant	Element that is descendent of another (not just direct child)	p a {} Any <a> inside an (even if there are other elements nested in between them)
Child	Element that is a direct child of another	li>a {} Any <a> elements that are children of an element
Adjacent sibling	Element that is the next sibling of another	h1+p {} First <p> element after any <h1> element (but not other <p> elements)
General sibling	Element that is a sibling of another, but does not have to be directly preceding	h1~p {} If there are two <p> elements that are siblings of an <h1> element, this applies to both

Example: adjacent vs. general sibling

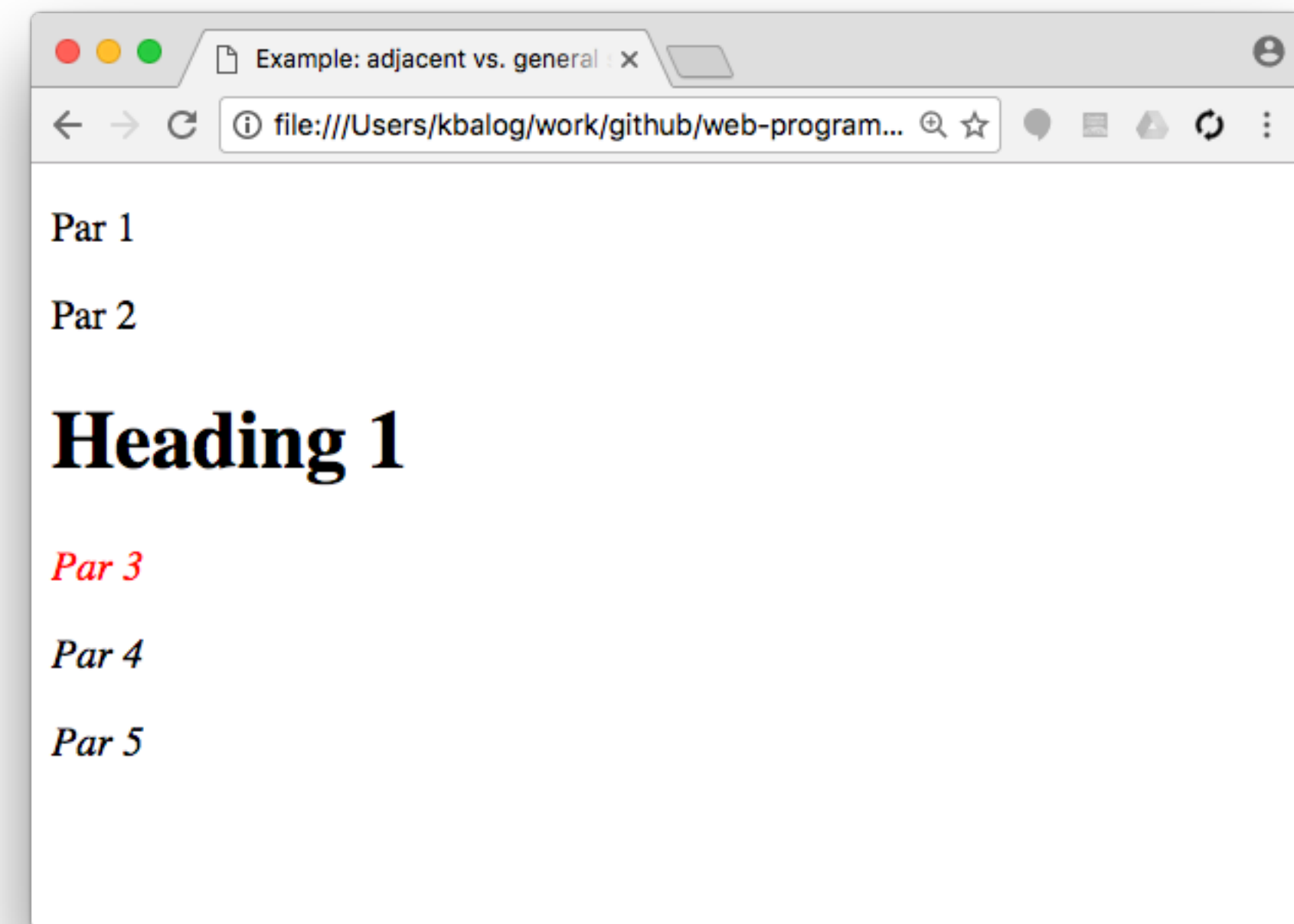
🐙 [examples/css/selectors/siblings.html](https://github.com/kbalog/web-programming-examples/blob/main/css/selectors/siblings.html)

CSS

```
h1 + p {  
  color: red;  
}  
  
h1 ~ p {  
  font-style: italic;  
}
```

HTML

```
<p>Par 1</p>  
<p>Par 2</p>  
<h1>Heading 1</h1>  
<p>Par 3</p>  
<p>Par 4</p>  
<p>Par 5</p>
```



Selectors (3)



Attribute selector	Element that have a specific attribute	p[title] {} Any <p> elements that have a title attribute
Pseudo-classes	Add special effects to some selectors, which are applied automatically in certain states	a:visited {} Any visited link
Pseudo-elements	Assign style to content that does not exist in the source document	p:first-line {} First line inside a <p> element

Question

- What's the difference?

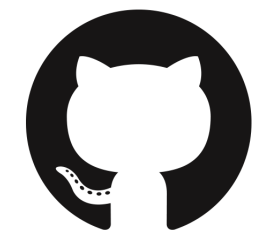
```
.intro a {...}
```

a element inside an
element that have the
intro class

```
a.intro {...}
```

only **a** elements that
have the **intro** class

Exercise #4



[https://github.com/kbalog/web-programming/tree/master/](https://github.com/kbalog/web-programming/tree/master/exercises/css/selectors)
exercises/css/selectors

CSS Priority Scheme

- This is the “cascading” part...
 - Many properties might affect the same element
 - Some of these might conflict with each other
 - Cascading decides which to apply

CSS priority scheme

#	CSS source type	Description
1	User defined	User-defined CSS in the browser
2	Inline	HTML element's style property
3	Media type	Media-specific CSS
4	Importance	!important overwrites previous types
5	Selector specificity	More specific selector over generic ones
6	Rule order	Last rule of declaration
7	Parent inheritance	Not specified is inherited from parent
8	CSS definition	Any CSS definition
9	Browser default	Initial values

CSS priority scheme

#	CSS source type	Description
1	User defined	User-defined CSS in the browser
2	Inline	HTML element's style property
3	Media type	Media-specific CSS
4	Importance	!important overwrites previous types
5	Selector specificity	More specific selector over generic ones
6	Rule order	Last rule of declaration
7	Parent inheritance	Not specified is inherited from parent
8	CSS definition	Any CSS definition
9	Browser default	Initial values

Inheritance

- Some properties are **inherited** by child elements
 - Font-family, color, etc.
- Others are **not inherited** by child elements
 - Background-color, border, etc.
- Inheritance can be forced using **inherit**

```
body {...}
.page {
  background-color: #efefef;
  padding: inherit;
}
```

CSS priority scheme

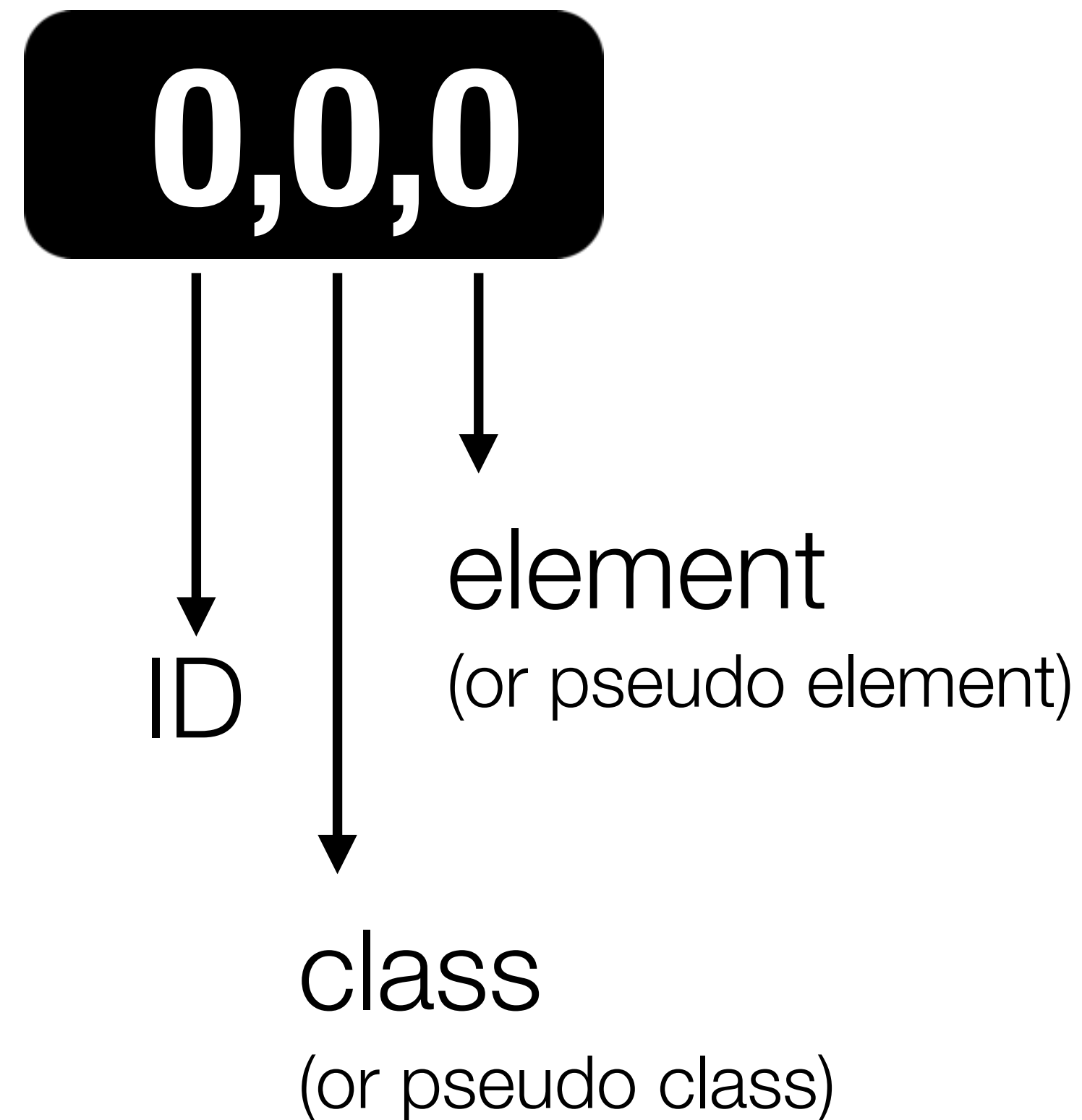
#	CSS source type	Description
1	User defined	User-defined CSS in the browser
2	Inline	HTML element's style property
3	Media type	Media-specific CSS
4	Importance	!important overwrites previous types
5	Selector specificity	More specific selector over generic ones
6	Rule order	Last rule of declaration
7	Parent inheritance	Not specified is inherited from parent
8	CSS definition	Any CSS definition
9	Browser default	Initial values

Specificity hierarchy

- If multiple selectors apply to the same element, the one with **higher specificity** wins
- Every selector has its place in the *specificity hierarchy*
 1. IDs
#div
 2. Classes, attributes, pseudo-classes
.classes, [attributes], :hover
 3. Elements (types) and pseudo-elements
p, :after

Computing specificity

- Think in a number system (with a large base)



Computing specificity

- Think in a number system (with a large base)

body #content .data img:hover

1,2,2

↓
ID

#content

↓
element **body img**
(or pseudo element)

↓
class **.data :hover**
(or pseudo class)

Specificity wars

- http://www.stuffandnonsense.co.uk/archives/css_specificity_wars.html



Emperor

1,0,0

ID



Vader

0,1,0

class



Storm trooper

0,0,1

element



a

1 x element selector

Sith: 0, 0, 1



p a

2 x element selectors

Sith: 0, 0, 2



.whatever

1 x class selector

Sith: 0, 1, 0



a.whatever

1 x element selector
1 x class selector

Sith: 0, 1, 1



p a.whatever

2 x element selectors
1 x class selector

Sith: 0, 1, 2



.whatever .whatever

2 x class selectors

Sith: 0, 2, 0



p.whatever a.whatever

2 x element selectors
2 x class selectors

Sith: 0, 2, 2



#whatever

1 x id selector

Sith: 1, 0, 0



a#whatever

1 x element selector
1 x id selector

Sith: 1, 0, 1



.whatever a#whatever

1 x element selectors
1 x class selector
1 x id selector

Sith: 1, 1, 1



.whatever .whatever #whatever

2 x class selectors
1 x id selector

Sith: 1, 2, 0

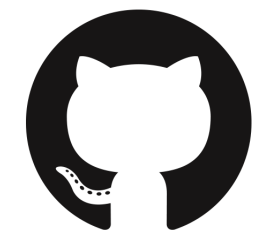


#whatever #whatever

2 x id selectors

Sith: 2, 0, 0

Exercise #5



[https://github.com/kbalog/web-programming/tree/master/](https://github.com/kbalog/web-programming/tree/master/exercises/css/selectors)
exercises/css/selectors

Solutions

#	CSS	Score	Explanation
1	<code>* { }</code>	0	
2	<code>li { }</code>	1	one element
3	<code>li:first-line { }</code>	2	element + pseudo-element
4	<code>ul li { }</code>	2	two elements
5	<code>ul ol+li { }</code>	3	three elements
6	<code>h1 + *[rel=up] { }</code>	11	one attribute, one element
7	<code>ul ol li.red { }</code>	13	one class, three elements
8	<code>li.red.level { }</code>	21	two classes, one element
9	<code>style=""</code>	1000	one inline styling
10	<code>p { }</code>	1	one element
11	<code>div p { }</code>	2	two elements
12	<code>.sith</code>	10	one class
13	<code>div p.sith { }</code>	12	two elements and a class
14	<code>#sith</code>	100	one id
15	<code>body #darkside .sith p { }</code>	112	element, ID, class, element (1+100+10+1)

Online specificity calculator

<http://specificity.keegan.st>

Specificity Calculator

A visual way to understand [CSS specificity](#). Change the selectors or paste in your own.

li:first-child h2 .title

0

Inline styles

0

IDs

2

Classes, attributes
and pseudo-classes

2

Elements and
pseudo-elements

+ Duplicate

Quiz

- The answer is the color of the text after CSS is applied
 - I.e., the HTML part is always the same

```
<div id="main" class="container">  
  <p id="foo" class="bar boo">Something clever goes here</p>  
</div>
```

Keep in mind

- The color property is **inherited** by child elements
- However, any style declaration (even with the lowest specificity) overwrites the inherited value
- Specificity is to be computed only when there are multiple declarations that apply to the same element

#1

```
HTML <div id="main" class="container">  
      <p id="foo" class="bar boo">Something clever goes here</p>  
    </div>
```

```
CSS body {color: red;}  
    p {color: blue;}
```

- The answer is the color of the text after CSS is applied

- ☐ red
- ☐ blue
- ☐ black

#1 Solution

```
HTML <div id="main" class="container">  
      <p id="foo" class="bar boo">Something clever goes here</p>  
    </div>
```

```
CSS body {color: red;}  
p {color: blue;}
```

- The answer is the color of the text after CSS is applied

- ☐ red
- ☒ blue
- ☐ black

Explanation:

The red color is inherited from body. The explicit style declaration for the p element overwrites it.

#2

```
HTML <div id="main" class="container">  
      <p id="foo" class="bar boo">Something clever goes here</p>  
    </div>
```

```
CSS p.bar {color: red;}  
    p.boo {color: blue;}
```

- The answer is the color of the text after CSS is applied

- ☐ red
- ☐ blue
- ☐ black

#2 Solution

HTML `<div id="main" class="container">
 <p id="foo" class="bar boo">Something clever goes here</p>
</div>`

CSS `p.bar {color: red;}
p.boo {color: blue;}`

- The answer is the color of the text after CSS is applied

- ☐ red
- ☒ blue
- ☐ black

Explanation:

p.bar and p.boo have the same specificity. The last rule of declaration decides.

#3

HTML `<div id="main" class="container">
 <p id="foo" class="bar boo">Something clever goes here</p>
</div>`

CSS `p {color: red;}
.container {color: blue;}`

- The answer is the color of the text after CSS is applied

- ☐ red
- ☐ blue
- ☐ black

#3 Solution

HTML `<div id="main" class="container">
 <p id="foo" class="bar boo">Something clever goes here</p>
</div>`

CSS `p {color: red;}
.container {color: blue;}`

- The answer is the color of the text after CSS is applied

- ☒ red
- ☐ blue
- ☐ black

Explanation:

The blue color is inherited from div.container.
The explicit style declaration for the p element overwrites it.

#4

```
HTML <div id="main" class="container">  
      <p id="foo" class="bar boo">Something clever goes here</p>  
    </div>
```

```
CSS #main {color: red;}  
    body .container {color: blue;}
```

- The answer is the color of the text after CSS is applied

- ☐ red
- ☐ blue
- ☐ black

#4 Solution

HTML `<div id="main" class="container">
 <p id="foo" class="bar boo">Something clever goes here</p>
</div>`

CSS `#main {color: red;}
body .container {color: blue;}`

- The answer is the color of the text after CSS is applied

- ☒ red
- ☐ blue
- ☐ black

Explanation:

The color is inherited from the parent div. For that div, the ID #main has a higher specificity (1-0-0) than "body .container" (0-1-1).

#5

HTML

```
<div id="main" class="container">  
  <p id="foo" class="bar boo">Something clever goes here</p>  
</div>
```

CSS

```
#foo {color: red;}  
#main {color: blue;}
```

- The answer is the color of the text after CSS is applied

- ☐ red
- ☐ blue
- ☐ black

#5 Solution

HTML `<div id="main" class="container">
 <p id="foo" class="bar boo">Something clever goes here</p>
</div>`

CSS `#foo {color: red;}
#main {color: blue;}`

- The answer is the color of the text after CSS is applied

- ☒ red
- ☐ blue
- ☐ black

Explanation:

The color inherited from the parent div (blue) is overwritten by the declaration for the ID #foo.

#6

HTML

```
<div id="main" class="container">  
  <p id="foo" class="bar boo">Something clever goes here</p>  
</div>
```

CSS

```
.container p {color: red;}  
div .boo {color: blue;}
```

- The answer is the color of the text after CSS is applied

- ☐ red
- ☐ blue
- ☐ black

#6 Solution

```
HTML <div id="main" class="container">  
      <p id="foo" class="bar boo">Something clever goes here</p>  
    </div>
```

```
CSS .container p {color: red;}  
    div .boo {color: blue;}
```

- The answer is the color of the text after CSS is applied

- ☐ red
- ☒ blue
- ☐ black

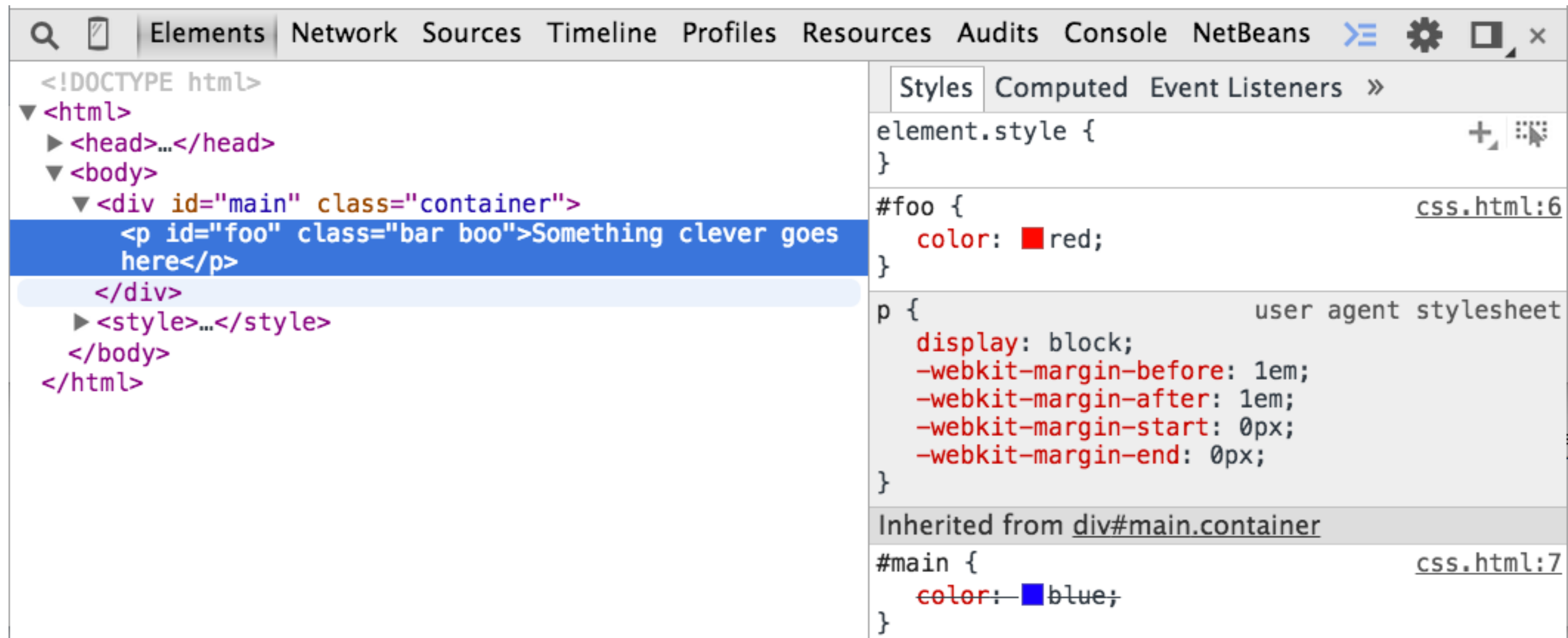
Explanation:

Both declarations apply to the `<p>` element (the first because `p` the second because `.boo`).

They have the same specificity (0-1-1), therefore the last rule of declaration decides.

When in doubt

- Use the browser's developer functions



Best practices

- Minimize the number of selectors
- Use ID to make a rule more specific
- Never use **!important**