

# 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

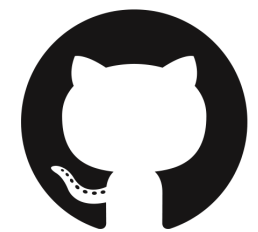
- Using single element as a selector:

```
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)



[github.com/dat310-2024/info/tree/master/](https://github.com/dat310-2024/info/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

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

# 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

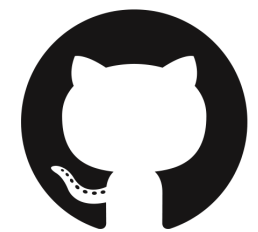


# Exercise #2



[github.com/dat310-2024/info/tree/master/](https://github.com/dat310-2024/info/tree/master/exercises/css/selectors)  
**exercises/css/selectors**

# Exercise #3



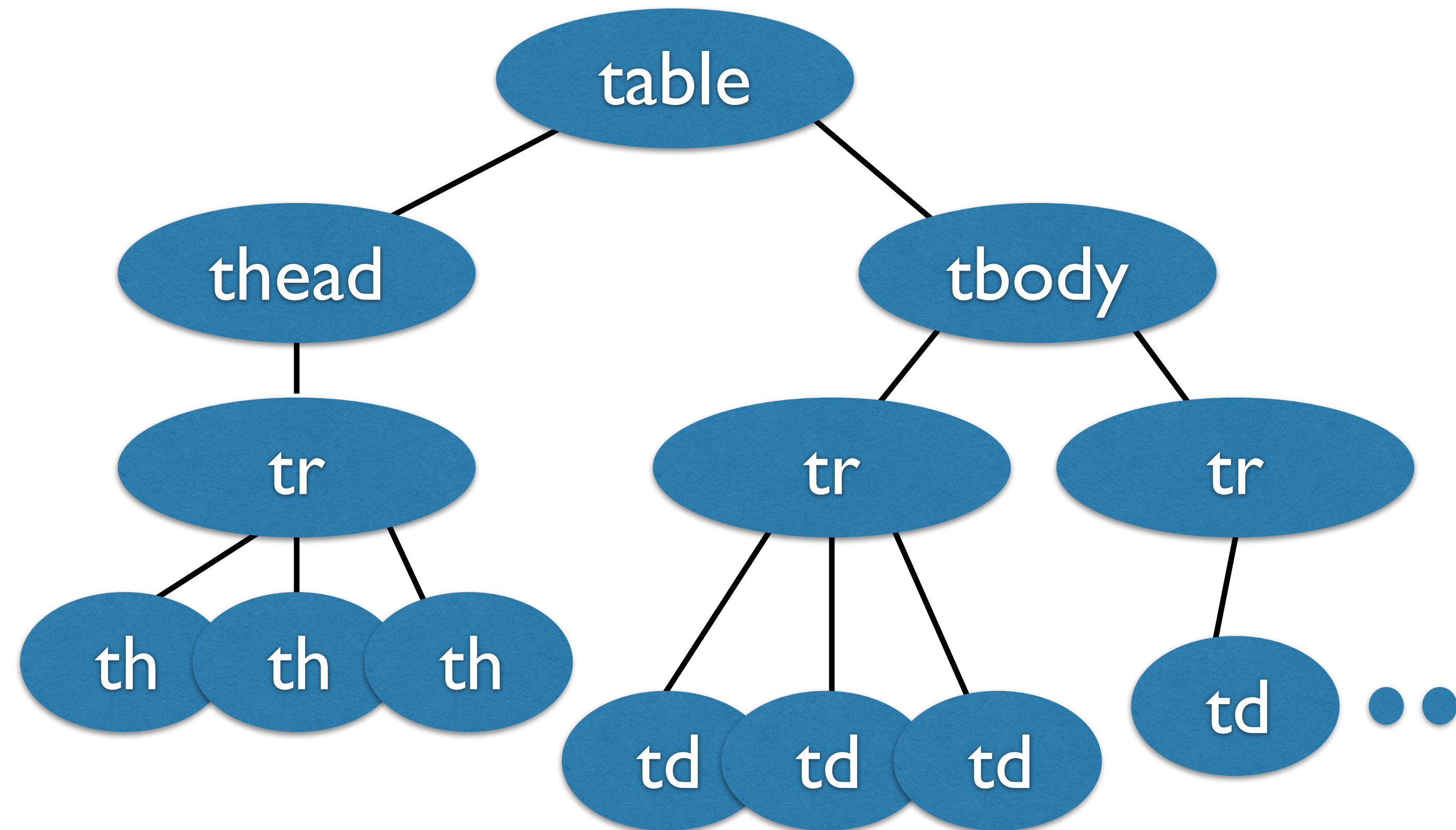
[github.com/dat310-2024/info/tree/master/](https://github.com/dat310-2024/info/tree/master/exercises/css/selectors)  
**exercises/css/selectors**

# Selectors

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

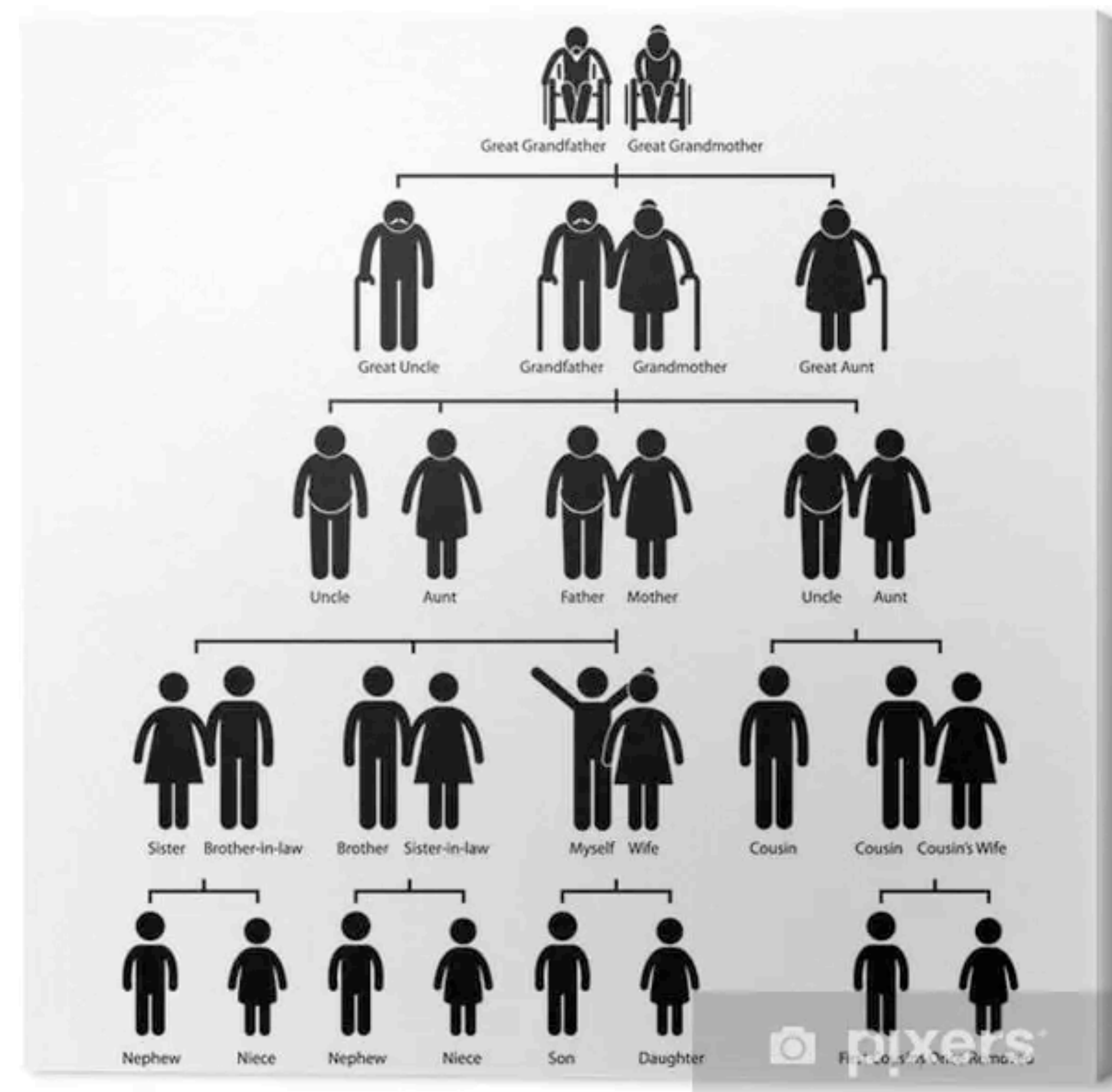
# Elements tree

```
<table border="1">
  <thead>
    <tr>
      <th>First name</th>
      <th>Last name</th>
      <th>Points</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>John</td>
      <td>Smith</td>
      <td>100</td>
    </tr>
    [...]
  </tbody>
</table>
```



# Elements tree

```
<table border="1">
  <thead>
    <tr>
      <th>First name</th>
      <th>Last name</th>
      <th>Points</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>John</td>
      <td>Smith</td>
      <td>100</td>
    </tr>
    [...]
  </tbody>
</table>
```



# Elements tree

```
<table border="1">
  <thead>
    <tr>
      <th>First name</th>
      <th>Last name</th>
      <th>Points</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>John</td>
      <td>Smith</td>
      <td>100</td>
    </tr>
    [...]
  </tbody>
</table>
```

**Child:** td elements are Childs of the tr element.

**Siblings:** td elements in same row are siblings.

**Descendant:** all td and tr elements are descendants of table

# Selectors (2)

## Selectors combinators



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



# Example: adjacent vs. general sibling

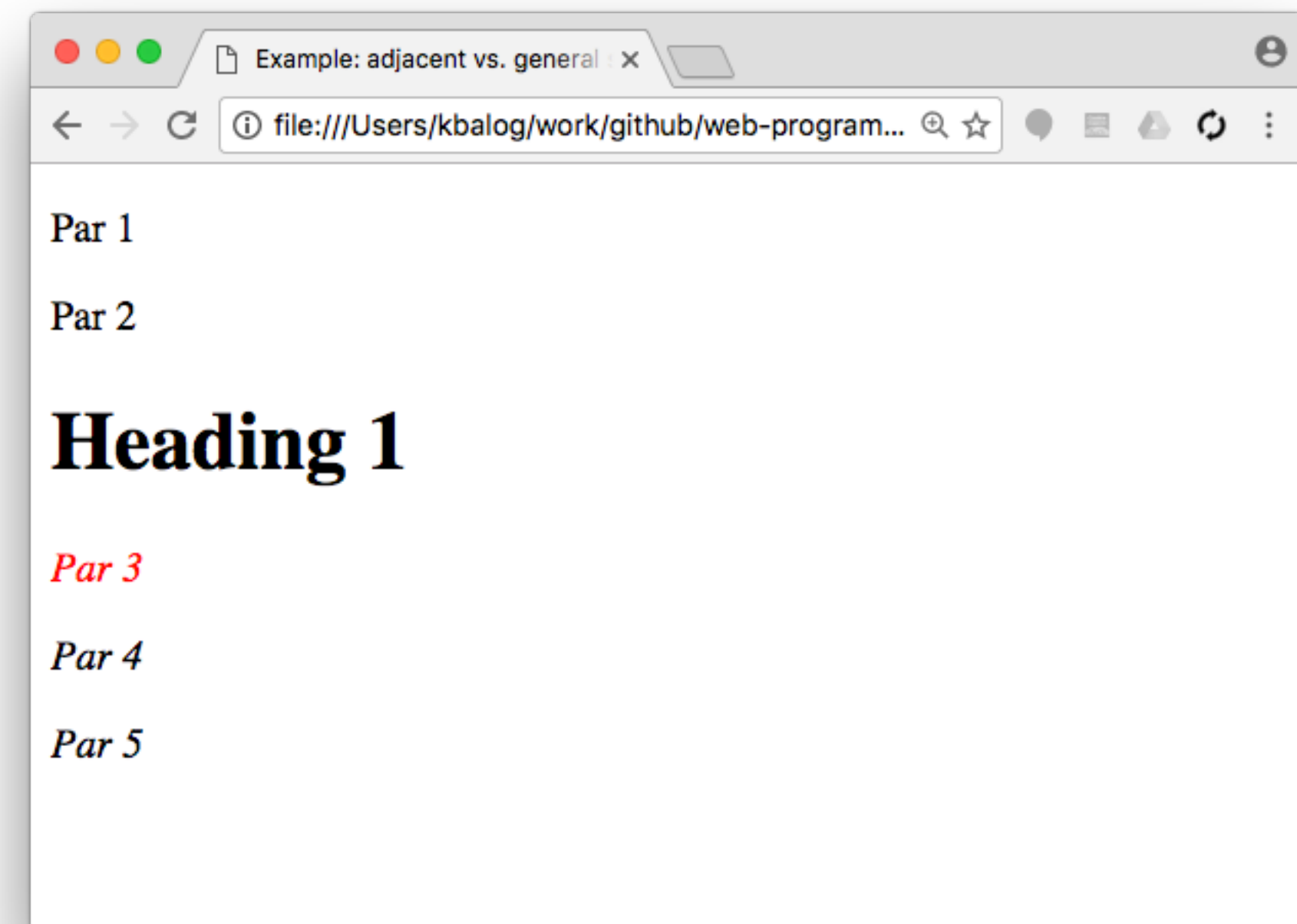
🐙 [examples/css/selectors/siblings.html](https://github.com/kbalog/web-programming-examples/blob/main/css/selectors/siblings.html)

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)



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

# Question

- What's the difference?

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

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

# 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

# Question

- What's the difference?

```
#header.callout {...}
```

```
#header .callout {...}
```

# Question

- What's the difference?

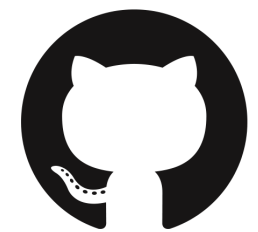
```
#header.callout {...}
```

element that has ID  
**header** as well as  
class **callout**

```
#header .callout {...}
```

all elements with the class  
name **callout** that are  
descendants of the element  
with ID **header**

# Exercise #4



[github.com/dat310-2024/info/tree/master/](https://github.com/dat310-2024/info/tree/master/exercises/css/selectors)  
**exercises/css/selectors**

# List properties

- Shape of list item markers
  - Property: **list-style-type**
  - Values for unordered lists:
    - **circle, square, ...**
  - Values for ordered lists:
    - **upper-roman, lower-alpha, ...**
- Remove list markers
  - **list-style-type: none**
- Using an image as the list item marker
  - **list-style-image: url('filename.png')**

# 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

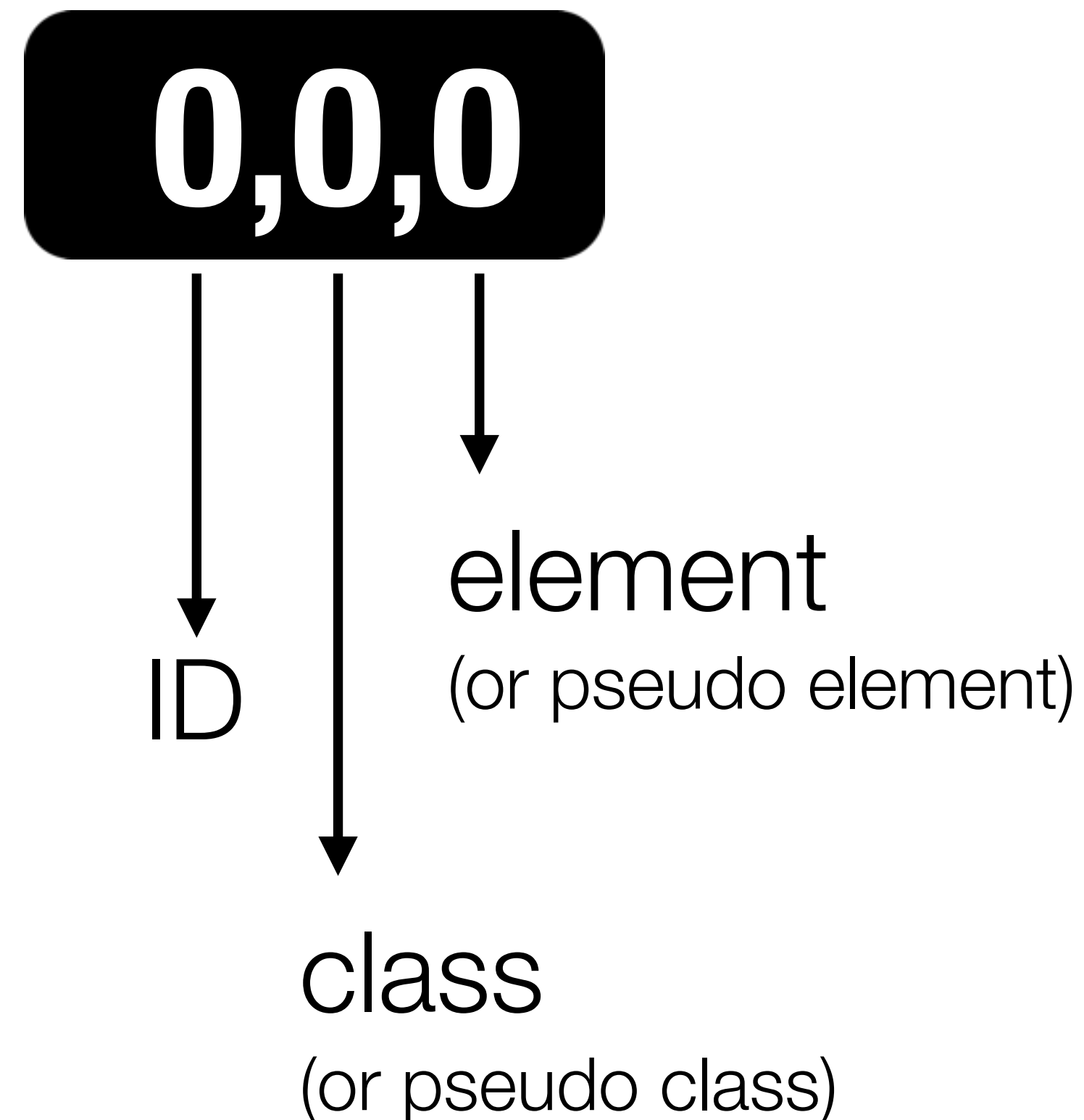
#	CSS source type	Description
1	User defined	User-defined CSS in the browser
2	Inline	HTML element's style property
3	Media type	Media-specific CSS
4	Importance	!important overwrites previous types
5	Selector specificity	More specific selector over generic ones
6	Rule order	Last rule of declaration
7	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](http://www.stuffandnonsense.co.uk/archives/css_specificity_wars.html)



Vader

**1,0,0**

ID



Maul

**0,1,0**

class



Storm trooper

**0,0,1**

element





**a**

1 x element selector

Sith power: 0,0,1



**p a**

2 x element selectors

Sith power: 0,0,2



**.foo**

1 x class selector \*

Sith power: 0,1,0



**a.foo**

1 x element selector  
1 x class selector

Sith power: 0,1,1



**p a.foo**

2 x element selectors  
1 x class selector

Sith power: 0,1,2



**.foo .bar**

2 x class selectors

Sith power: 0,2,0



**p.foo a.bar**

2 x element selectors  
2 x class selectors

Sith power: 0,2,2



**#foo**

1 x id selector

Sith power: 1,0,0



**a#foo**

1 x element selector  
1 x id selector

Sith power: 1,0,1



**.foo a#bar**

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

Sith power: 1,1,1



**.foo .foo #foo**

2 x class selectors  
1 x id selector

Sith power: 1,2,0

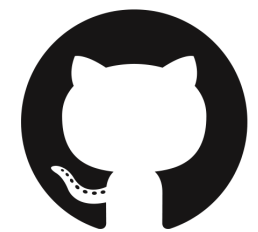


**style**

1 x style attribute

Sith power: 1,0,0,0

# Exercise #5



[github.com/dat310-2024/info/tree/master/](https://github.com/dat310-2024/info/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

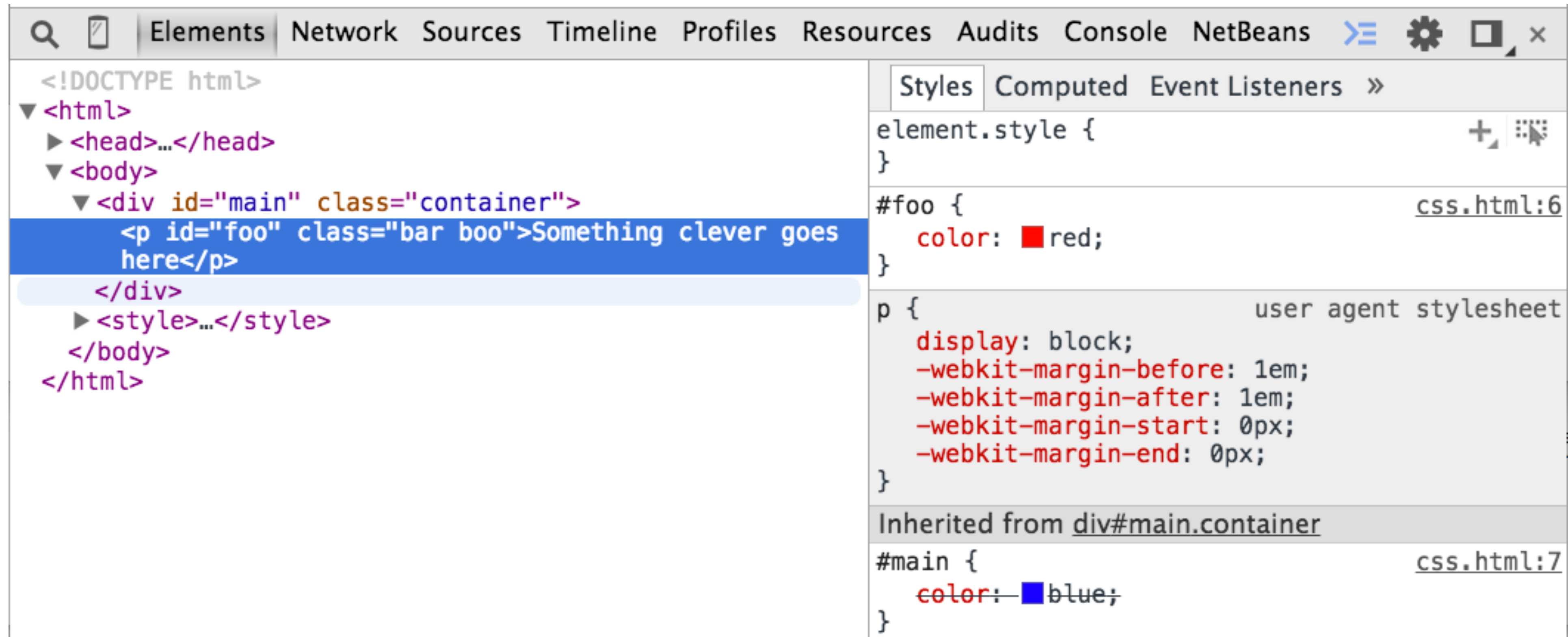
## Explanation:

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**