Programming Languages for Web Applications

Cascading Style Sheets

Robbie Hott with content from Praphamontripong, Soffa

Announcements

- Sprint 1 due tonight
  - Project pitches this week with our TAs, please sign up soon!
  - *Reminder: sprints may not be turned in late*
- Homework 2 due next Monday
- Office hours:
  - Prof Hott: Mondays 3-5p, Tuesdays 2-3p
  - TA Office Hours: Posted on course website

The Basic Idea

The Tardis Tales

A collection of Doctor Who themed fan fiction, centered around the 10th and 11th doctors and their companions.

Story 1 – The two doctors

…

+

<html> <body> <p> <nav> <h1> <h2> <a> <table> <b> <u> <i> <blink> <img>

=

The Tardis Tales

A collection of Doctor Who themed fan fiction, centered around the **10th** and **11th** doctors and their companions

- Story 1 – The two doctors

…

Your Content

HTML Markup
(structure)

Your Website

The Basic Idea – Take Two

The Tardis Tales

A collection of Doctor Who themed fan fiction, centered around the 10th and 11th doctors and their companions.

Story 1 – The two doctors

…

+

<html> <body> <p> <nav> <h1> <h2> <a> <table> <b> <u> <i> <blink> <img>

+

```
body {
  background-color: #3578cd;
  color: #fff;
}

p {
  font-family: Times, serif;
  color: #fff;
}

h1 {
  font-family: Arial;
}

...
```

=

The Tardis Tales

A collection of Doctor Who themed fan fiction, centered around the **10th** and **11th** doctors and their companions

- Story 1 – The two doctors

…

Your Content

HTML Markup
(structure)

CSS
(styling)

Your Website

Cascading Style Sheets (CSS)

- Language used to describe the presentation of a web page
- Designed to **separate concerns**
  - HTML: content and structure of the document
  - CSS: look and feel (style and layout)
- Separating the visual presentation (CSS) from the structure (HTML) increases readability and maintainability
  - Reusing style across elements and pages
  - Update HTML without needing to update styles

CSS Rules - Structure

- CSS rules contain two main parts:
  - **Selector** – a pattern used to select which HTML element(s) to be styled
  - **Declaration** – description of how to style selected elements

$$\underset{\text{selector}}{\texttt{p}}\ \overbrace{\texttt{\{}\ \underbrace{\texttt{color: red;}}_{\text{declaration}}\ \underset{\text{property}}{\texttt{font-weight:}}\ \underset{\text{value}}{\texttt{bold;}}\ \texttt{\}}}^{\text{declaration block}}$$

CSS Rules - Structure

- A selector can have multiple declarations
- Multiple selectors can share a declaration block

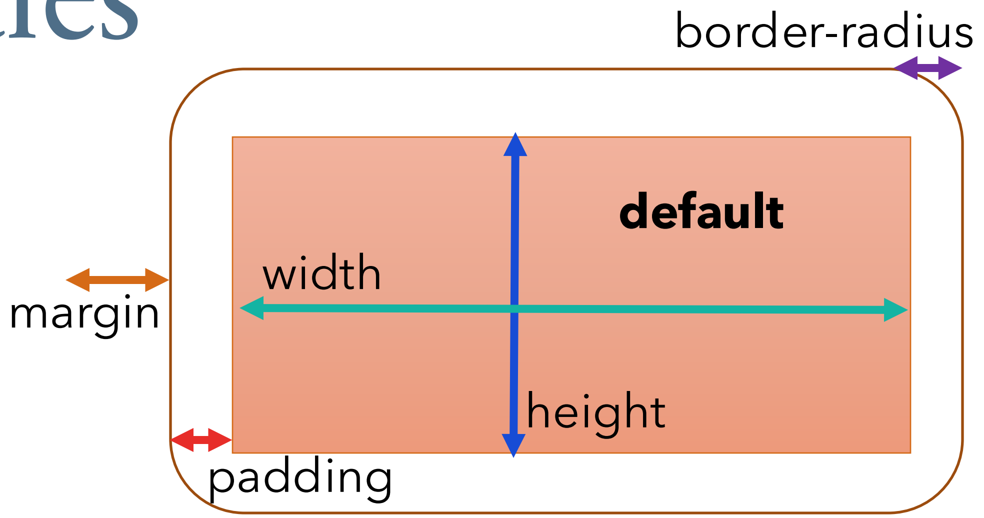
```
h1, h2, h3, p {
    text-align: center;
    color: red;
    font-weight: bold;
}
```

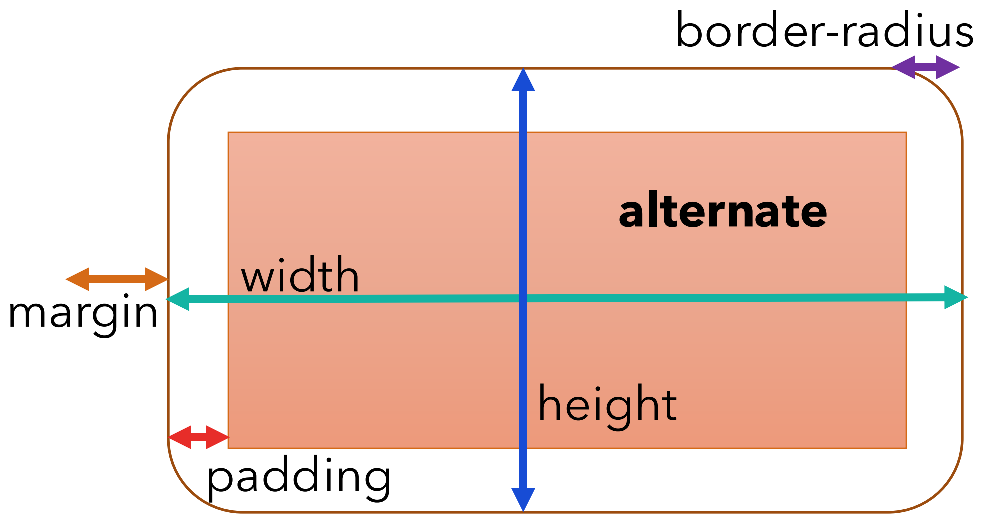
CSS Common Properties

https://cs4640.cs.virginia.edu/readings/markup-style/css-rules.html

Using CSS – Box Properties

- Boxes by default are just large enough to fit contents
  - Padding, margin, and border are **outside** the box
- Alternate box model typically used
  - Padding sets spacing inside box
  - Border thickness included in box
    Margin sets spacing outside box
- Specify size in pixels (`px`) or percent (`%`)
  - Percent is relative to container's dimensions





Introducing CSS in HTML

Three ways to add CSS to HTML

- **Inline** – specify the style properties in the opening tag of an element
  - Applies to *that element* only
- **Document-level** – specify style in the document `<head>`
- **External-level** – specify style in a separate `.css` file, link to the file in the document `<head>`

Inline CSS

- Key-value properties are added to the style attribute

```
<tag style="property1:value1; property2:value2; ..."> ... </tag>
```

- For example:

```
<p style="text-align:center; color:red; font-weight:bold">
   This is a centered paragraph of bold red text.
</p>
```

Document-Level CSS

- Style specifications are added to `<head>` in the `<style>` tag
  - Body of the `<style>` tag is raw CSS format

```
<head>
   <style>
   p {
      text-align: center;
      color: red;
      font-weight: bold;
   }
   ...
   </style>
</head>
<body>
   <p>This is a centered paragraph of bold red text.</p>
</body>
```

External-Level CSS

- CSS written in separate file, included by HTML

file.html

```
<head>
   <link rel="stylesheet"
      type="text/css"
      href="style.css">
</head>
<body>
   <p>This is a centered
      paragraph of bold red
      text.</p>
</body>
```

style.css

```
p {
    text-align: center;
    color: red;
    font-weight: bold;
}
...
```

Types of Selectors

- **Element** - refers to an HTML element
  - Also known as simple selector or type selector

```
h1, p, div { color: red; font-weight: bold; }
```

- **ID** - refers to one HTML element on the page with that id attribute
  - Each page can only have one element with a given id (unique)

```
#intro { font-family: Arial, sans serif; }
```

```
<p id="intro">Intro text goes here.</p>
```

Types of Selectors

- **Class** – selects elements with a particular class attribute
  - Can apply to multiple elements of the given type

```
p.warning { color: red; font-weight: bold; }
```

```
<p class="warning">Warning text goes here.</p>
```

- **Generic Class** – selects any element with a particular class attribute
  - Can apply to multiple elements of any type

```
.highlight { font-weight: bold; font-size: 18px }
```

```
<h4 class="highlight">Highlight heading.</h4>
<p class="highlight">Highlight text.</p>
```

Types of Selectors

- **Pseudo-class** – selects elements when they are in a particular state
  - Ex: when something happens in the browser
  - Many pseudo-classes
    - `:active` - elements activated by user (for click, between mouse down and up)
    - `:checked` - radio, checkbox, option elements checked by user
    - `:disabled` - elements that cannot receive focus
    - `:focus` - element that has the user's focus
    - `:hover` - elements currently hovered over by mouse
    - `:link` - link element that has not yet been visited
    - `:visited` - link element that has been visited

```
a:hover { color: red; text-decoration: none; }
```

Types of Selectors

- **Pseudo-class** – selects elements when they are in a particular state
  - Some help us select a pattern of children
    - `:first-child`
    - `:last-child`
    - `:nth-child`
    - `:nth-last-child`
    - `:first-of-type`
    - `:last-of-type`
    - `:nth-of-type`
    - `:nth-last-of-type`

Make even rows of a table shaded light cyan

```
tr:nth-child(even) {
    background-color: LightCyan;
}
```

Types of Selectors

- **Pseudo-element** – selects part of an element to apply the style
  - Many pseudo-element selectors
    - `::after` - just after the element (could be used to add content)
    - `::before` - just before the element (could be used to add content)
    - `::first-letter` - first letter of the element text
    - `::first-line` - first line of the element text
    - `::selection` - the selected part of the document
    - `::backdrop` - immediately below the element when rendered in fullscreen

```
h1::first-letter { color: red; font-size: 200%; }
```

Types of Selectors

- **Universal** – selects all elements in the document or all elements inside another element
  - Every element in the HTML

```
* { color: blue; font-size: 12px; }
```

  - Every element inside a `<div>` element

```
div * { color: green; font-size: 11px; }
```

Types of Selectors

- **Attribute** – selects elements based on the presence or value of an attribute
  - `[attribute]` - select all elements with attribute present
  - `[attribute=match]` - select all elements that have attribute with this value (equals)
  - `[attribute^=match]` - select all elements that have attribute beginning with this value
  - `[attribute$=match]` - select all elements that have attribute ending with this value
  - `[attribute*=match]` - select all elements that have attribute containing this value

Types of Selectors

- **Attribute** – selects elements based on the presence or value of an attribute
  - Most useful for form elements (UI for validation and error handling)

```
  [value] { background-color: yellow; color: red; }

input[value] { background-color: yellow; color: red; }
```

```
<form action="action.php" method="post">
   <label for="fname">First Name:</label>
   <input type="text" id="fname" name="fname" value="Julia" />
</form>
```

Combining Selectors

Selectors can be combined to refine the document elements that are selected based on HTML document hierarchy

- **Descendant selectors** – space separated selectors
  - Any descendent in the tree will be selected
  - Ex: any `span` inside of a `div` (at any level)

```
div span { color: green; font-size: 11px; }
```

Combining Selectors

- **Direct child selectors** – selectors separated by **>**
  - Only immediate children will be selected
  - Ex: only `p` elements that are direct children of a `div`

```
div > p { color: green; font-size: 11px; }
```

Combining Selectors

- **Adjacent sibling selectors** – selectors separated by **+**
  - Targets adjacent sibling of a specified element based on selector
  - Ex: The next `li` element that is a sibling of the `li` element with class `selected`

```
li.selected + li { background-color: deepskyblue; }
```

```
<ul>
  <li>First</li>
  <li class="selected">Second</li>
  <li>Third</li>
  <li>Fourth</li>
</ul>
```

Combining Selectors

- **General sibling selectors** – selectors separated by **~**
  - Targets all elements that are next siblings of a specified element
  - Ex: All next sibling `li` elements of the `li` element with class `selected`

```
li.selected ~ li { background-color: deepskyblue; }
```

```
<ul>
  <li>First</li>
  <li class="selected">Second</li>
  <li>Third</li>
  <li>Fourth</li>
</ul>
```

Cascading Style Sheets

- What if multiple rules apply to a given element?
  - Which one is chosen? How does the element get styled?
- They all get applied, but factors determine which will take precedence!
  - Think: inheritance – in Java, the child classes take precedence

Cascading Style Sheets

- Cascade: the order of CSS rules matter
    - When two rules have equal "specificity," then the later rule will be used
- Specificity: a weight calculated for a given CSS declaration
    - Rules with higher specificity will be chosen

Cascading Style Sheets

- Consider these rules, which would take preference?

```
h1 {
        color: green;
}

h1 {
        color: red;
}
```

Cascade!

```
<h1 class="heading">An Example Heading.</h1>
```

Cascading Style Sheets

- Consider these rules, which would take preference?

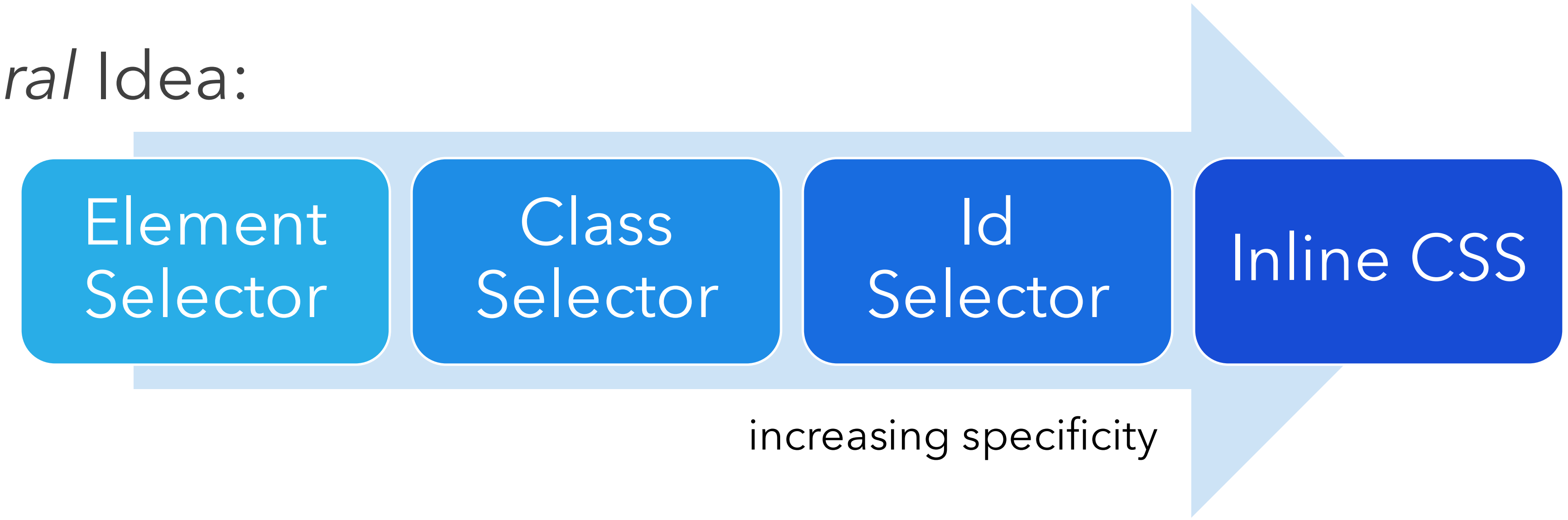
```
h1.heading {
        color: green;
}

h1 {
        color: red;
}
```

Specificity!

```
<h1 class="heading">An Example Heading.</h1>
```

Specificity

- *General* Idea:



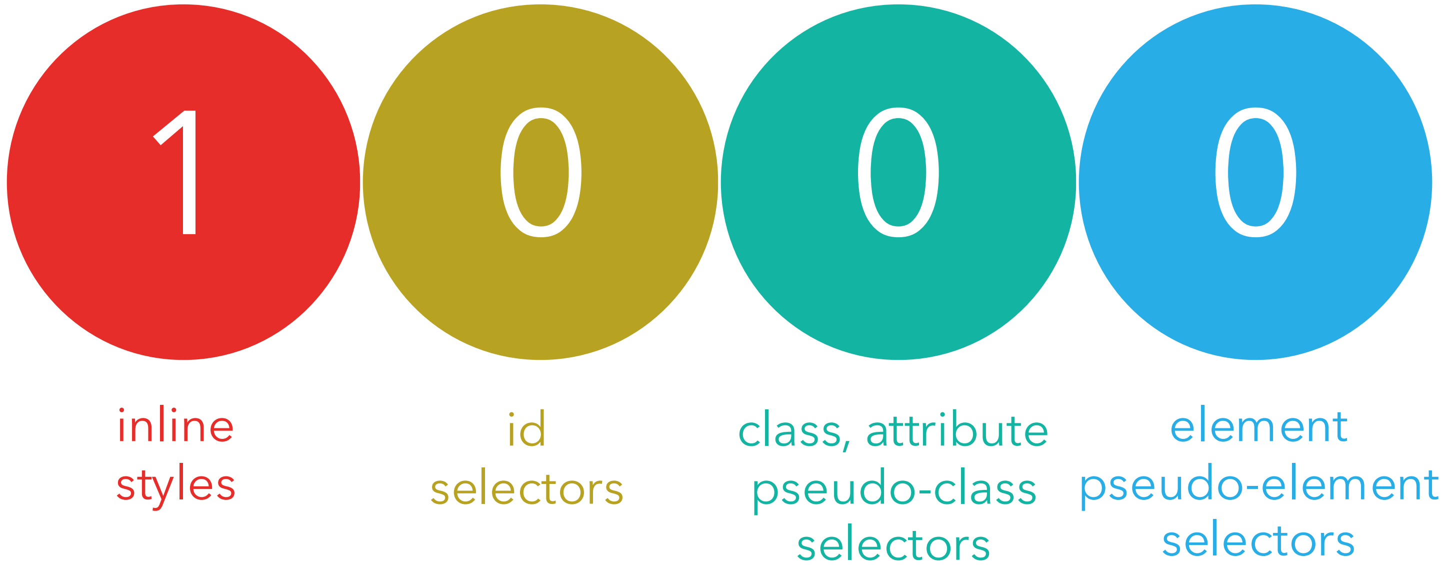
  - Roughly: more selectors = has more precedence
- In reality: *Much more nuanced!*

Specificity

0	0	0	0
inline styles	id selectors	class, attribute, pseudo-class selectors	element, pseudo-element selectors
Defined in the html `style=""` attribute	Number of each in the overall selector	Number of each in the overall selector	Number of each in the overall selector

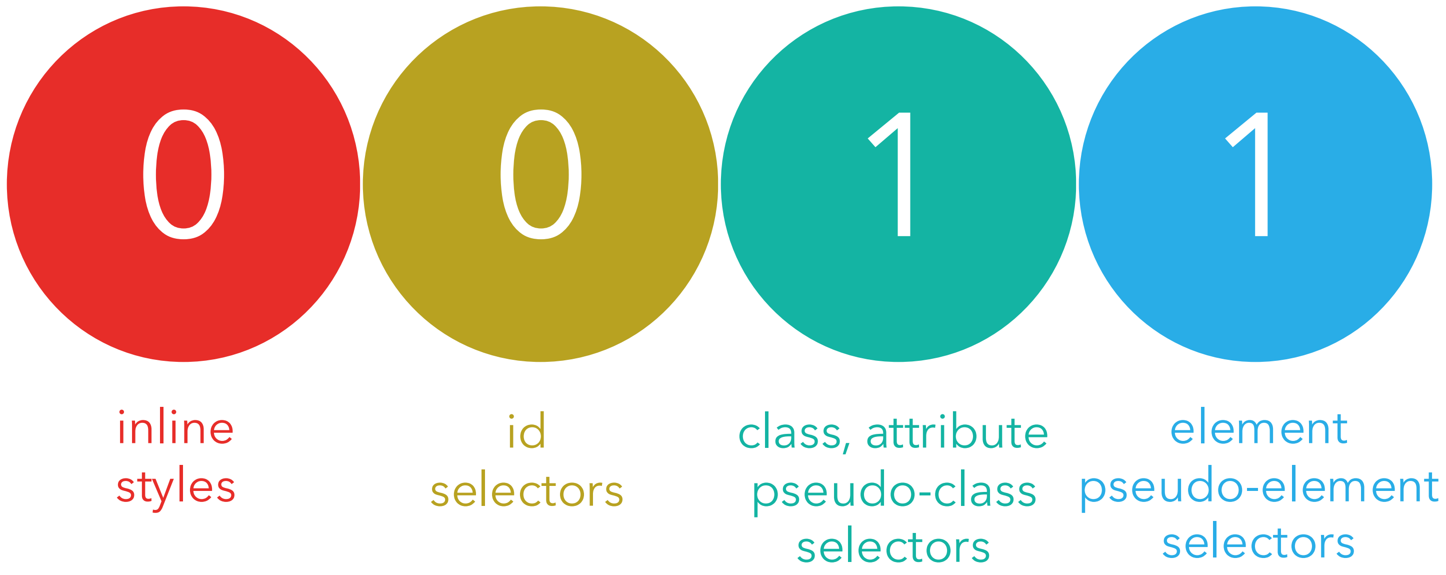
Specificity

```
<h1 style="font-family: Arial">An Example Heading.</h1>
```


1
0
0
0
inline styles
id selectors
class, attribute pseudo-class selectors
element pseudo-element selectors

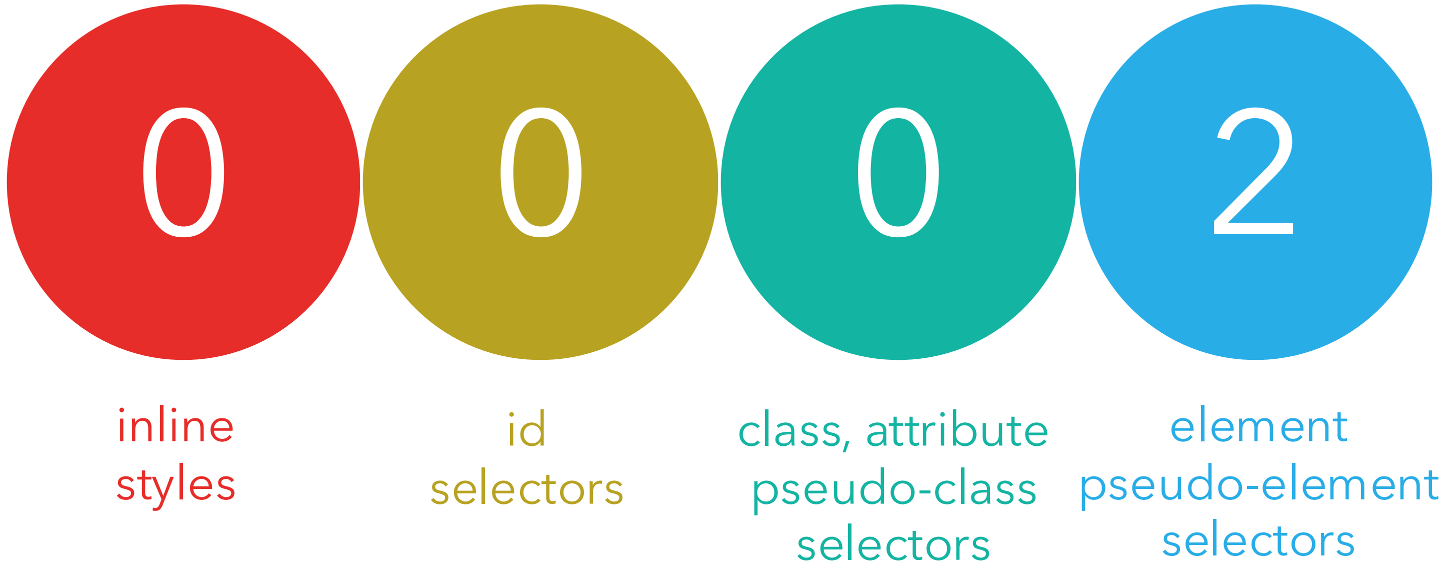
Specificity

```
h1.heading { color: green; }
```


0
0
1
1
inline styles
id selectors
class, attribute pseudo-class selectors
element pseudo-element selectors

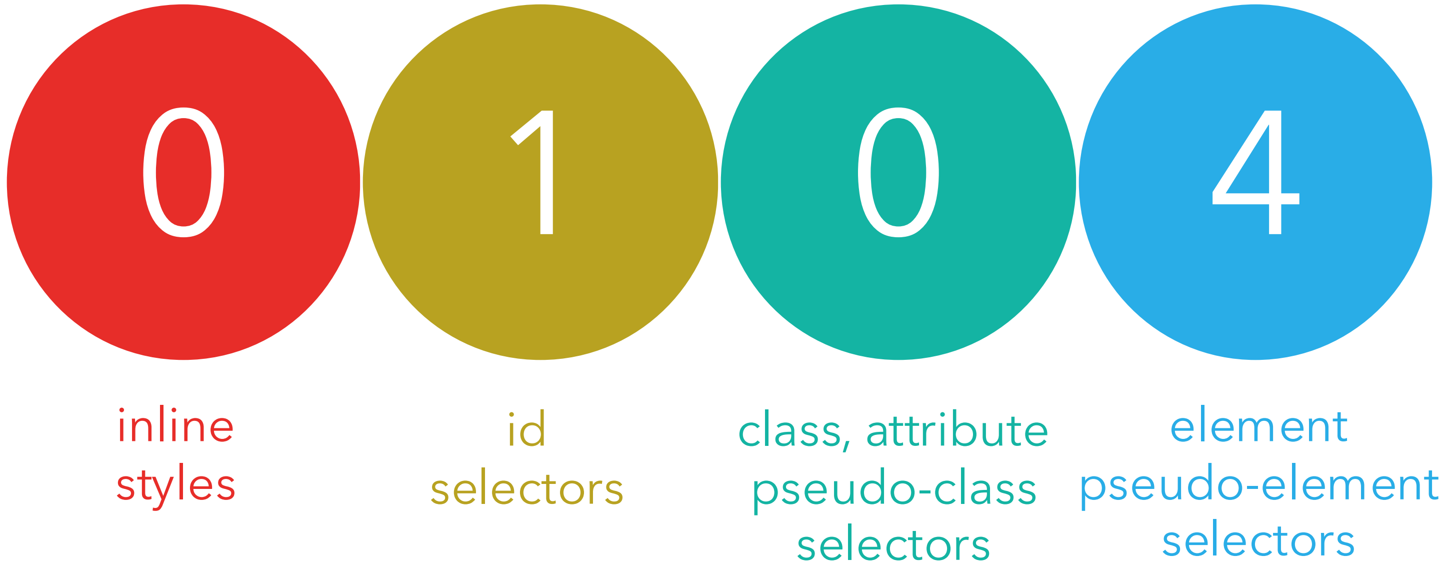
Specificity

```
div > span { font-weight: bold; }
```


0
0
0
2
inline styles
id selectors
class, attribute pseudo-class selectors
element pseudo-element selectors

Specificity

```
#outer div ul li a { text-decoration: none; }
```


0
1
0
4
inline styles
id selectors
class, attribute pseudo-class selectors
element pseudo-element selectors

CSS Specificity

Activity