CSS Selectors: Classes and Ids

Classes and ids

There are 3 basic types of CSS selectors:

Element selector (this is the one we've been using)	**p**	All **<p>** elements
✧ **ID selector** ✧	**#abc**	element with **id="abc"**
✧ **Class selector** ✧	**.abc**	elements with **class="abc"**

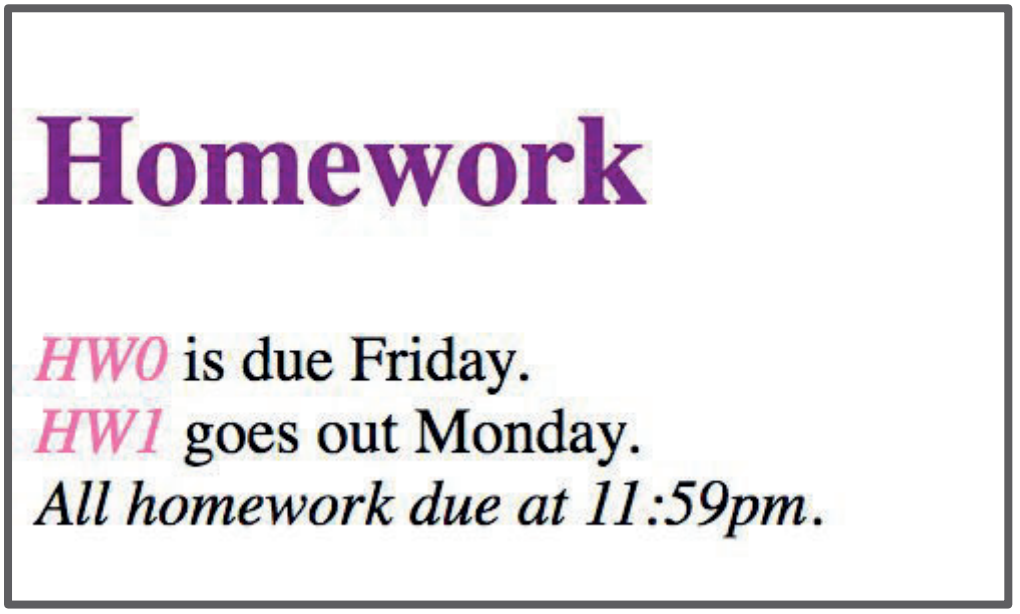
```
<h1 id="title">Homework</h1>
<em class="hw">HW0</em> is due Friday.<br/>
<em class="hw">HW1</em> goes out Monday.<br/>
<em>All homework due at 11:59pm.</em>
```

Classes and ids

```
<h1 id="title">Homework</h1>
<em class="hw">HW0</em> is due Friday.<br/>
<em class="hw">HW1</em> goes out Monday.<br/>
<em>All homework due at 11:59pm.</em>
```

```
.hw {
  color: hotpink;
}

#title {
  color: purple;
}
```



More on `class` and `id`

- **`class`** and **`id`** are special HTML attributes that can be used on any HTML element
  - **`class`**: Used on 1 or more elements; identifies a **collection** of elements
  - **`id`**: Used on exactly 1 element per page; identifies **one unique** element
- Can apply multiple classes by space-separating them: `<span class="hw new">HW1</span>`
- Often used with span and div to create generic elements: e.g. **`<span class="highlight">`** is like creating a "highlight" element

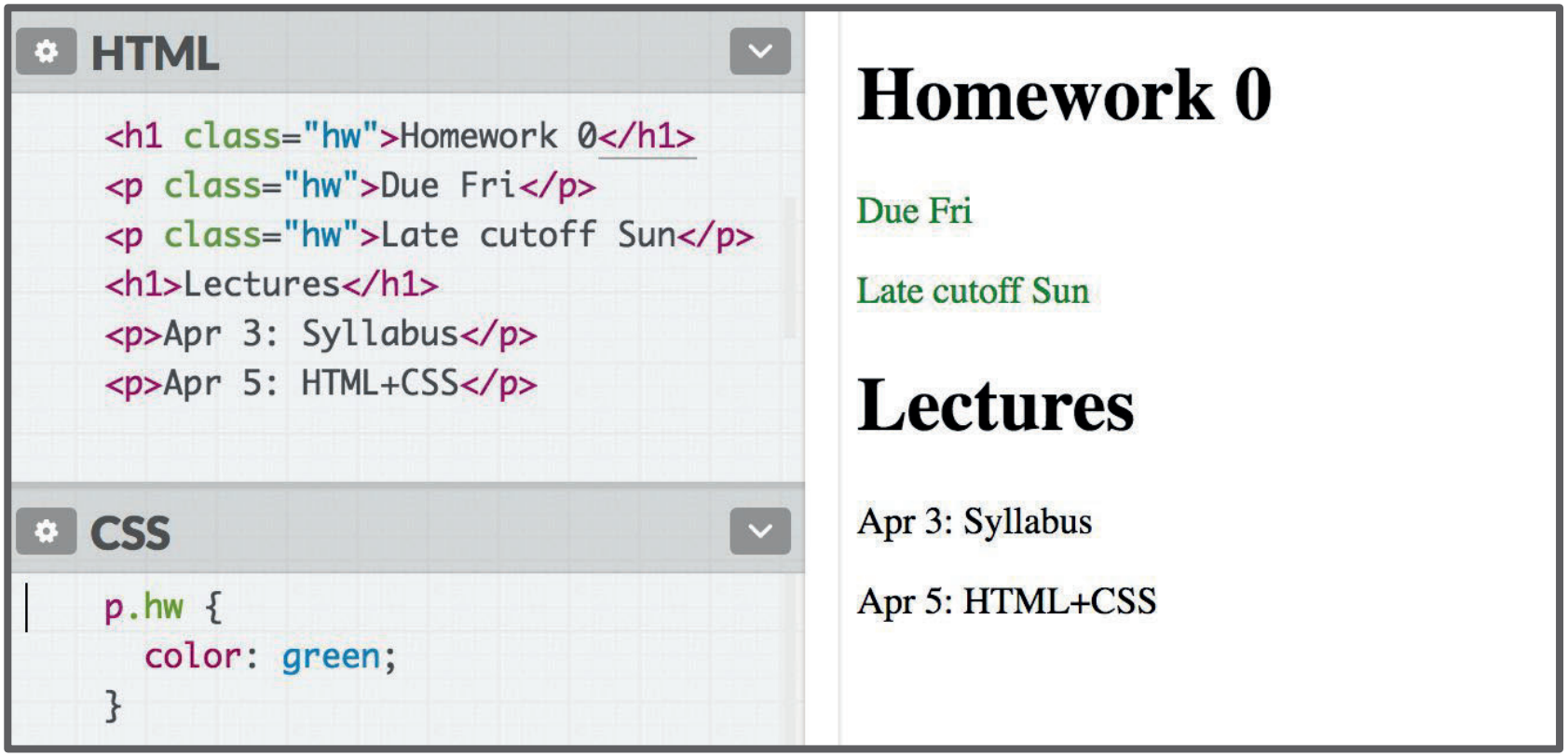
Other selectors:
Next time!

Overflow slides

(we didn't cover these)

element.className

Syntax	Example	Example described
element*.*className	**p.abc**	**<p>** elements with abc class



Descendent selector

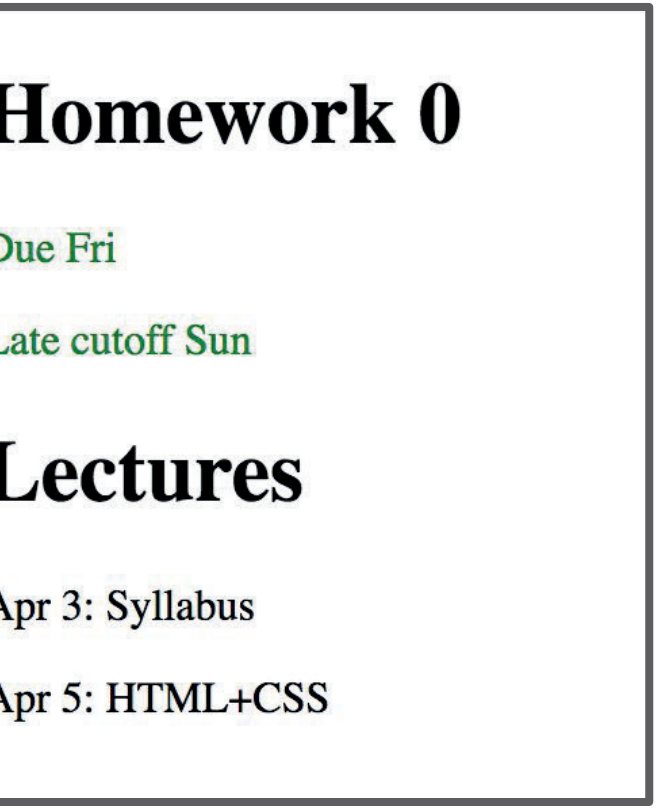
Syntax	Example	Example described
selector selector	`div strong`	`<strong>` elements that are descendants of a `<div>`

HTML

```
<div class="hw">
  <h1>Homework 0</h1>
  <p>Due Fri</p>
  <p>Late cutoff Sun</p>
</div>
```

CSS

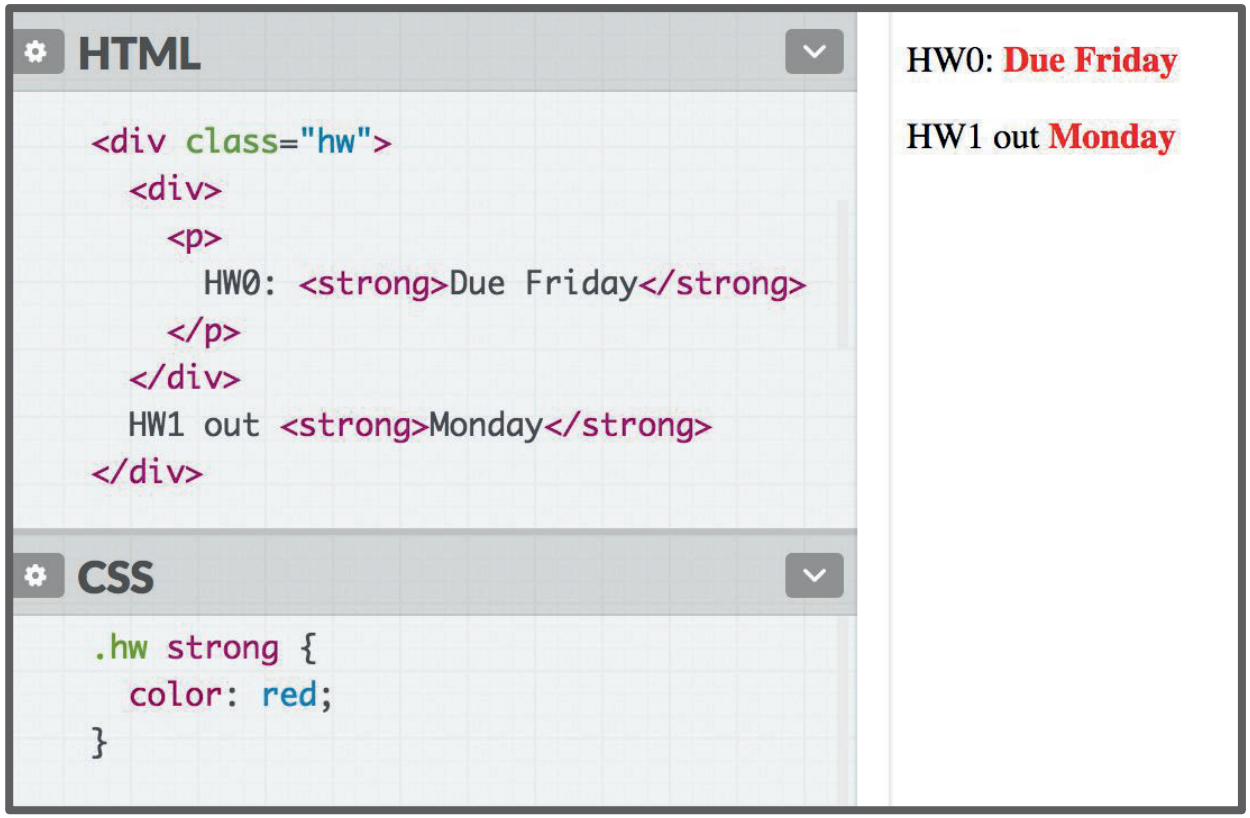
```
.hw p {
  color: green;
}
```



Descendent selector

Syntax	Example	Example described
selector selector	`div strong`	`<strong>` elements that are descendants of a `<div>`

Note: The element does not have to be a direct child. The descendent may be nested many layers in.



Descendent selector

Syntax	Example	Example described
selector selector	`div strong`	**** elements that are descendants of a ** **

Discouraged:

```
<h1 class="hw">Homework 0</h1>
<p class="hw">Due Fri</p>
<p class="hw">Late cutoff Sun</p>
```

vs

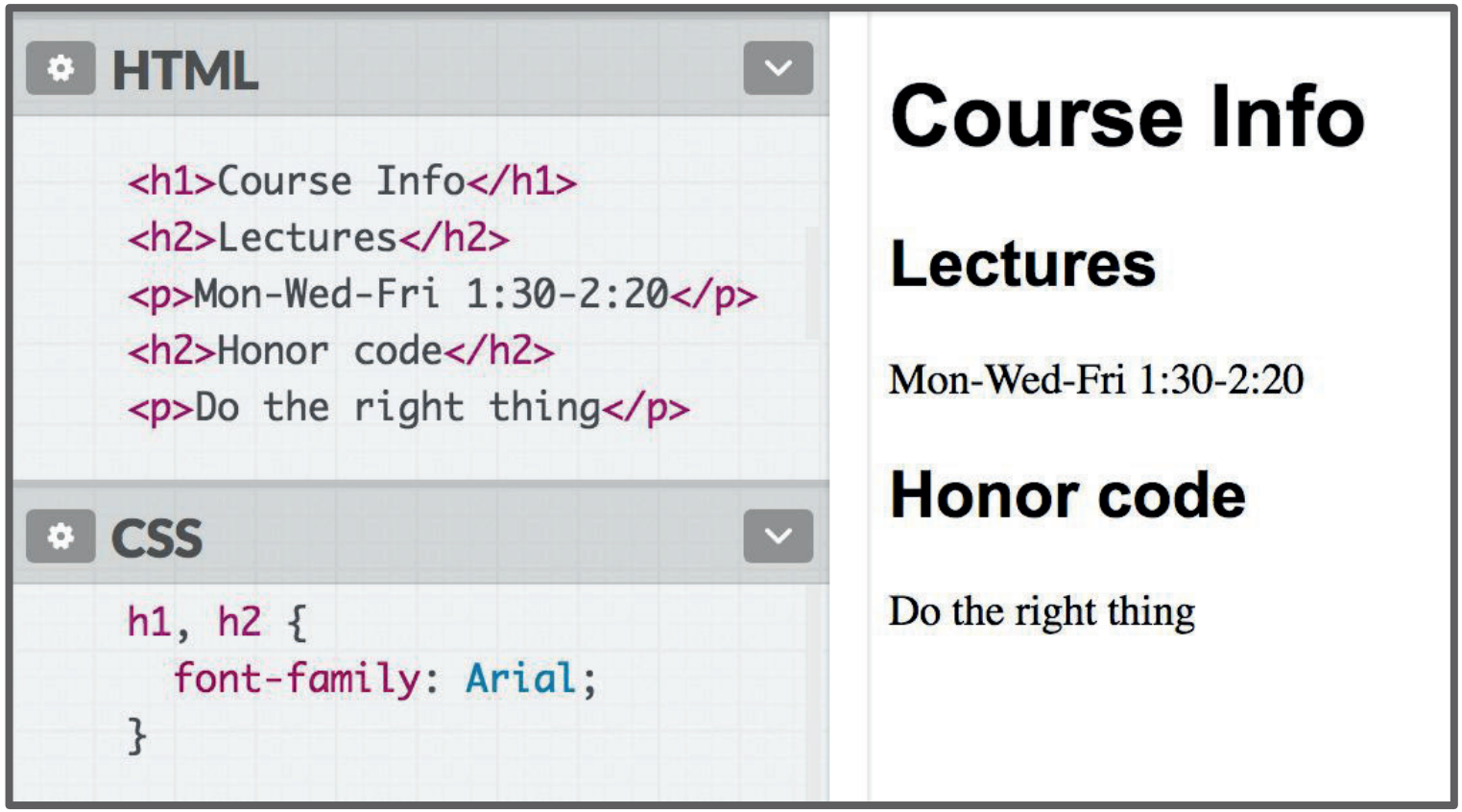
Preferred:

```
<div class="hw">
  <h1>Homework 0</h1>
  <p>Due Fri</p>
  <p>Late cutoff Sun</p>
</div>
```

Instead of applying a class to several adjacent elements, wrap the group in a `<div>` container and style the contents via descendent selectors.

selector, selector (comma)

Syntax	Example	Example described
selector, selector	`h2, div`	`<h2>` elements and `<div>`s



Selector summary

Example	Example described
`p`	All **<p>** elements
`.abc`	All elements with the **abc class**, i.e. **class="abc"**
`#abc`	Element with the **abc id**, i.e. **id="abc"**
`p.abc`	**<p>** elements with **abc class**
`p#abc`	**<p>** element with **abc id** (**p** is redundant)
`div strong`	**** elements that are descendants of a ** **
`h2, div`	**<h2>** elements and ** **s

Grouping selectors

2 Common bugs:

`p.abc` **vs** `p .abc`

`p .abc` **vs** `p, .abc`

- A <p> element with the **abc** class **vs**
  An element with the **abc** class that descends from <**p**>

- An element with the **abc** class that descends from <**p**> **vs**
  All <**p**> elements *and* all elements with the **abc** class

Combining selectors

You can combine selectors:

```
#main li.important strong {
  color: red;
}
```

Q: What does this select?

Grouping selectors

Q: What does this select?

```
#main li.important strong {
  color: red;
}
```

A: Read from right to left:

- `<strong>` tags that are children of `<li>` tags that have an "`important`" class that are children of the element with the "`main`" id.

Colliding styles

When styles collide, the most specific rule wins (specificity)

```
div strong { color: red; }
strong { color: blue; }

<div>
   <strong>What color am I?</strong>
</div>
```

Colliding styles

When styles collide, the most specific rule wins (specificity)

```
div strong { color: red; }
strong { color: blue; }

<div>
   <strong>What color am I?</strong>
</div>
```

Colliding styles

Specificity precedence rules (details):

- `id`s are more specific than `class`es
- `class`es are more specific than element names
- elements are more specific than children of those elements

Colliding styles

- If elements have the same specificity, the later rule wins.

```
strong { color: red; }
strong { color: blue; }

<div>
   <strong>What color am I?</strong>
</div>
```

Aside: The process of figuring out what rule applies to a given element is called the cascade. This is where the "C" in *Cascading* Style Sheets comes from.

Inheritance

We saw earlier that CSS styles are inherited from parent to child.

Instead of selecting all elements individually:

```
a, h1, p, strong {
  font-family: Helvetica;
}
```

You can style the parent and the children will inherit the styles.

```
body {
  font-family: Helvetica;
}
```

You can override this style via specificity:

```
h1, h2 {
  font-family: Consolas;
}
```

Inheritance

While many CSS styles are inherited from parent to child, **not all CSS properties are inherited**.

```
a {
  display: block;
  font-family: Arial;
}
```

`<em>` inherits the `font-family` property, but not `display:`

```
<a href="/home">
  Back to <em>Home</em>
</a>
```

Back to *Home*

Inheritance

While many CSS styles are inherited from parent to child, **not all CSS properties are inherited**.

- There's no rule for what properties are inherited or not; the inheritance behavior defined in the CSS spec.
- You can look it up via MDN, e.g.
  `font-family`: Inherited yes
  `display`: Inherited no
- Generally text-related properties are inherited and layout-related properties are not.
- (You can also change this via the `inherit` CSS property, which is somewhat esoteric and not often use)