CSS Cont'd: Cascading Style Sheets

Lecture 17

Classes

- Not all paragraphs created equally
  - Some paragraphs are not finalized (draft), so want them styled differently
- Solution: class attribute

```
<p class="draft">… </p>
```

- CSS syntax for selector: ***elt.class***

```
p.draft { color: gray; }
```

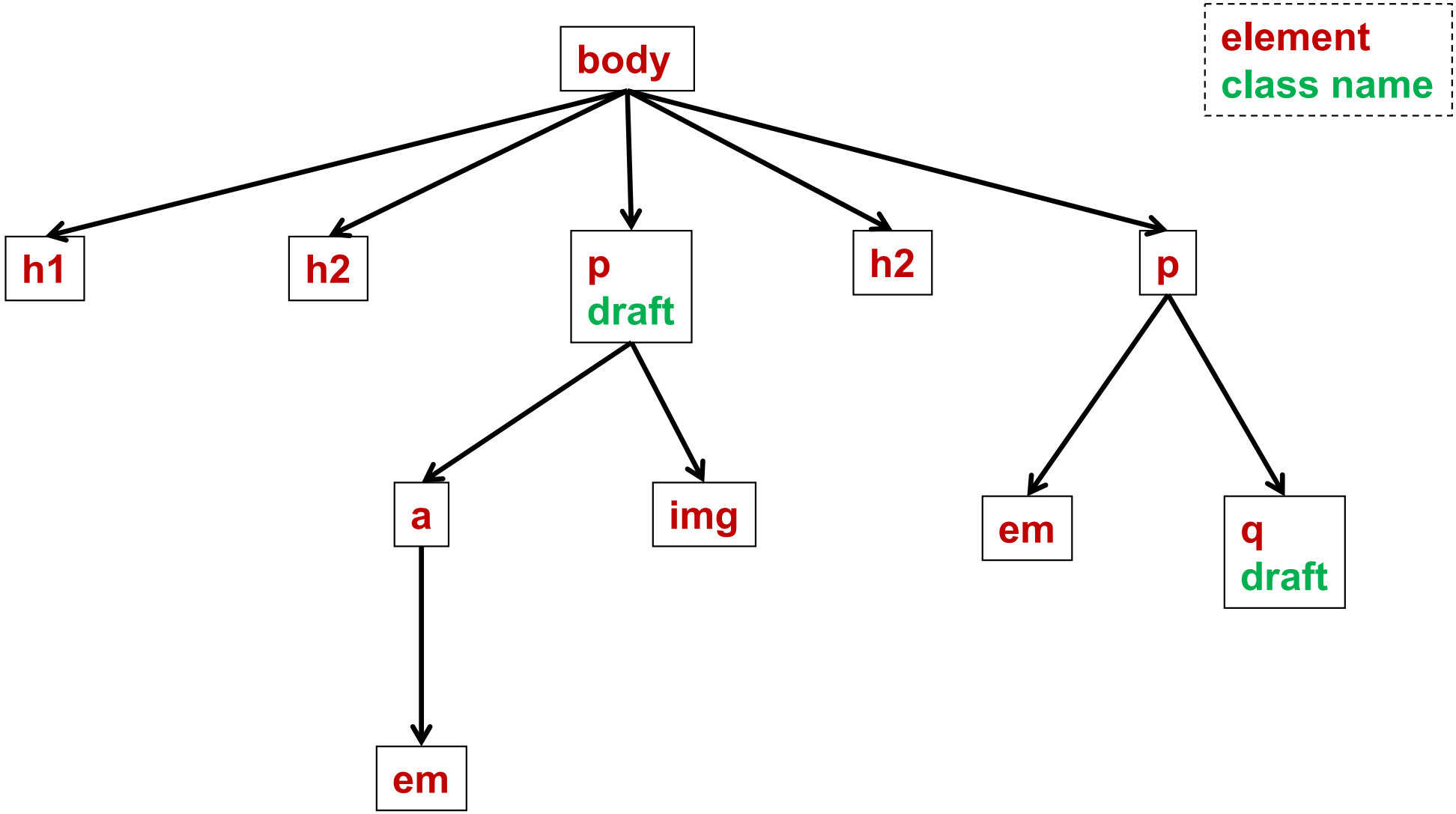
- Wildcard (any element): ***.class***

```
.draft { font-style: italic; }
```

- An element can be in multiple classes
  - Recall: attributes are a map, ie names unique

```
<p class="draft even">… </p>
```

Classes Add to Tree Structure



Notes on Classes

- When an element belongs to multiple classes, which style gets applied?
  - Different properties are combined (union)
  - Conflicts on same property need to be resolved (more later)
- Classes should reflect semantics or structure, not visual formatting
  - Bad class name: green
  - Good class name: draft
- Example: css-classes

Problem

- Multiple block elements that need to be styled together
  - Example: Header and paragraph(s) are both part of the same warning

```
<h2 class="warning">…</h2>

<p class="warning"> … </p>
```

- This approach is awkward
  - Every block element in group needs to be decorated in this way
  - Difficult to style the entire unit (*e.g.*, add a border around the whole warning)

Solution: Div Element

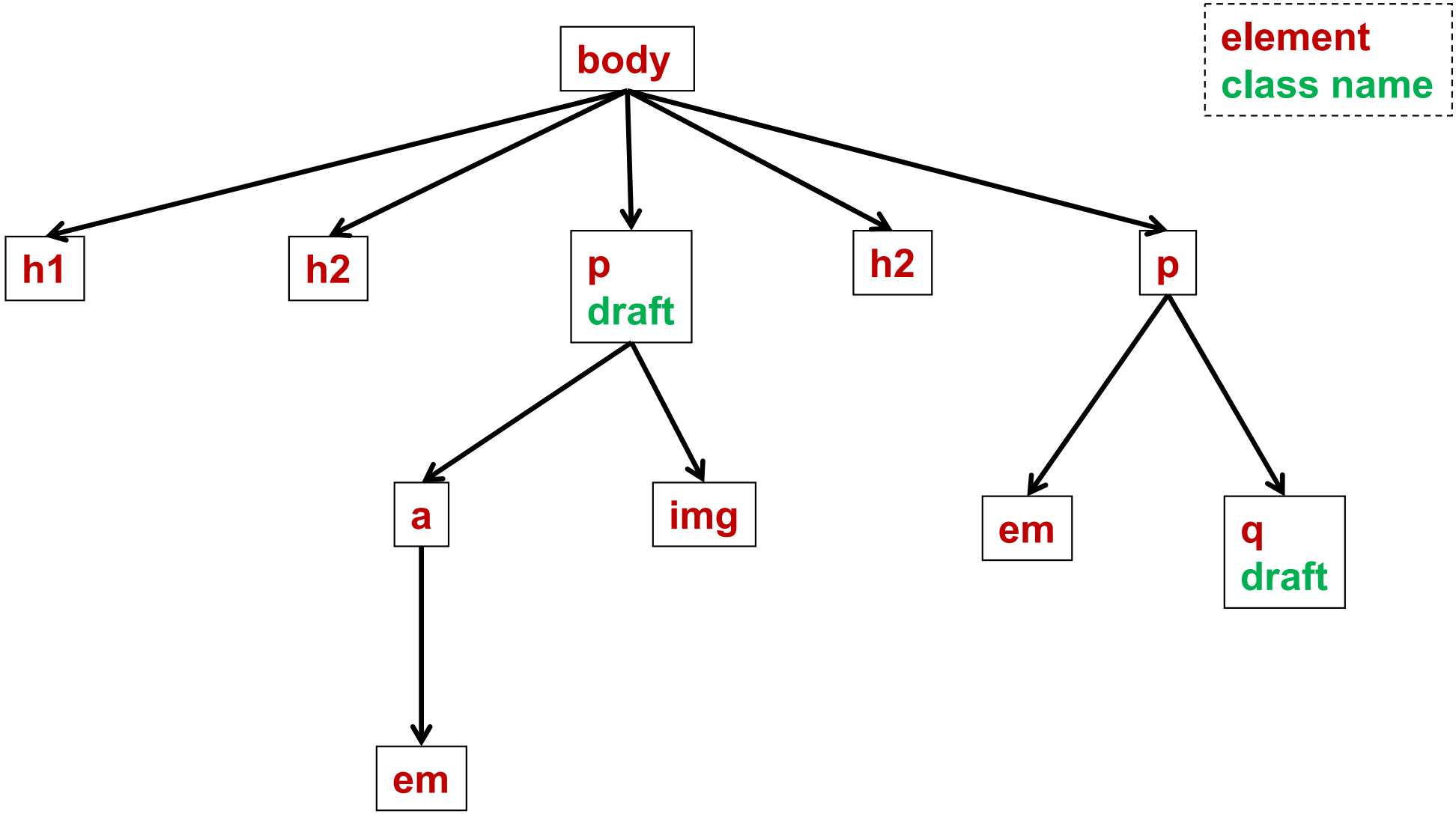
- **`div`** gives a *logical* block element
- Can be styled just like any other block element
  - Font, dimension, border, margin, etc

```
.warning { border: thick; }
```

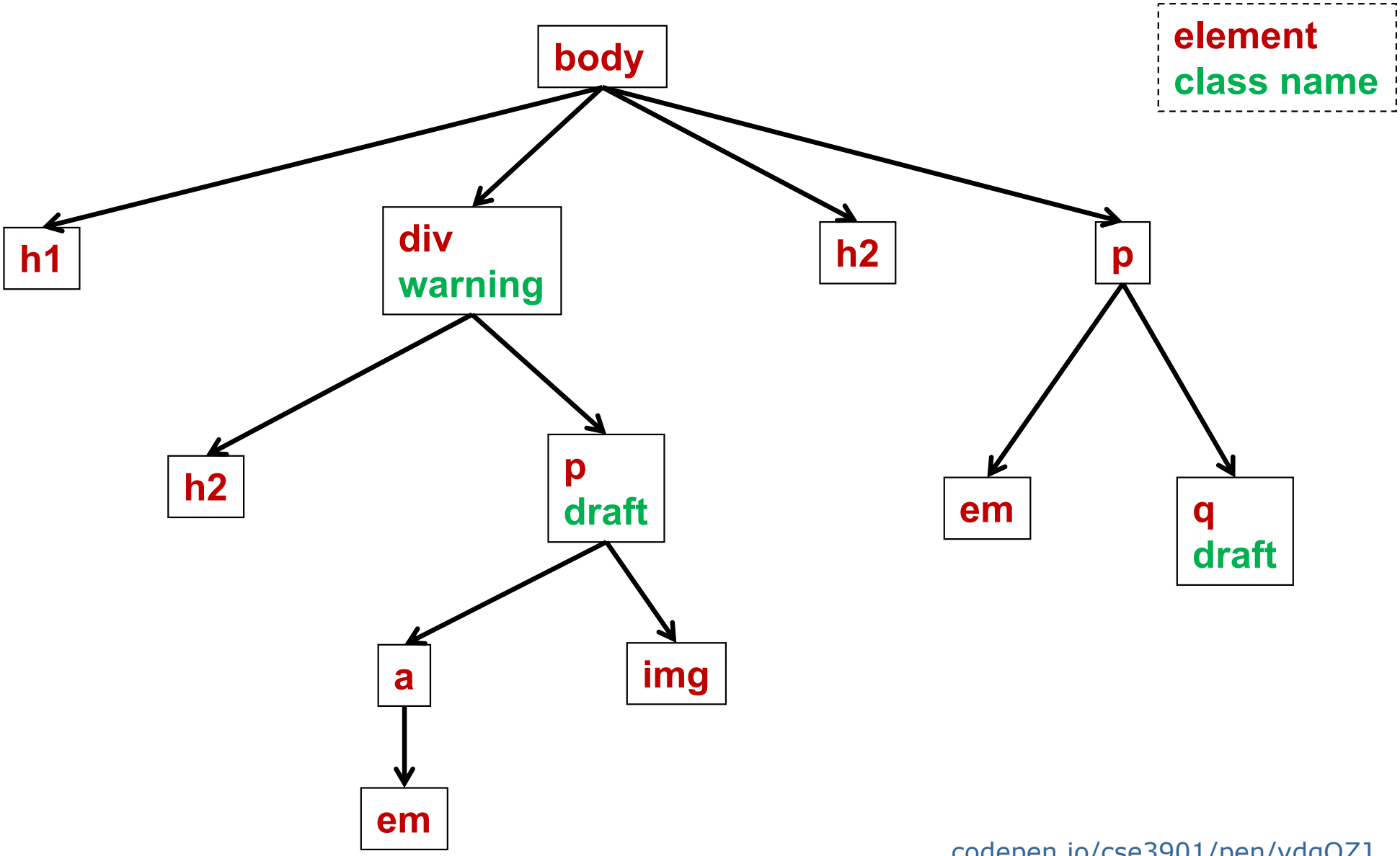
- Can have block elements as children
  - Style inherited by children

```
<div class="warning">
  <h2> … </h2>
  <p> … </p>
</div>
```

Original Tree



Divs in the Tree



codepen.io/cse3901/pen/vdgQZJ

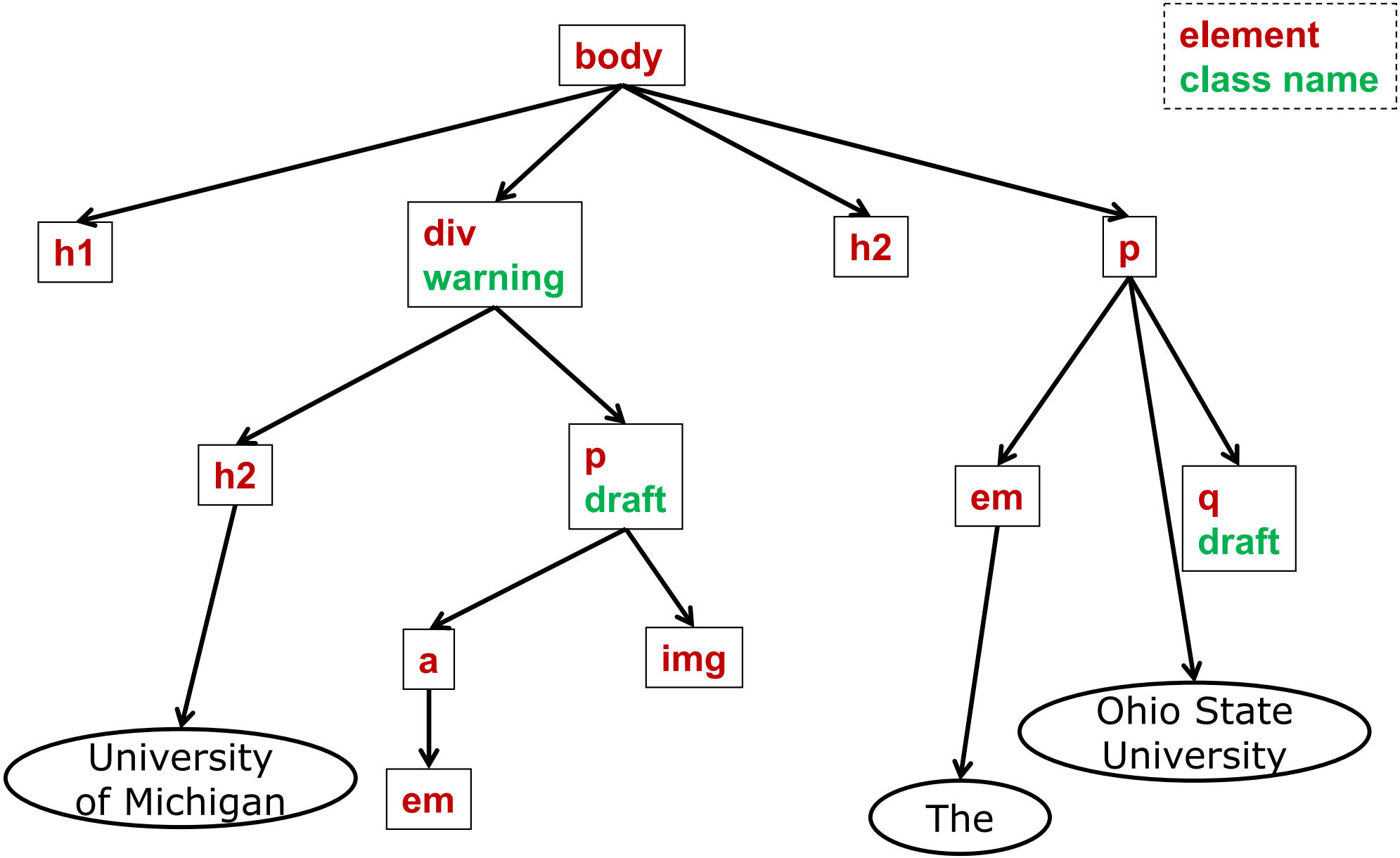
Span Element

- **`div`** is a (logical) block level element
  - Gives line breaks
- Sometimes styling/semantics belongs to *inline* elements
  - Text discussing different textbooks, where titles appear here and there
- Solution: **`span`** tag

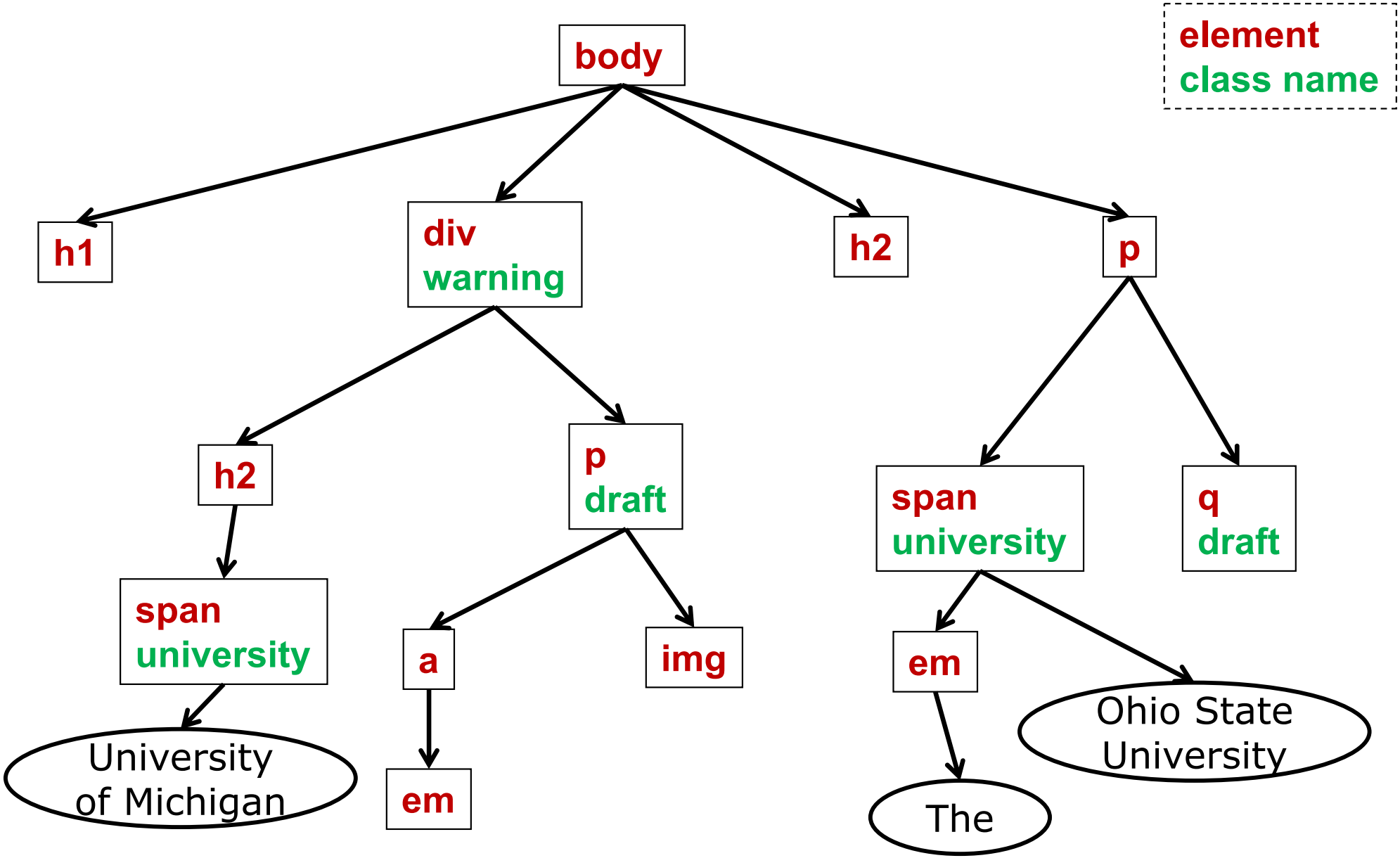
```
<p> One book to consider is the
<span class="book">Book of Ruby</span>, …
```

- Now all book titles can be styled consistently
- Like **`div`**, **`span`** is often used with classes

Original Tree



Adding Spans to the Tree



Ancestors in Selectors

- Sometimes you care about *where* in the tree an element occurs
  - University names appearing *somewhere inside* warnings need a different styling
- CSS syntax: `ancestor ancestor… elt`

```
.warning .university
```

- Note: *big* difference between

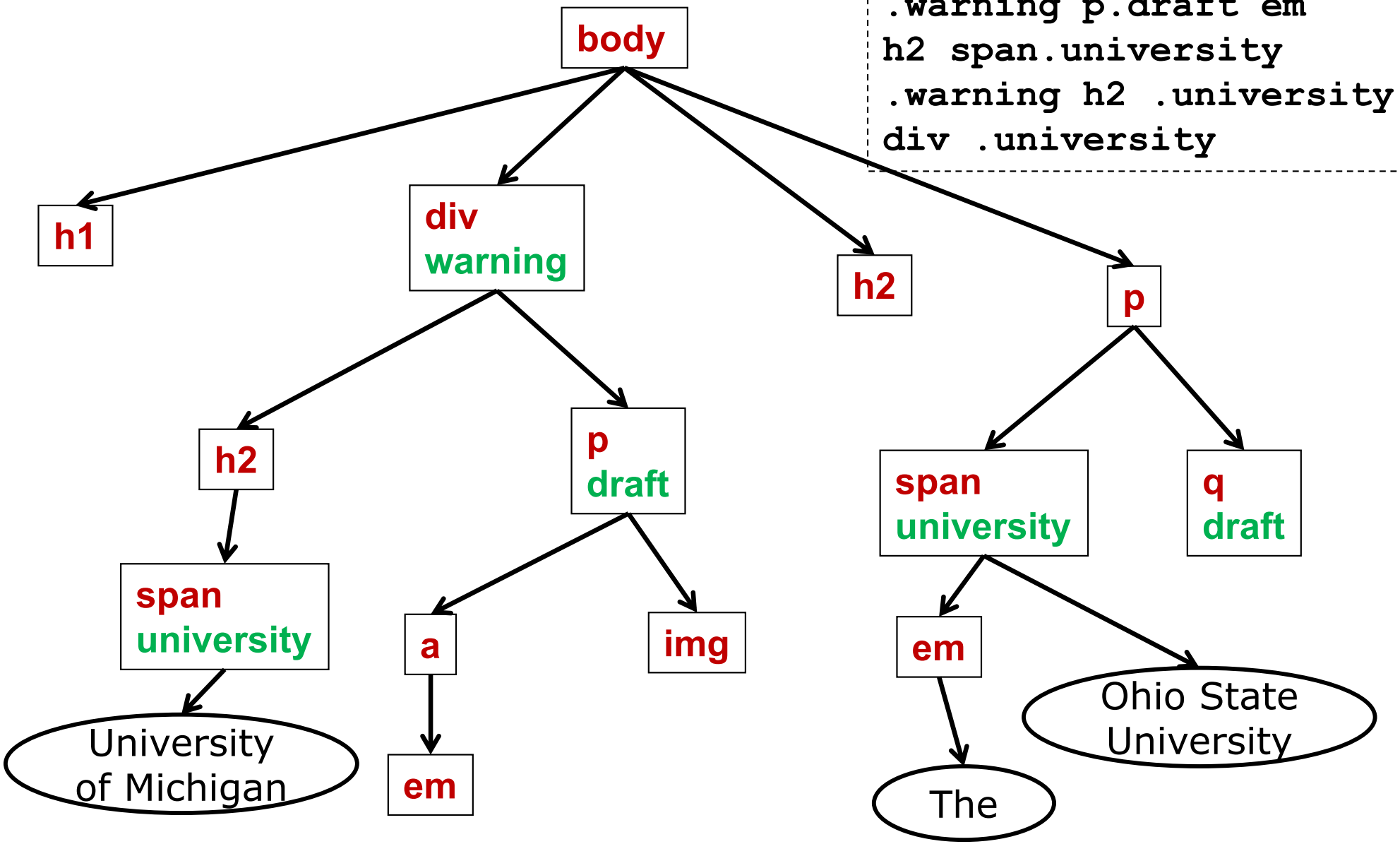
```
.warning em .university
```

```
.warning em, .university
```

```
.warning, em .university
```

Your Turn


```
.warning p.draft em
h2 span.university
.warning h2 .university
div .university
```



More Exotic Paths in Selectors

- Child: **>**

```
.warning > p
.warning li > em
```

- Adjacent sibling: **+**

```
h1 + p /* only first p after h1 */
```

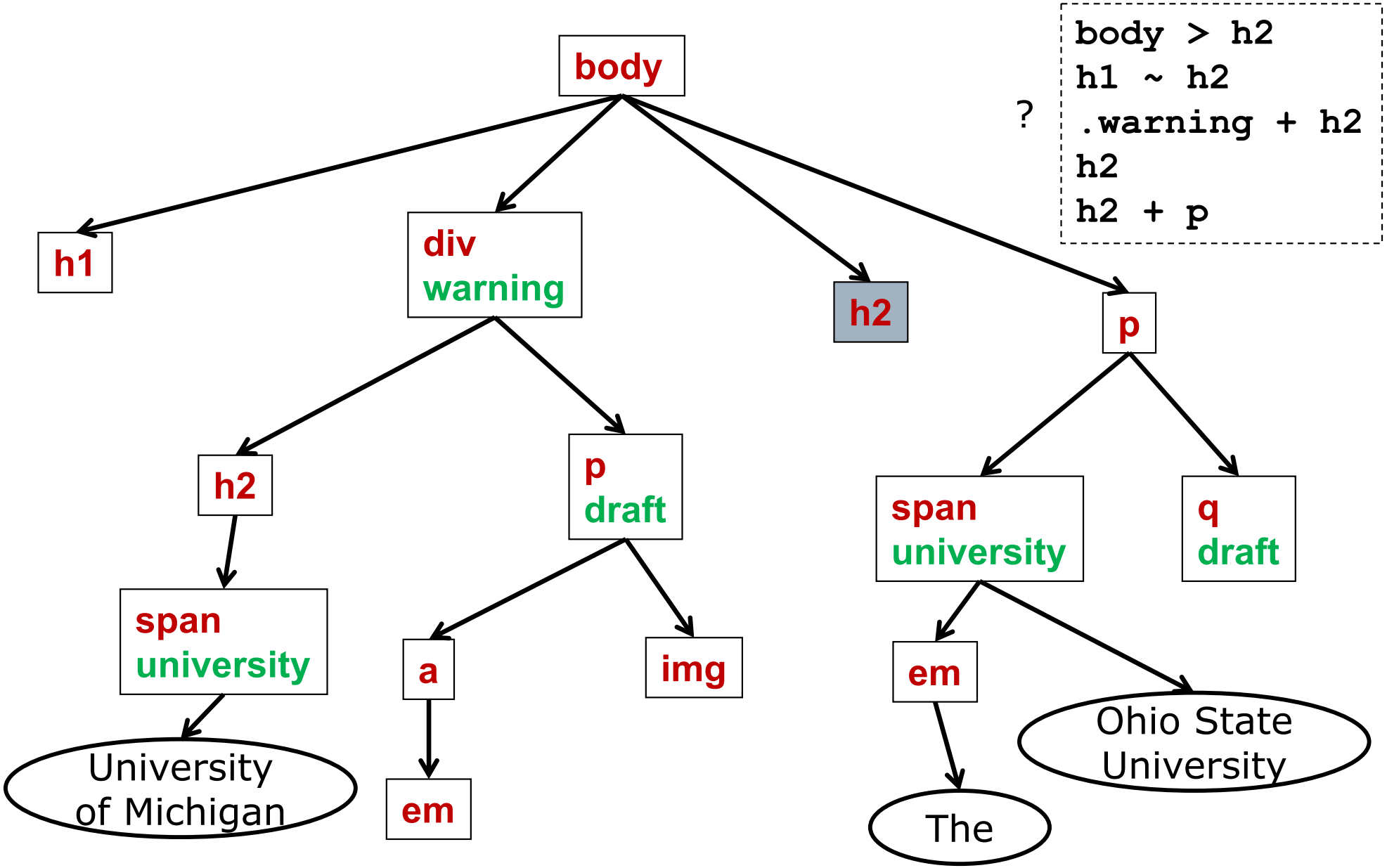
- General sibling: **~**

```
h1 ~ p /* sibling p's after h1 */
```

- Attributes: **[*attr*="*value*"], *=, $=**

```
input[type="button"]
a[href$=".pdf"] /* class website */
```

Your Turn: Select Shaded Node



Id = Class Plus Two Invariants

- Some classes are meant to be unique
  - *At most one* such element per page

```
<div class="sponsors">
```

- Solution: id attribute

```
<div id="sponsors">
```

- CSS syntax for selector: ***elt#id***

```
p#sponsors { color: red; }
```

- Wildcard (any element): ***#id***

```
#headline { box-style: thin; }
```

- An element can have *at most one* id

Scraping With Selectors

- Nokogiri: A Ruby gem for parsing and scraping HTML
  - Given CSS selector, returns matching elements in page
  - Returns a NodeSet, which acts like an array

```
agent = Mechanize.new
page = agent.get
          'http://www.cse.osu.edu/news'
news = page.css 'h2.content-headline'
news.class #=> Nokogiri::XML::NodeSet
news.size #=> 11
news.each { |title| puts title.text }
```

Summary

- Classes and Ids
  - Class gives an extra dimension to tree
  - ID is unique: at most one per page
  - CSS selector syntax (. vs #)
- Divs and Spans
  - Div is a logical block element
  - Span is a logical inline element
  - Often used together with classes/ids
- Selectors with ancestors, siblings
  - CSS selector syntax (space, >, +, ~)