Cascading Style Sheets

CITS3403 and CITS5505 - Agile Web Development

Unit Coordinator: Maira Alvi

2025 Semester 1

CSS basics

What is CSS

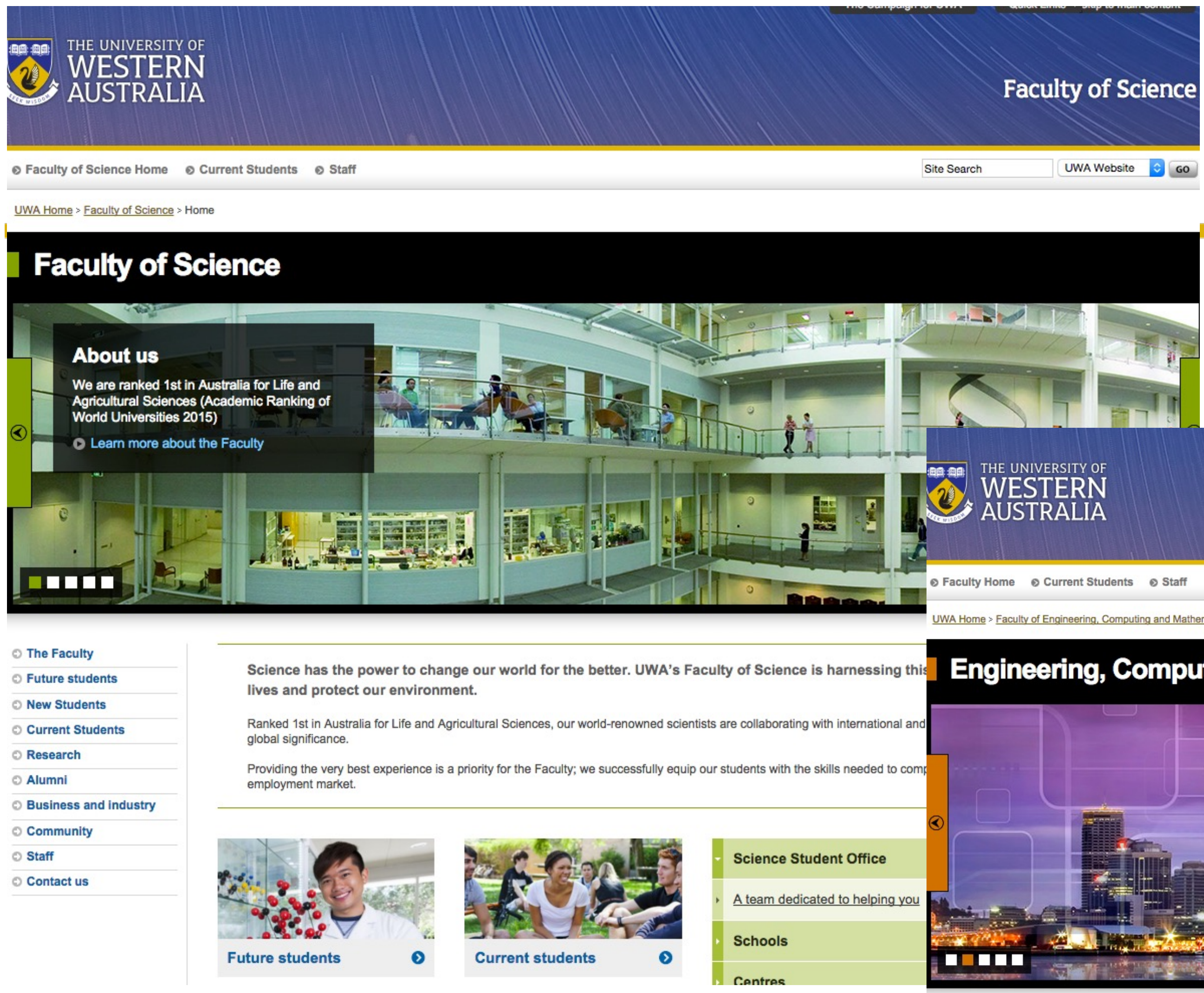
- CSS stands for *Cascading Style Sheets*
  - a stylesheet language for the web
  - used to specify the presentation (layout and style) of markup languages
  - can be applied to any XML document as well as HTML.
  - superseded many HTML attributes that mixed presentation with content

```css
body {
    background-color: lightblue;
}
h1 {
    color: white;
    text-align: center;
}
p {
    font-family: verdana;
    font-size: 20px;
}
```

Advantages of CSS

- Separation of content and presentation

- Advantages for the web
  1. *Speed* - stylesheet(s) downloaded once, rather than with each page (if content and style information is intermingled).
  2. *Maintainability* - can be “centrally” maintained, easier to update
  3. *Accessibility* - pages appear similar on different browsers and devices.
  4. *Portability* – consistent styling across all devices supporting browsers.
  5. *Reduced work* – e.g., don’t have to specify alignment for every element.
  6. *Consistency* - make an organisation’s web pages have a consistent “look and feel” that matches the corporate ID, brand - e.g., UWA…



THE UNIVERSITY OF WESTERN AUSTRALIA

Faculty of Science

Faculty of Science Home | Current Students | Staff

Site Search | UWA Website | GO

UWA Home > Faculty of Science > Home

Faculty of Science

About us

We are ranked 1st in Australia for Life and Agricultural Sciences (Academic Ranking of World Universities 2015)

Learn more about the Faculty

- The Faculty
- Future students
- New Students
- Current Students
- Research
- Alumni
- Business and industry
- Community
- Staff
- Contact us

Science has the power to change our world for the better. UWA's Faculty of Science is harnessing this lives and protect our environment.

Ranked 1st in Australia for Life and Agricultural Sciences, our world-renowned scientists are collaborating with international and global significance.

Providing the very best experience is a priority for the Faculty; we successfully equip our students with the skills needed to comp employment market.

Future students

Current students

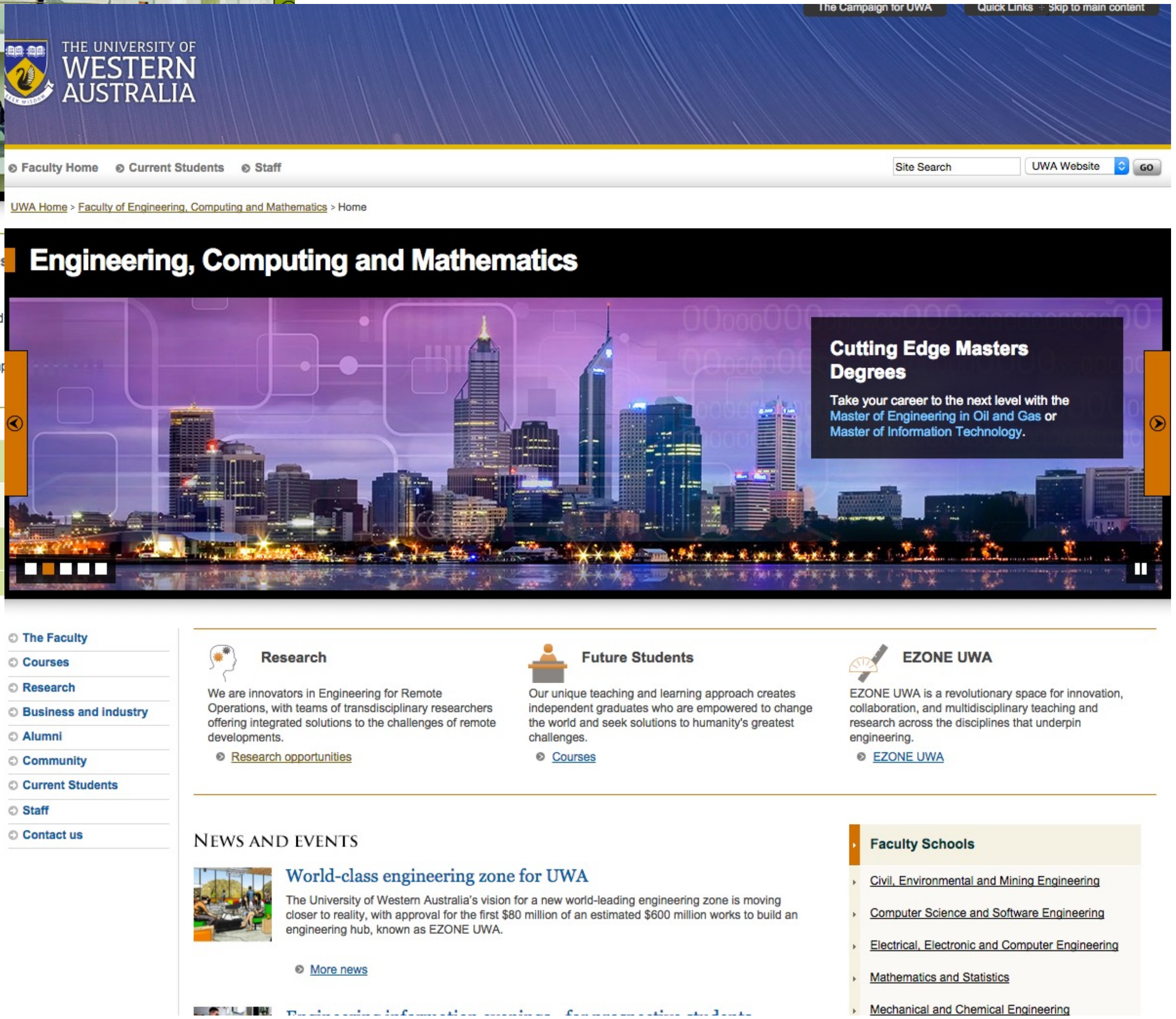
Science Student Office

A team dedicated to helping you

Schools

Centres

THE UNIVERSITY OF WESTERN AUSTRALIA



The Campaign for UWA | Quick Links | Skip to main content

THE UNIVERSITY OF WESTERN AUSTRALIA

Faculty Home | Current Students | Staff

Site Search | UWA Website | GO

UWA Home > Faculty of Engineering, Computing and Mathematics > Home

Engineering, Computing and Mathematics

Cutting Edge Masters Degrees

Take your career to the next level with the Master of Engineering in Oil and Gas or Master of Information Technology.

- The Faculty
- Courses
- Research
- Business and industry
- Alumni
- Community
- Current Students
- Staff
- Contact us

Research

We are innovators in Engineering for Remote Operations, with teams of transdisciplinary researchers offering integrated solutions to the challenges of remote developments.

Research opportunities

Future Students

Our unique teaching and learning approach creates independent graduates who are empowered to change the world and seek solutions to humanity's greatest challenges.

Courses

EZONE UWA

EZONE UWA is a revolutionary space for innovation, collaboration, and multidisciplinary teaching and research across the disciplines that underpin engineering.

EZONE UWA

NEWS AND EVENTS

World-class engineering zone for UWA

The University of Western Australia's vision for a new world-leading engineering zone is moving closer to reality, with approval for the first $80 million of an estimated $600 million works to build an engineering hub, known as EZONE UWA.

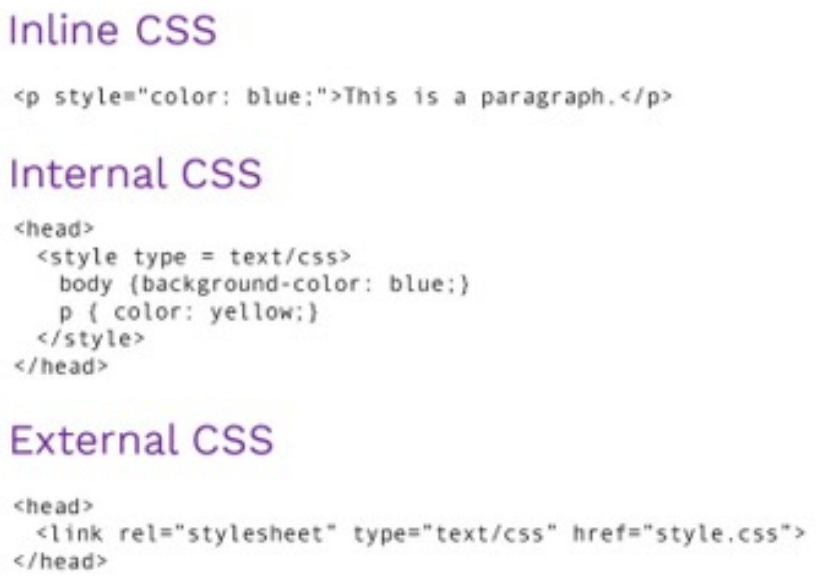
More news

Faculty Schools

- Civil, Environmental and Mining Engineering
- Computer Science and Software Engineering
- Electrical, Electronic and Computer Engineering
- Mathematics and Statistics
- Mechanical and Chemical Engineering

Why "cascading"?

- There are three levels of style sheets
  - *Inline* styles – applies to a single tag only.
  - *Document* style sheets - appears in the document's <head> element and applies to the whole document.
  - *External* style sheets - separate files, potentially on any server on the Internet, and can be applied to any number of documents in their <head> element.



Inline CSS

```
<p style="color: blue;">This is a paragraph.</p>
```

Internal CSS

```
<head>
  <style type = text/css>
    body {background-color: blue;}
    p { color: yellow;}
  </style>
</head>
```

External CSS

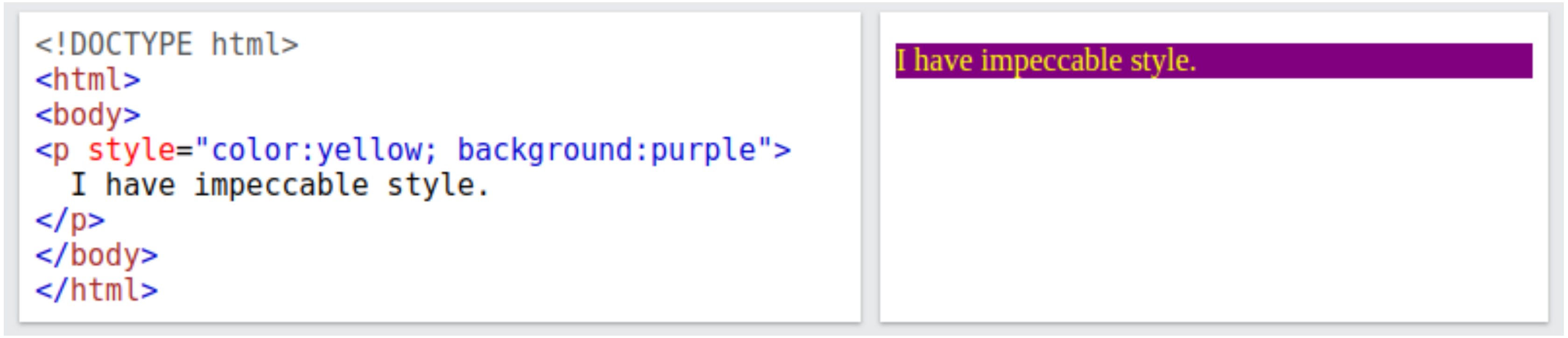
```
<head>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
```

Inline styles

- Style specification appears as the value of the `style` attribute of almost any tag.
  - General form:

```
style="property_1: value_1;
       property_2: value_2;
       …
       property_n: value_n"
```

  - Example:



```
<!DOCTYPE html>
<html>
<body>
<p style="color:yellow; background:purple">
  I have impeccable style.
</p>
</body>
</html>
```

I have impeccable style.

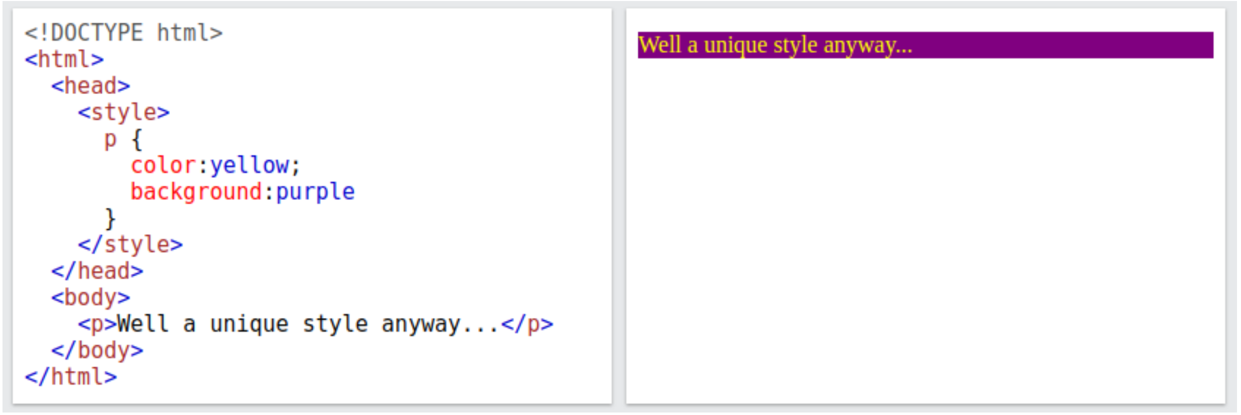
- Warning: Inline styles defeats the purpose of style sheets i.e. uniform style. Use it for debugging but not much else!

Document styles

- Style specification appears as a list of rules that are the *content* of a `<style>` tag contained in the document `<head>`.

- Specification form:

```
<style>
    rule1
    rule2
    ...
</style>
```



```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p {
        color:yellow;
        background:purple
      }
    </style>
  </head>
  <body>
    <p>Well a unique style anyway...</p>
  </body>
</html>
```

Well a unique style anyway...

- Rule form:

```
selector {
    property1:value1;
    property2:value2;
    ...
}
```

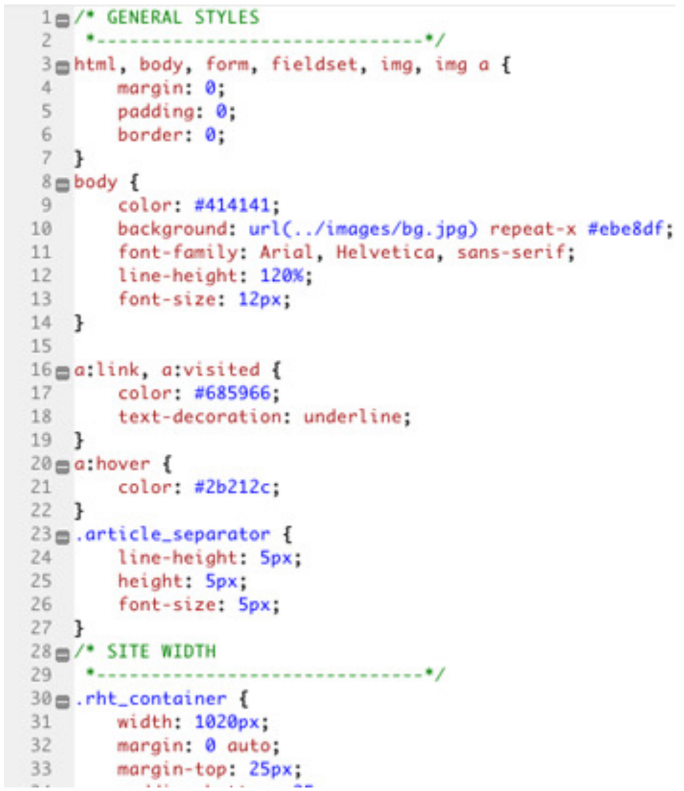
- Property-value pairs are separated by semicolons, just as in the value of a style attribute.

External styles

- A <link> tag inside <head> is used to specify that the browser is to fetch and use an external style sheet file, E.g. Wikipedia style sheet http://en.wikipedia.org//skins-1.5/common/shared.css?165

```
<link
 rel="stylesheet”
 type="text/css”
 href="http://tiny.url/some.css">
</link>
```

- The format is identical to the contents of a <style> tag for document-level style sheets.



```
/* GENERAL STYLES
 *------------------------------*/
html, body, form, fieldset, img, img a {
    margin: 0;
    padding: 0;
    border: 0;
}
body {
    color: #414141;
    background: url(../images/bg.jpg) repeat-x #ebe8df;
    font-family: Arial, Helvetica, sans-serif;
    line-height: 120%;
    font-size: 12px;
}

a:link, a:visited {
    color: #685966;
    text-decoration: underline;
}
a:hover {
    color: #2b212c;
}
.article_separator {
    line-height: 5px;
    height: 5px;
    font-size: 5px;
}
/* SITE WIDTH
 *------------------------------*/
.rht_container {
    width: 1020px;
    margin: 0 auto;
    margin-top: 25px;
```

CSS selectors

Selector basics

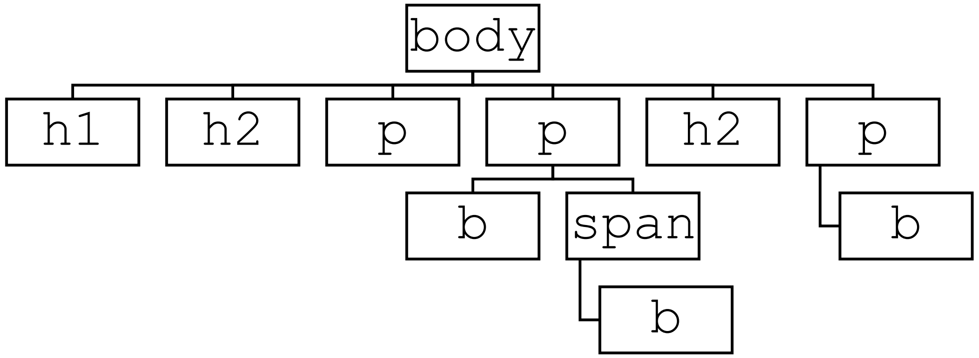
- A selector determines which elements the style applies to.
- There is a whole language for writing increasingly precise selectors.
- There are broadly two types, basic selectors that allow selection based one specific criteria, and combinators that then allow one to join multiple criteria together in various ways.
- Writing sets of CSS selectors that scale to large webpages and don't clash with each other is a skill!

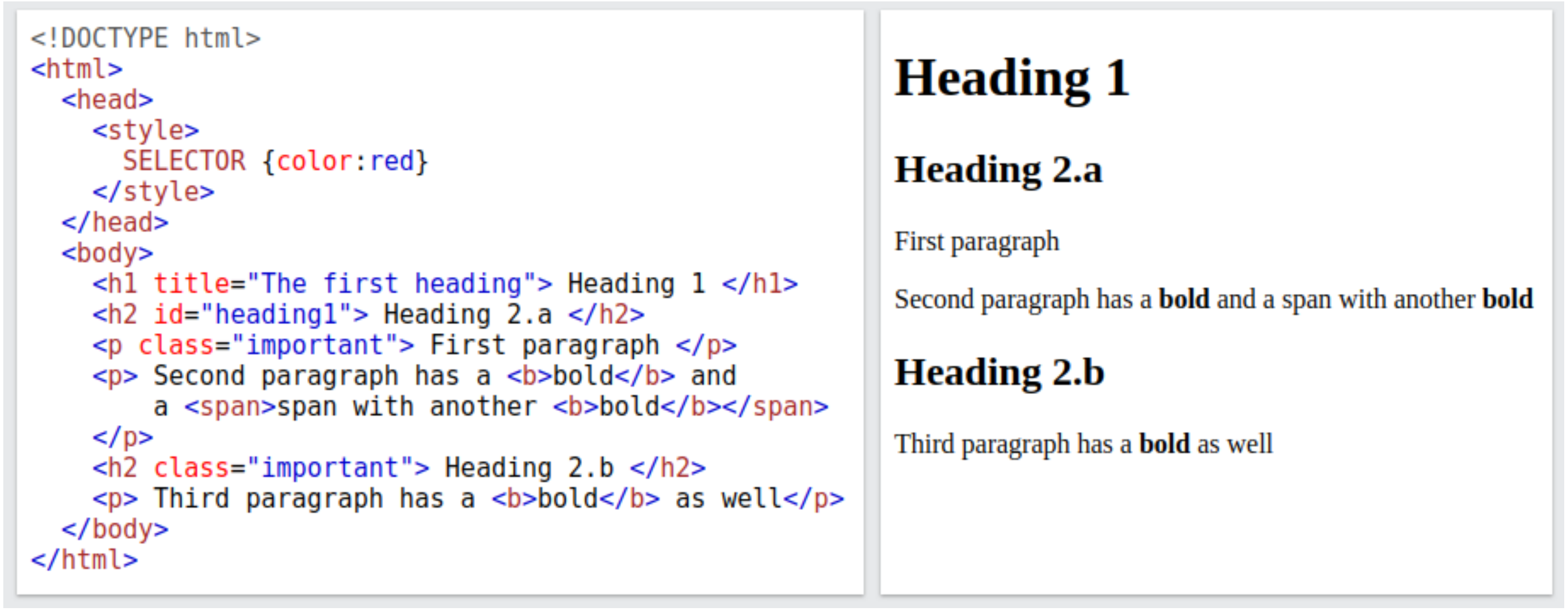
Basic selectors	Example
Universal selector	* {…}
Element selector	p {…}
Attribute selector	[secret="yes"] {…}
Class selector	.important {…}
ID selector	#1234 {…}
Pseudo-class selector	:onhover {…}
Pseudo-element selector	::first-letter {…}

Selector combinators	Example
Group selector	s1,s2,s3 {...}
Descendant selector	s1 s2 {...}
Direct descendant selector	s1 > s2 {...}
Sibling selector	s1 ~ s2 {...}
Direct sibling selector	s1 + s2 {...}

Selector example

- To illustrate we will use the following HTML and CSS, replacing the "SELECTOR" as appropriate.





```
<!DOCTYPE html>
<html>
  <head>
    <style>
      SELECTOR {color:red}
    </style>
  </head>
  <body>
    <h1 title="The first heading"> Heading 1 </h1>
    <h2 id="heading1"> Heading 2.a </h2>
    <p class="important"> First paragraph </p>
    <p> Second paragraph has a <b>bold</b> and
        a <span>span with another <b>bold</b></span>
    </p>
    <h2 class="important"> Heading 2.b </h2>
    <p> Third paragraph has a <b>bold</b> as well</p>
  </body>
</html>
```

Heading 1

Heading 2.a

First paragraph

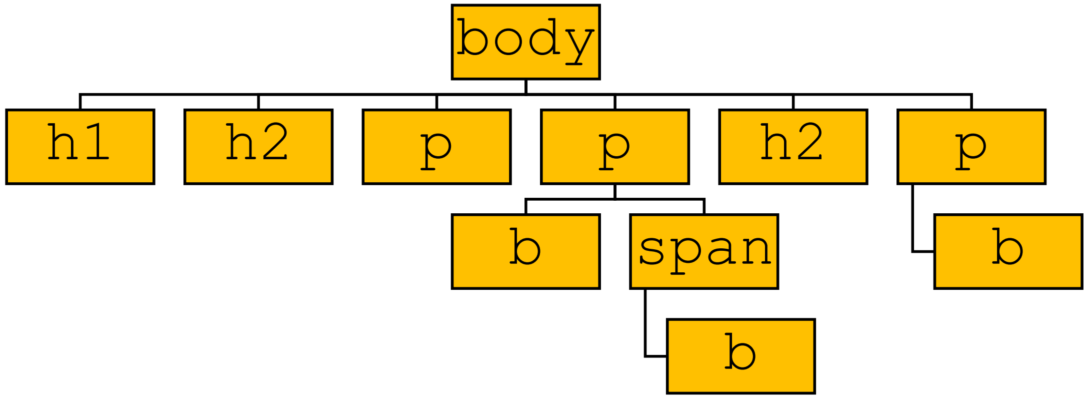
Second paragraph has a **bold** and a span with another **bold**

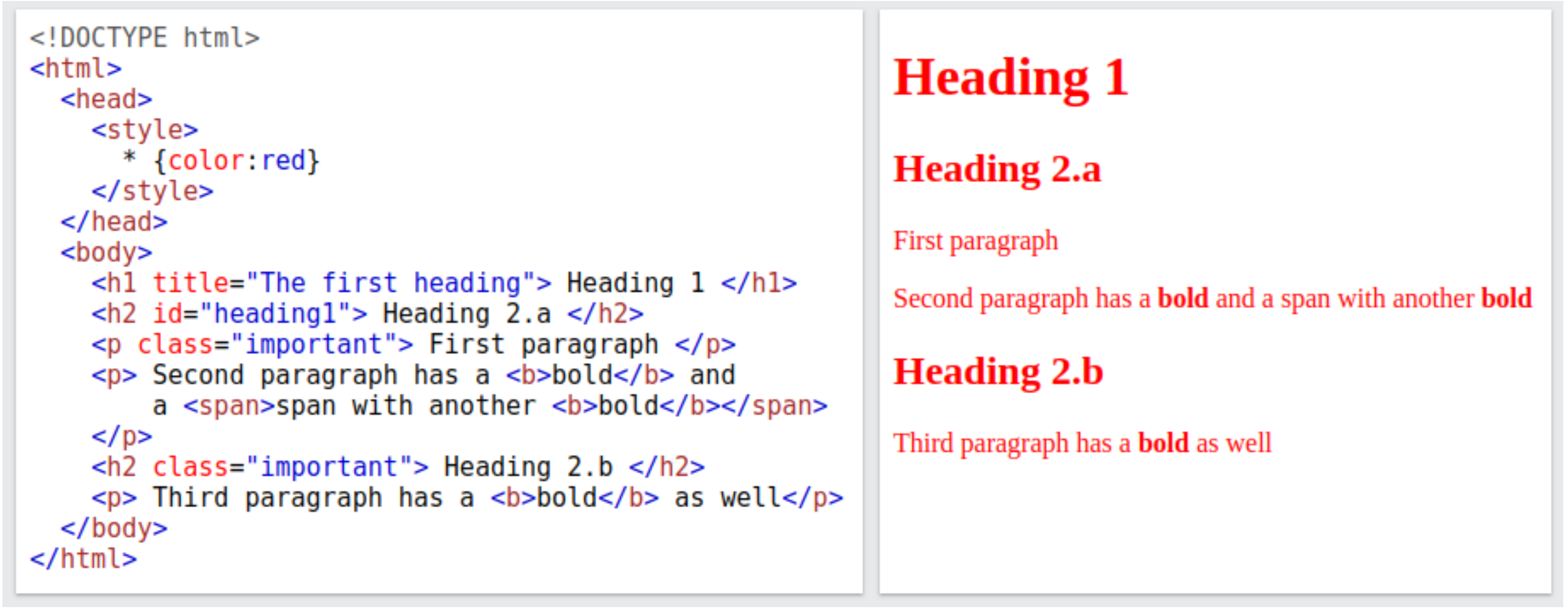
Heading 2.b

Third paragraph has a **bold** as well

Universal selector: *

- The universal selector * matches all elements.
- Matching on * results in:





```
<!DOCTYPE html>
<html>
  <head>
    <style>
      * {color:red}
    </style>
  </head>
  <body>
    <h1 title="The first heading"> Heading 1 </h1>
    <h2 id="heading1"> Heading 2.a </h2>
    <p class="important"> First paragraph </p>
    <p> Second paragraph has a <b>bold</b> and
        a <span>span with another <b>bold</b></span>
    </p>
    <h2 class="important"> Heading 2.b </h2>
    <p> Third paragraph has a <b>bold</b> as well</p>
  </body>
</html>
```

Heading 1

Heading 2.a

First paragraph

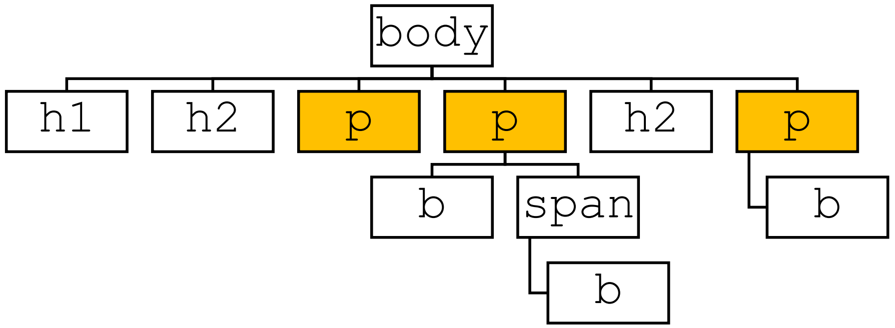
Second paragraph has a **bold** and a span with another **bold**

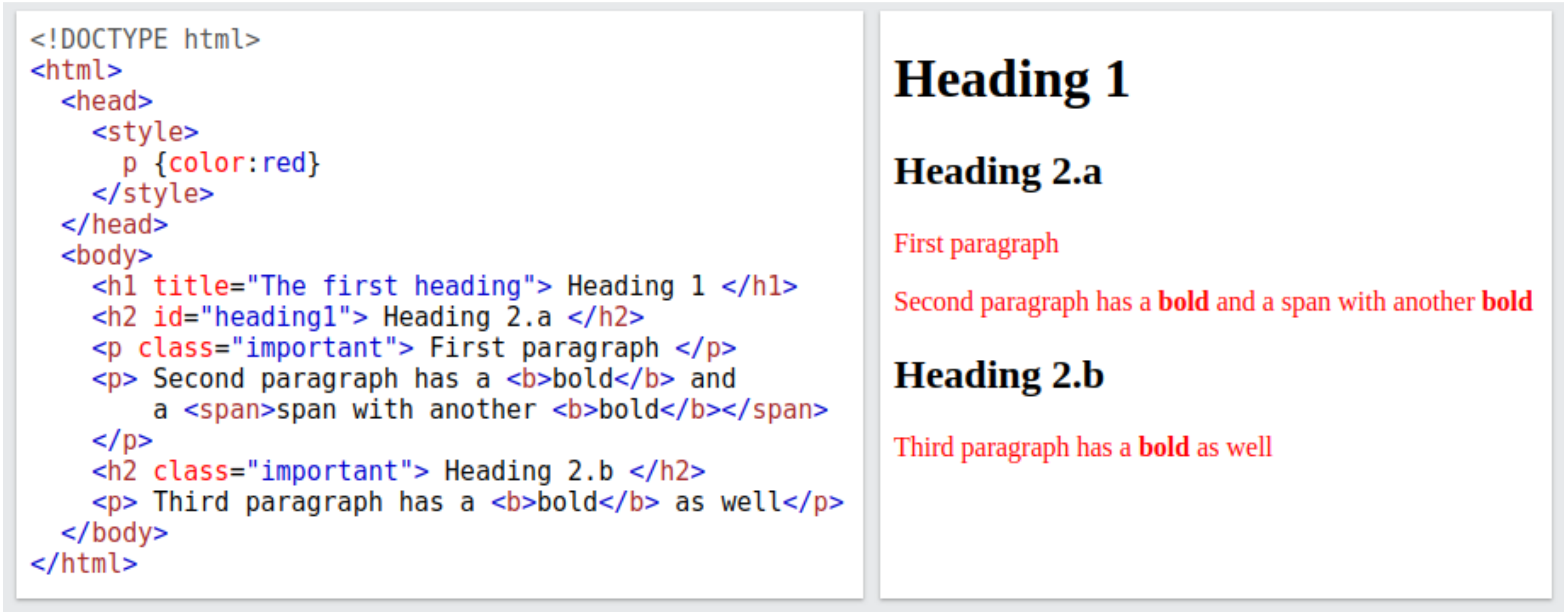
Heading 2.b

Third paragraph has a **bold** as well

Element selector: *element*

- An element selector matches one specific type of element.
- For example, matching on `p` results in all `<p>` elements being highlighted.





```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p {color:red}
    </style>
  </head>
  <body>
    <h1 title="The first heading"> Heading 1 </h1>
    <h2 id="heading1"> Heading 2.a </h2>
    <p class="important"> First paragraph </p>
    <p> Second paragraph has a <b>bold</b> and
        a <span>span with another <b>bold</b></span>
    </p>
    <h2 class="important"> Heading 2.b </h2>
    <p> Third paragraph has a <b>bold</b> as well</p>
  </body>
</html>
```

Heading 1

Heading 2.a

First paragraph

Second paragraph has a **bold** and a span with another **bold**

Heading 2.b

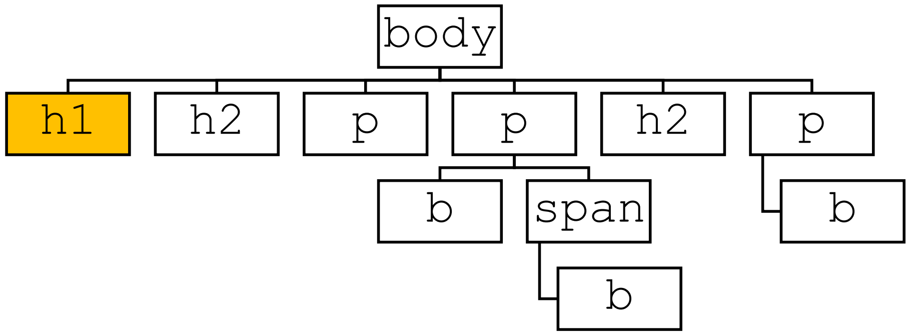
Third paragraph has a **bold** as well

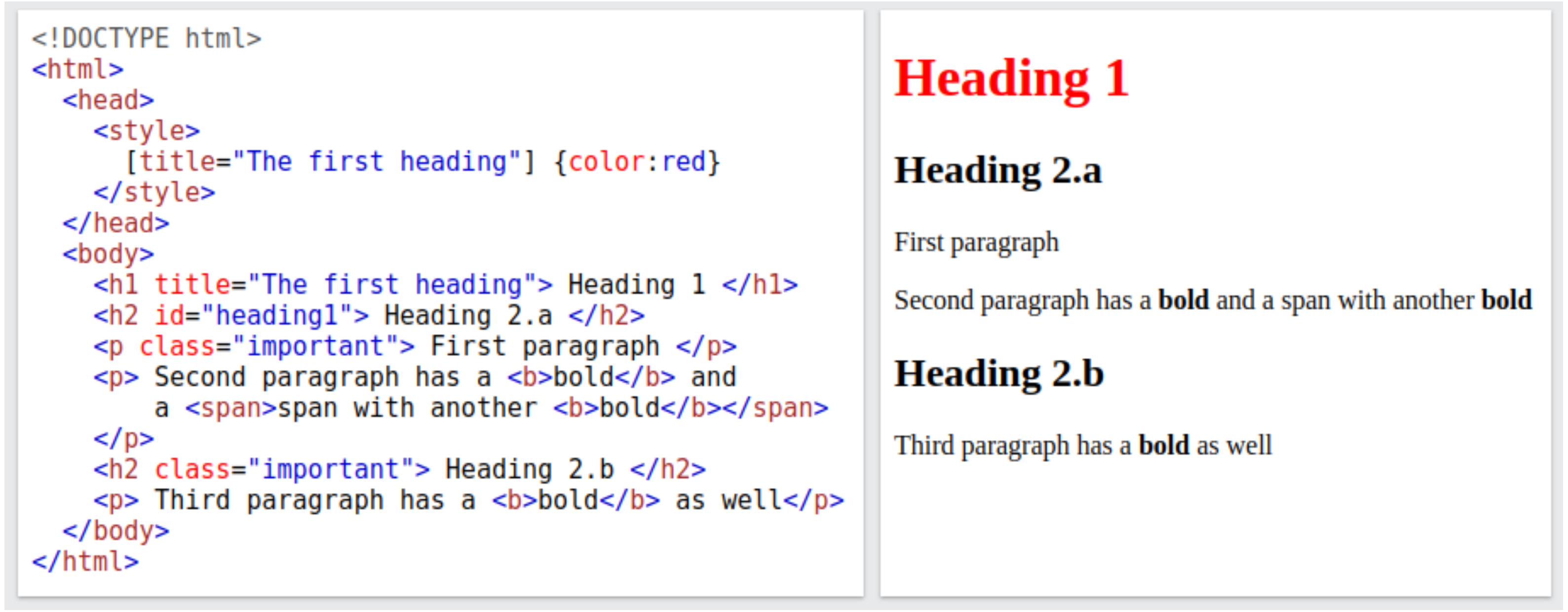
Attribute selector: [*attribute=value*]

- An attribute selector matches on any element that has a given attribute with the specified value.

- The value may be omitted to select elements with that attribute with any value.

- For example, matching on the attribute `[title="The first heading"]` results in:





```
<!DOCTYPE html>
<html>
  <head>
    <style>
      [title="The first heading"] {color:red}
    </style>
  </head>
  <body>
    <h1 title="The first heading"> Heading 1 </h1>
    <h2 id="heading1"> Heading 2.a </h2>
    <p class="important"> First paragraph </p>
    <p> Second paragraph has a <b>bold</b> and
        a <span>span with another <b>bold</b></span>
    </p>
    <h2 class="important"> Heading 2.b </h2>
    <p> Third paragraph has a <b>bold</b> as well</p>
  </body>
</html>
```

Heading 1

Heading 2.a

First paragraph

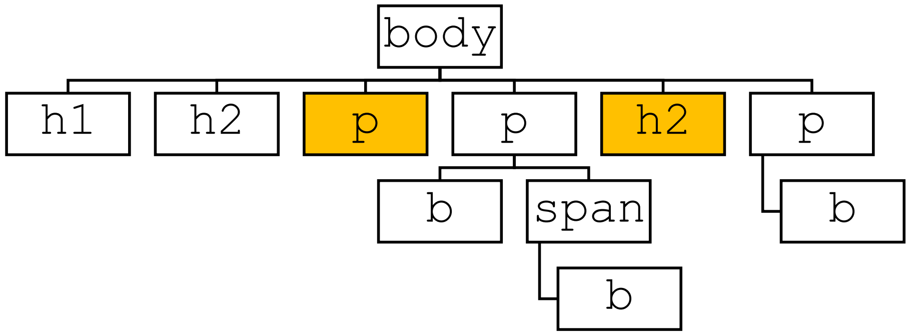
Second paragraph has a **bold** and a span with another **bold**

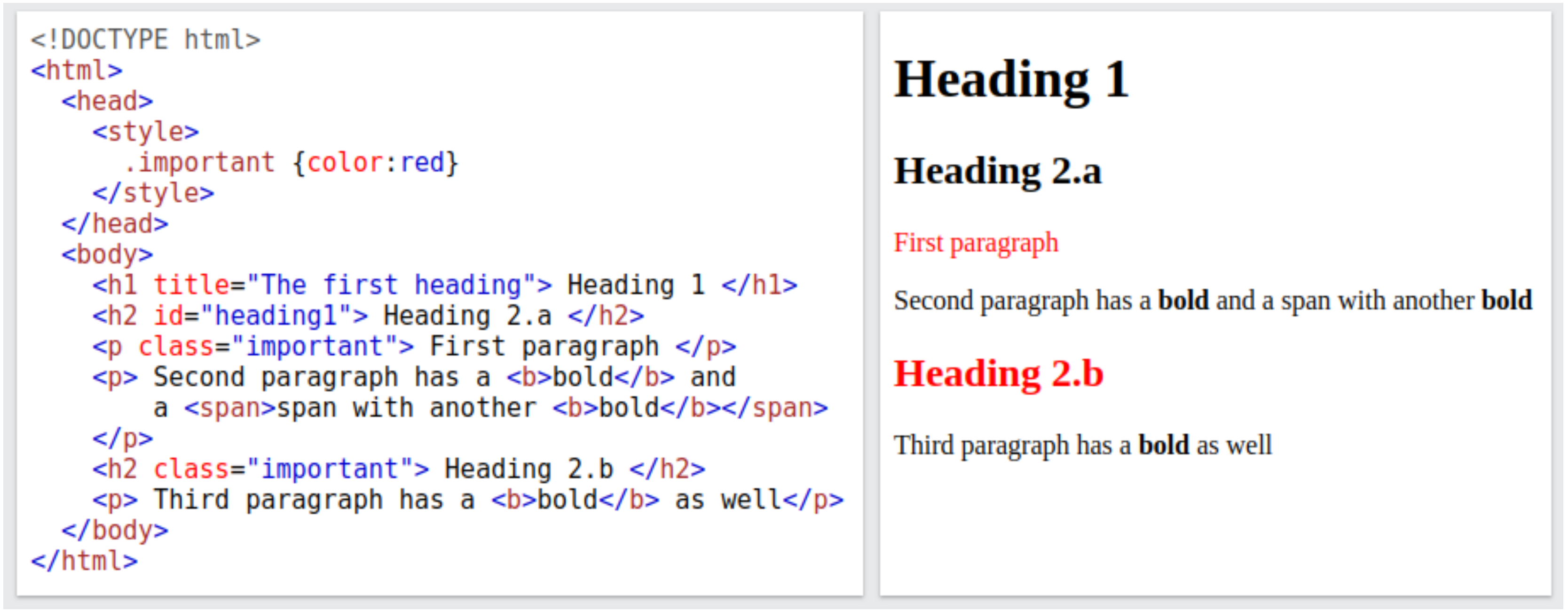
Heading 2.b

Third paragraph has a **bold** as well

Class selector: *.class*

- Class selectors allow the grouping of a set of elements (that may not even use the same tag).
- The class is set using the special `class` attribute.
- For example, matching on the class `.important` results in:





```
<!DOCTYPE html>
<html>
  <head>
    <style>
      .important {color:red}
    </style>
  </head>
  <body>
    <h1 title="The first heading"> Heading 1 </h1>
    <h2 id="heading1"> Heading 2.a </h2>
    <p class="important"> First paragraph </p>
    <p> Second paragraph has a <b>bold</b> and
       a <span>span with another <b>bold</b></span>
    </p>
    <h2 class="important"> Heading 2.b </h2>
    <p> Third paragraph has a <b>bold</b> as well</p>
  </body>
</html>
```

Heading 1

Heading 2.a

First paragraph

Second paragraph has a **bold** and a span with another **bold**

Heading 2.b

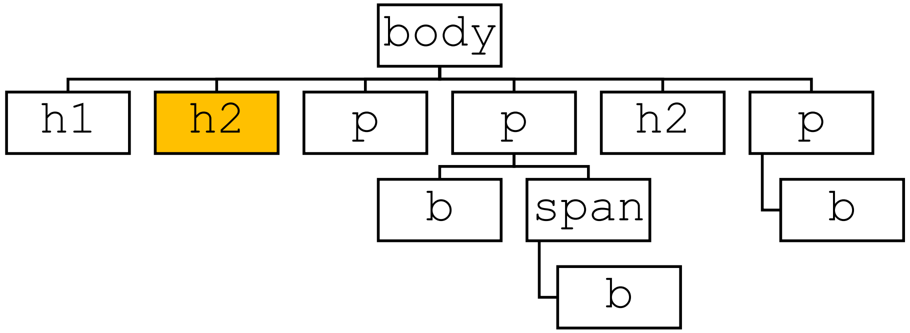
Third paragraph has a **bold** as well

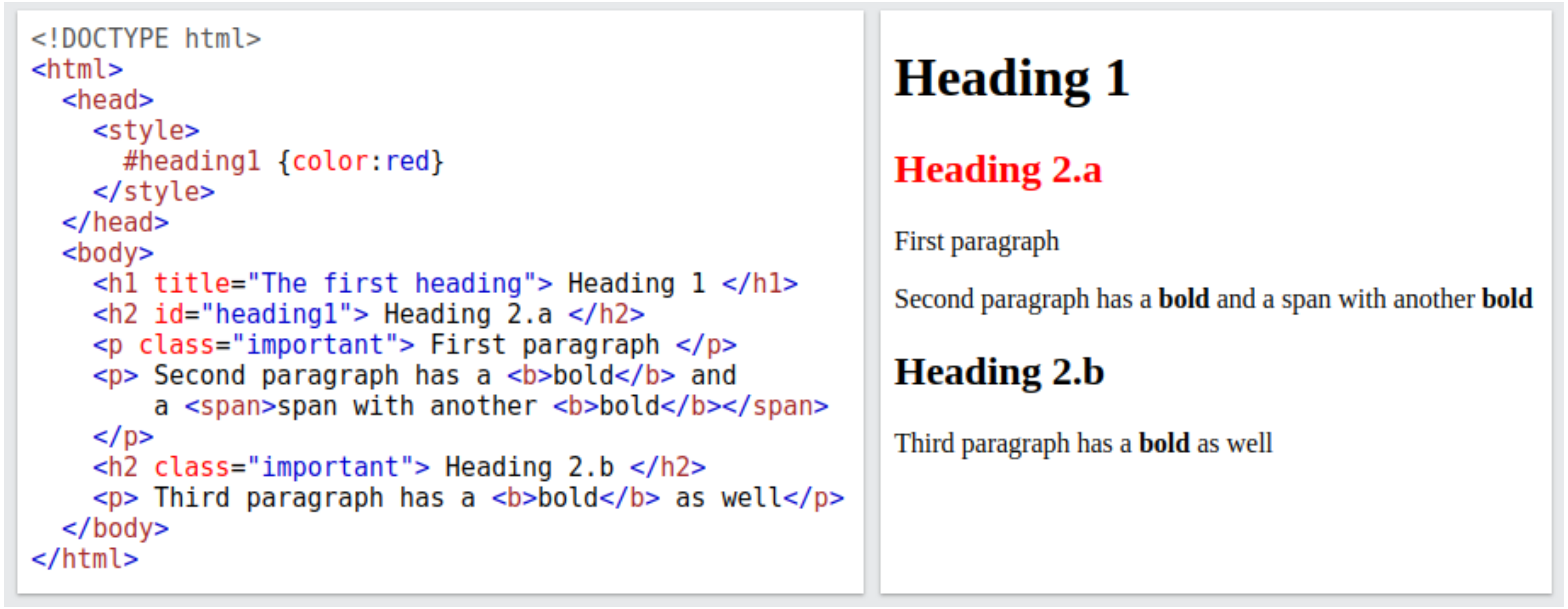
Id selector: *#id*

- An id selector selects an element with the special `id` attribute.

- Unlike class where each value can occur on many tags, each id value must only occur on a single tag the document, e.g. so they can be referenced in the URL and used for navigation.

- Matching on #`heading1` results in:





```
<!DOCTYPE html>
<html>
  <head>
    <style>
      #heading1 {color:red}
    </style>
  </head>
  <body>
    <h1 title="The first heading"> Heading 1 </h1>
    <h2 id="heading1"> Heading 2.a </h2>
    <p class="important"> First paragraph </p>
    <p> Second paragraph has a <b>bold</b> and
        a <span>span with another <b>bold</b></span>
    </p>
    <h2 class="important"> Heading 2.b </h2>
    <p> Third paragraph has a <b>bold</b> as well</p>
  </body>
</html>
```

Heading 1

Heading 2.a

First paragraph

Second paragraph has a **bold** and a span with another **bold**

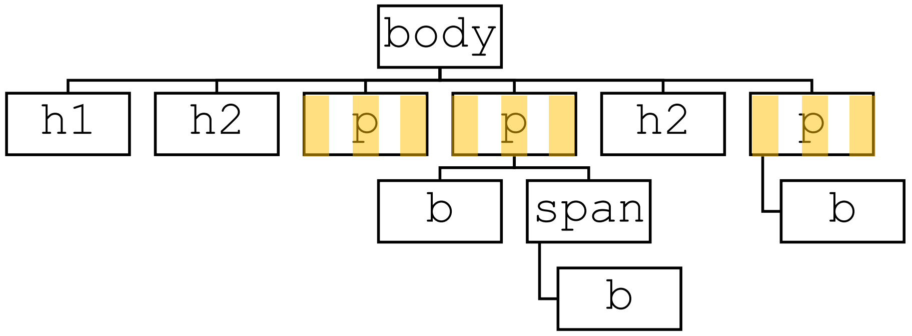
Heading 2.b

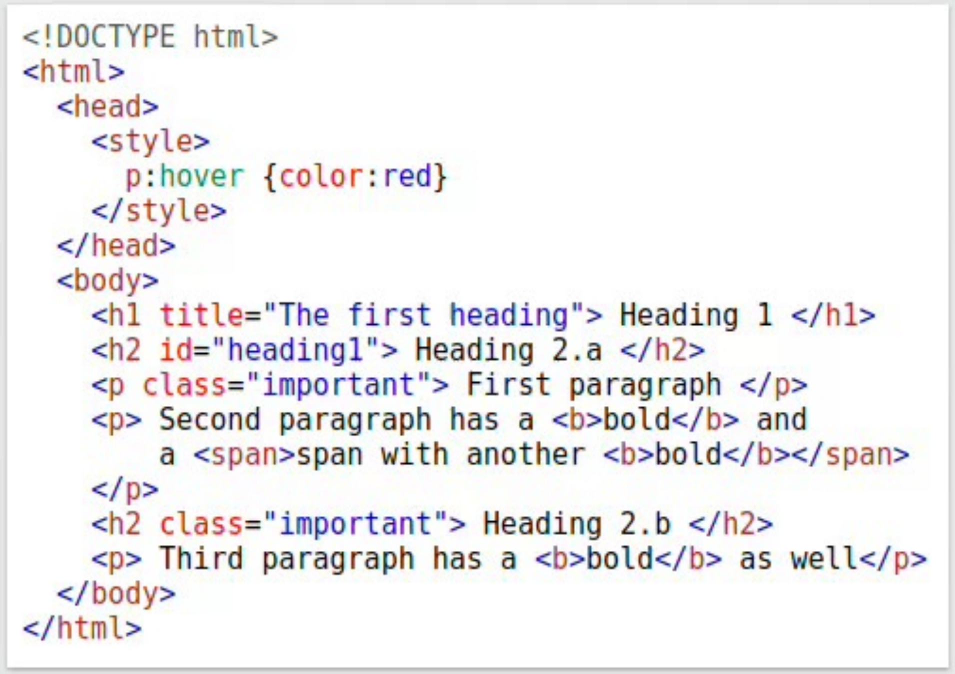
Third paragraph has a **bold** as well

Pseudo-class selector: *:state*

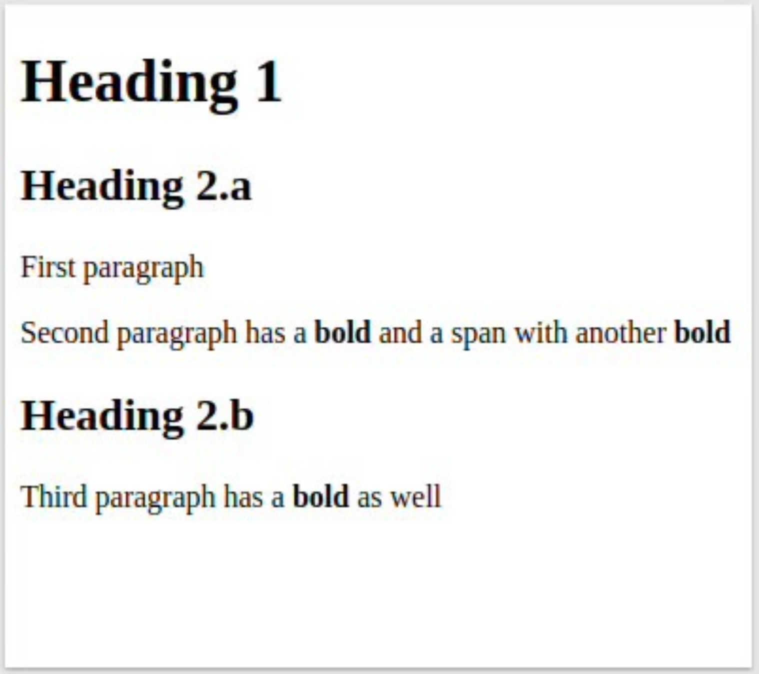
- Pseudo-class selectors allow selecting elements based on their *state* (e.g. hover, focus, valid, visited)

- For example, matching on the class `p:hover` results in the paragraph elements being highlighted when you hover over them:





```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p:hover {color:red}
    </style>
  </head>
  <body>
    <h1 title="The first heading"> Heading 1 </h1>
    <h2 id="heading1"> Heading 2.a </h2>
    <p class="important"> First paragraph </p>
    <p> Second paragraph has a <b>bold</b> and
        a <span>span with another <b>bold</b></span>
    </p>
    <h2 class="important"> Heading 2.b </h2>
    <p> Third paragraph has a <b>bold</b> as well</p>
  </body>
</html>
```



Heading 1

Heading 2.a

First paragraph

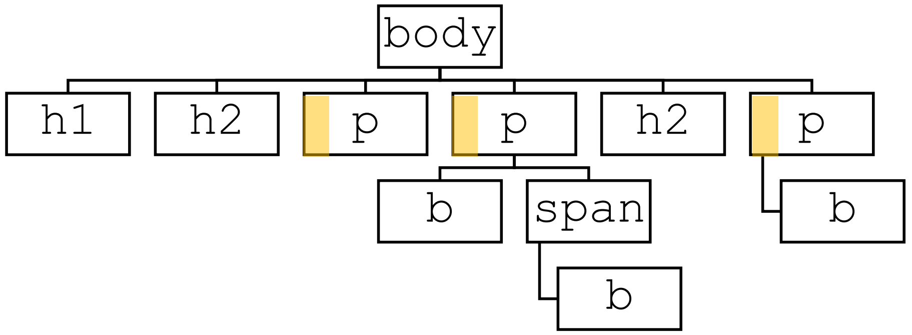
Second paragraph has a **bold** and a span with another **bold**

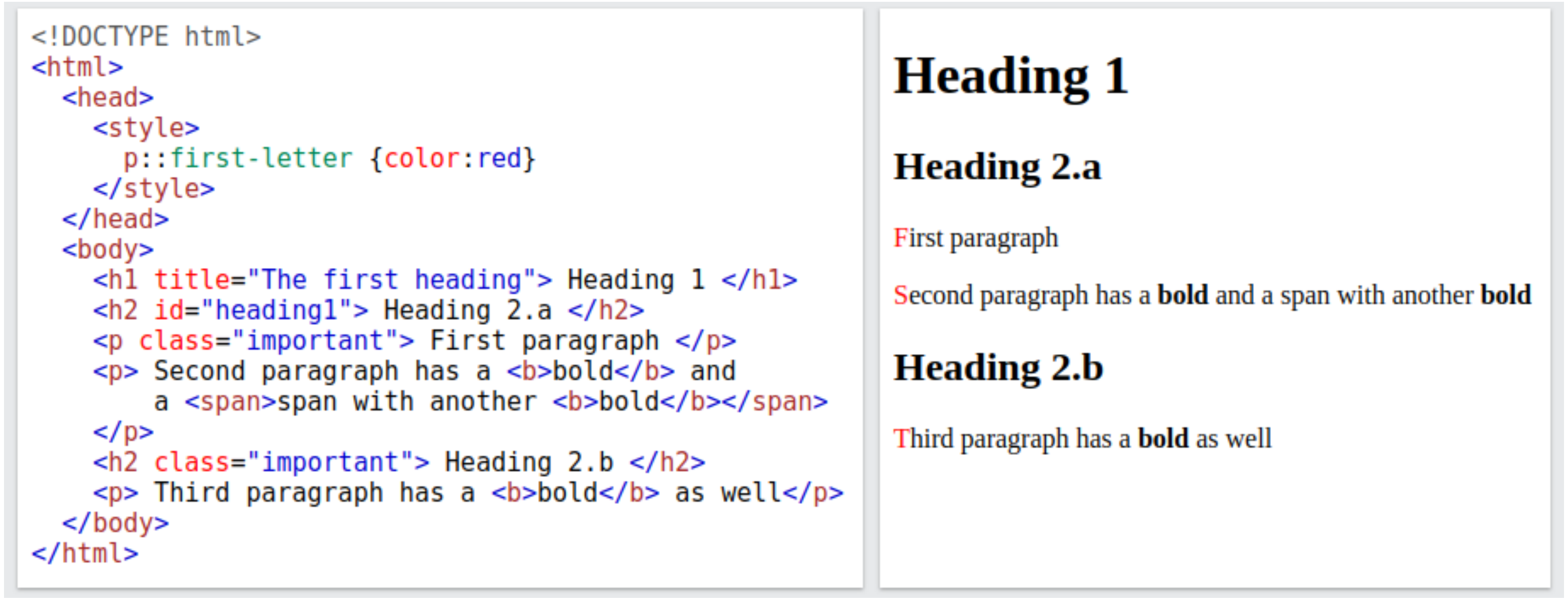
Heading 2.b

Third paragraph has a **bold** as well

Pseudo-element selector: *::pseudo-element*

- Pseudo-element selectors allow you style a specific part of an element (e.g. first-line, first-letter, backdrop)
- For example, matching on the class `p:first-letter` results in the first letter of each paragraph elements being highlighted.





```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p::first-letter {color:red}
    </style>
  </head>
  <body>
    <h1 title="The first heading"> Heading 1 </h1>
    <h2 id="heading1"> Heading 2.a </h2>
    <p class="important"> First paragraph </p>
    <p> Second paragraph has a <b>bold</b> and
        a <span>span with another <b>bold</b></span>
    </p>
    <h2 class="important"> Heading 2.b </h2>
    <p> Third paragraph has a <b>bold</b> as well</p>
  </body>
</html>
```

Heading 1

Heading 2.a

First paragraph

Second paragraph has a **bold** and a span with another **bold**

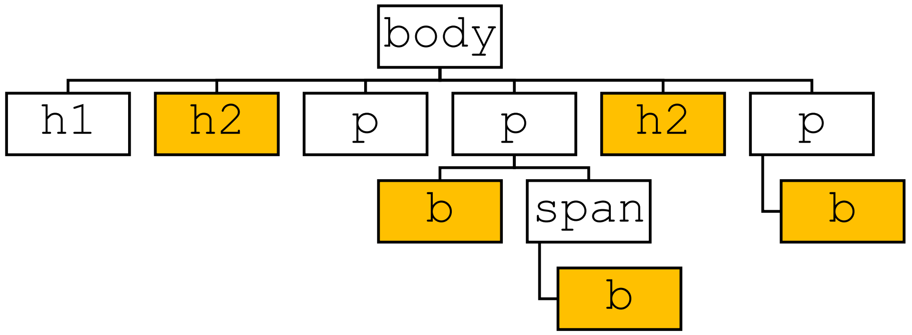
Heading 2.b

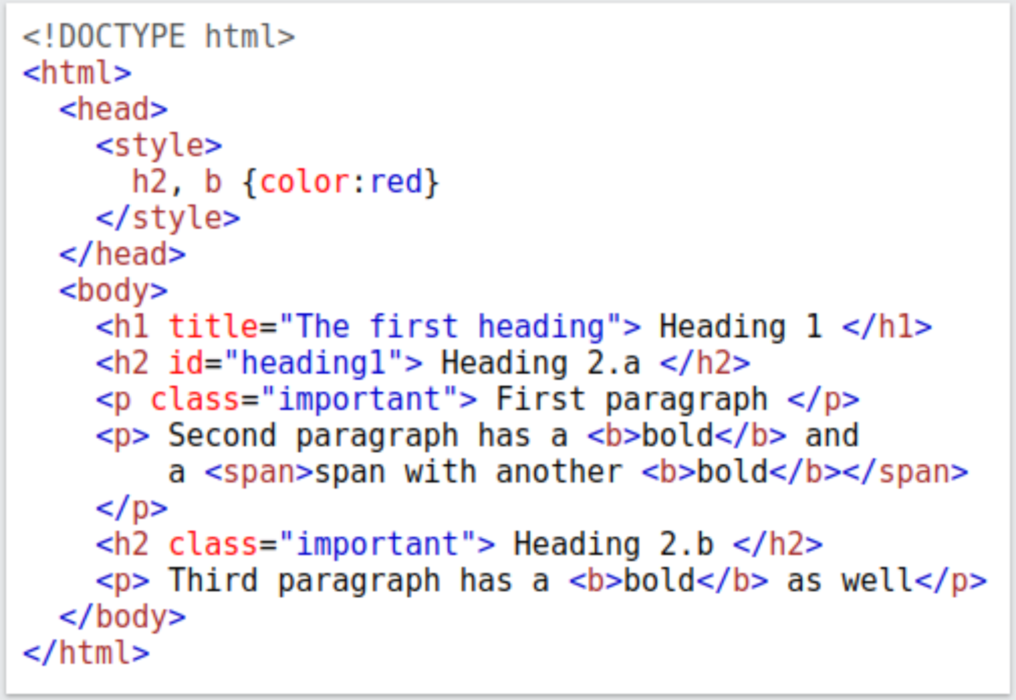
Third paragraph has a **bold** as well

Group selector: s1, s2

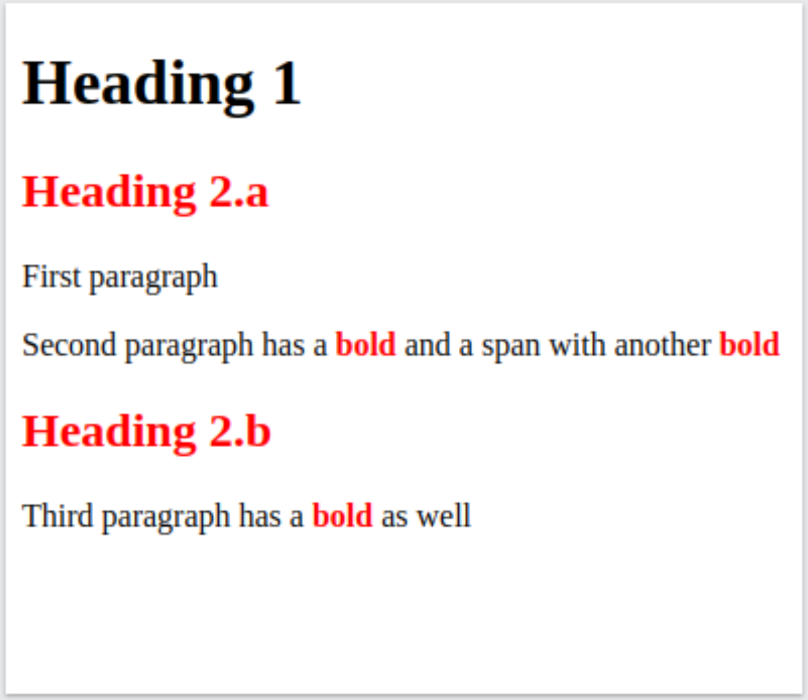
- Group selector `s1,s2` act as a logical *or*, applying the style to elements that match either `s1` or `s2`.

- For example, matching on `h2,b` results in both level 2 headers and bold text being selected:





```
<!DOCTYPE html>
<html>
  <head>
    <style>
      h2, b {color:red}
    </style>
  </head>
  <body>
    <h1 title="The first heading"> Heading 1 </h1>
    <h2 id="heading1"> Heading 2.a </h2>
    <p class="important"> First paragraph </p>
    <p> Second paragraph has a <b>bold</b> and
        a <span>span with another <b>bold</b></span>
    </p>
    <h2 class="important"> Heading 2.b </h2>
    <p> Third paragraph has a <b>bold</b> as well</p>
  </body>
</html>
```



Heading 1

Heading 2.a

First paragraph

Second paragraph has a **bold** and a span with another **bold**

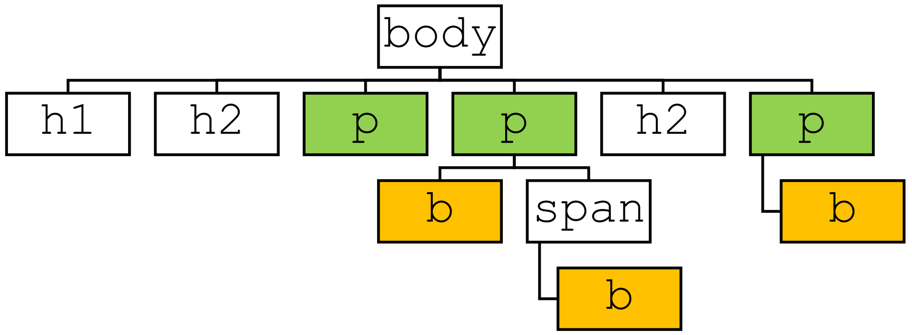
Heading 2.b

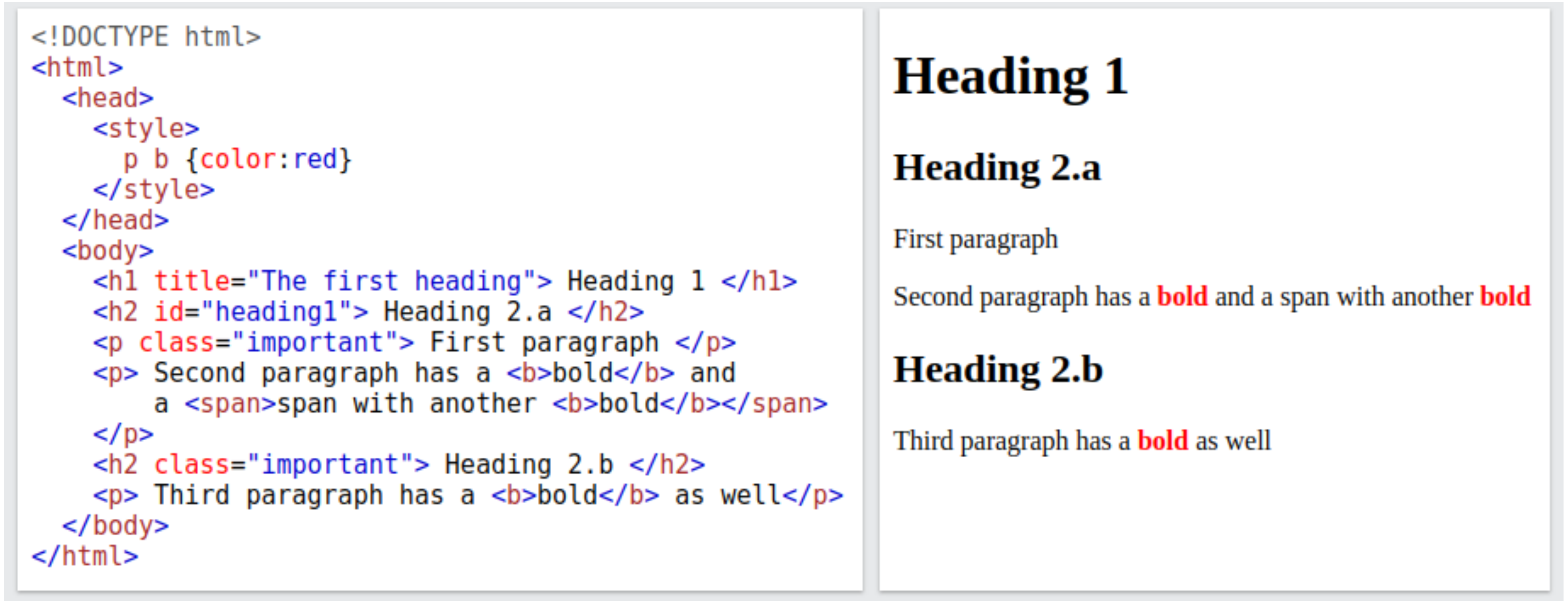
Third paragraph has a **bold** as well

Descendent selector: s1 s2

- The descendant selector `s1 s2` selects anything that matches s2 that is below something that matches s1 in the tree.

- For example, matching on `p b` results in every bold element below a paragraph element to be selected.





```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p b {color:red}
    </style>
  </head>
  <body>
    <h1 title="The first heading"> Heading 1 </h1>
    <h2 id="heading1"> Heading 2.a </h2>
    <p class="important"> First paragraph </p>
    <p> Second paragraph has a <b>bold</b> and
        a <span>span with another <b>bold</b></span>
    </p>
    <h2 class="important"> Heading 2.b </h2>
    <p> Third paragraph has a <b>bold</b> as well</p>
  </body>
</html>
```

Heading 1

Heading 2.a

First paragraph

Second paragraph has a **bold** and a span with another **bold**

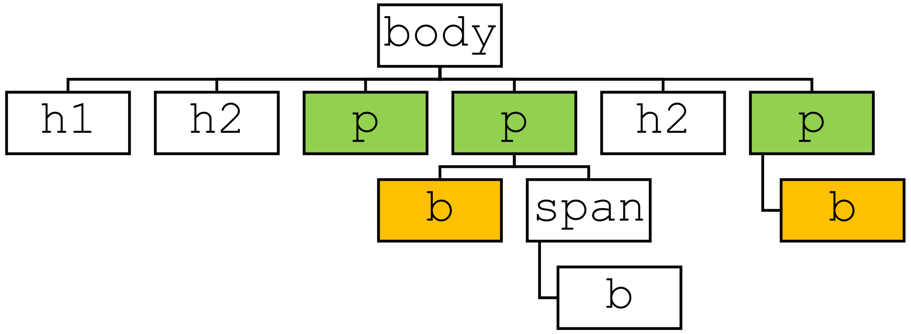
Heading 2.b

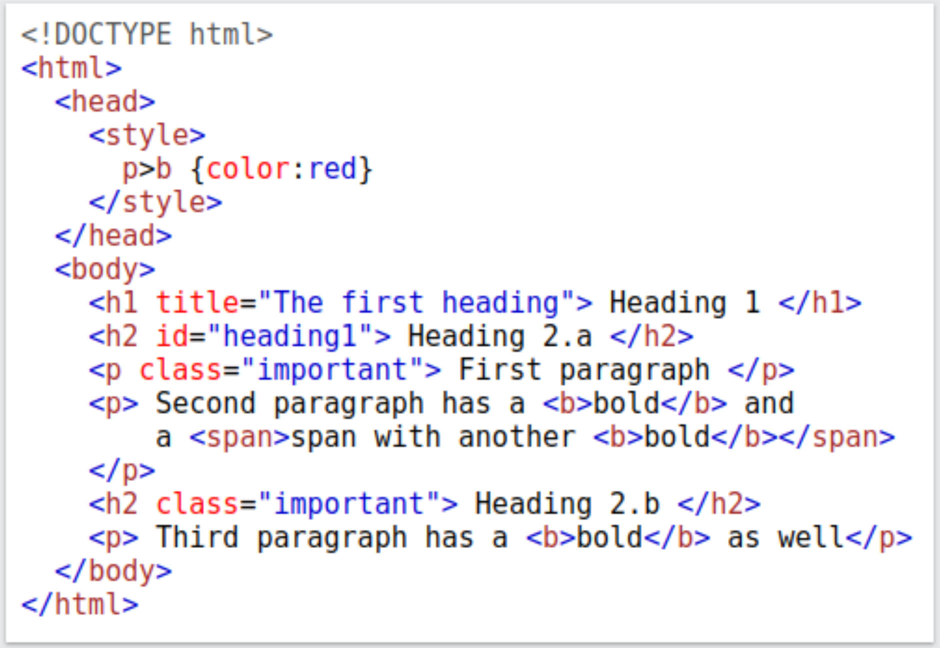
Third paragraph has a **bold** as well

Direct descendent selector: s1>s2

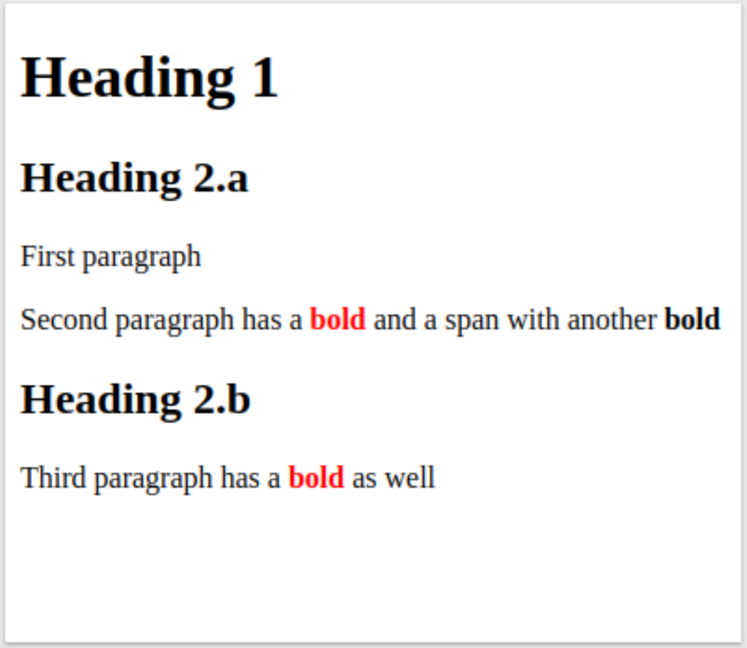
- The direct descendant selector `s1>s2` selects anything that matches s2 that is *directly* below something that matches s1 in the tree.

- For example, matching on `p>b` results in every bold element directly below a paragraph element to be selected.





```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p>b {color:red}
    </style>
  </head>
  <body>
    <h1 title="The first heading"> Heading 1 </h1>
    <h2 id="heading1"> Heading 2.a </h2>
    <p class="important"> First paragraph </p>
    <p> Second paragraph has a <b>bold</b> and
        a <span>span with another <b>bold</b></span>
    </p>
    <h2 class="important"> Heading 2.b </h2>
    <p> Third paragraph has a <b>bold</b> as well</p>
  </body>
</html>
```



Heading 1

Heading 2.a

First paragraph

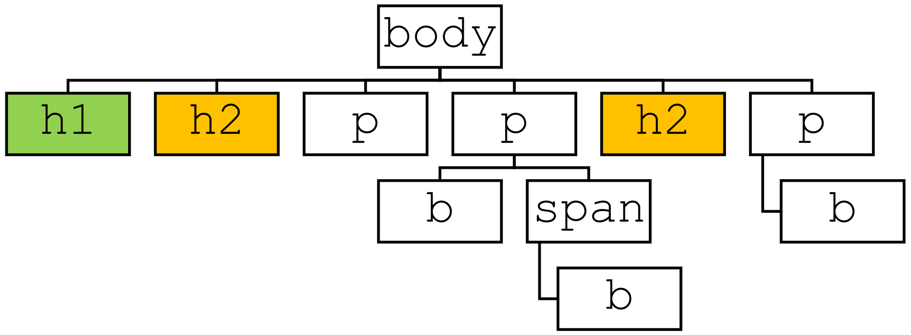
Second paragraph has a **bold** and a span with another **bold**

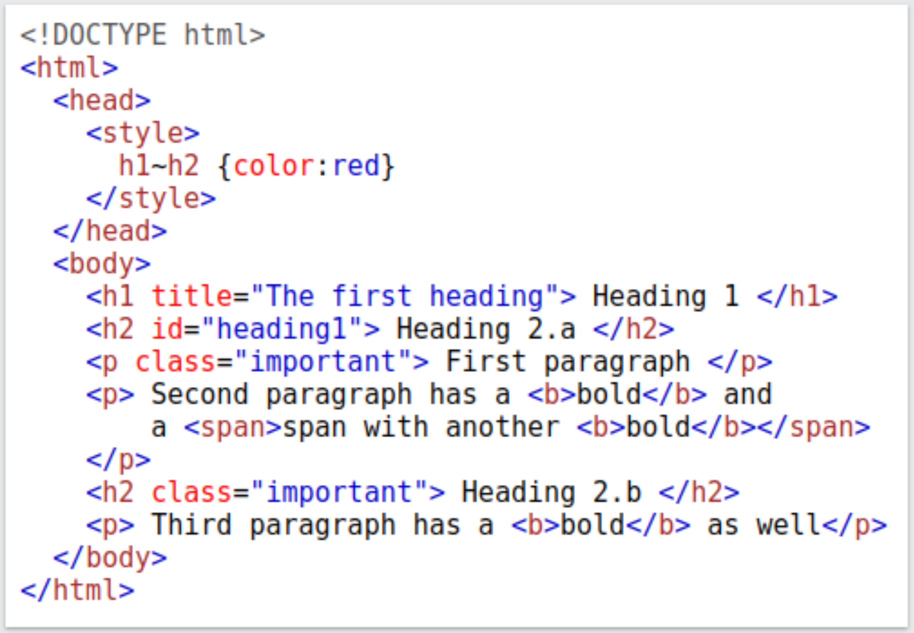
Heading 2.b

Third paragraph has a **bold** as well

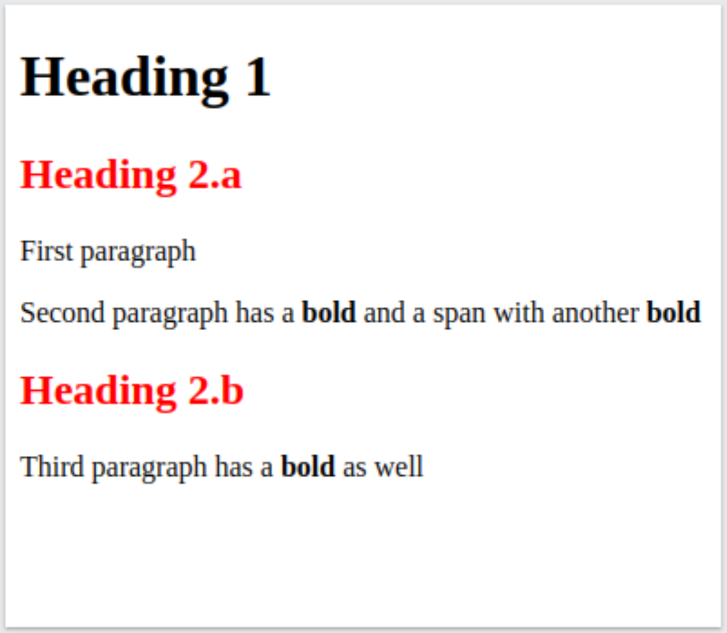
Sibling selector: s1~s2

- The sibling selector `s1~s2` selects anything that matches s2 that shares a parent with something that matches s1 in the tree.
- For example, matching on `h1~h2` results in every h2 element to the right of a h1 element being selected.





```
<!DOCTYPE html>
<html>
  <head>
    <style>
      h1~h2 {color:red}
    </style>
  </head>
  <body>
    <h1 title="The first heading"> Heading 1 </h1>
    <h2 id="heading1"> Heading 2.a </h2>
    <p class="important"> First paragraph </p>
    <p> Second paragraph has a <b>bold</b> and
        a <span>span with another <b>bold</b></span>
    </p>
    <h2 class="important"> Heading 2.b </h2>
    <p> Third paragraph has a <b>bold</b> as well</p>
  </body>
</html>
```



Heading 1

Heading 2.a

First paragraph

Second paragraph has a **bold** and a span with another **bold**

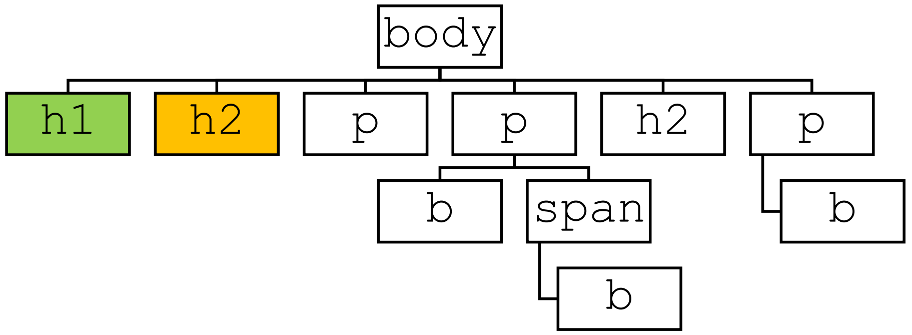
Heading 2.b

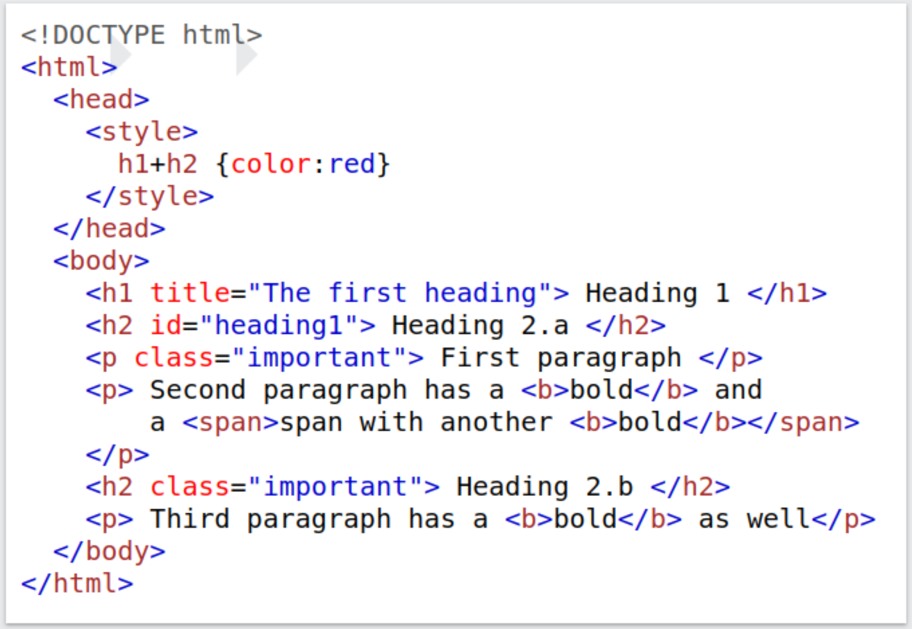
Third paragraph has a **bold** as well

Direct sibling selector: s1+s2

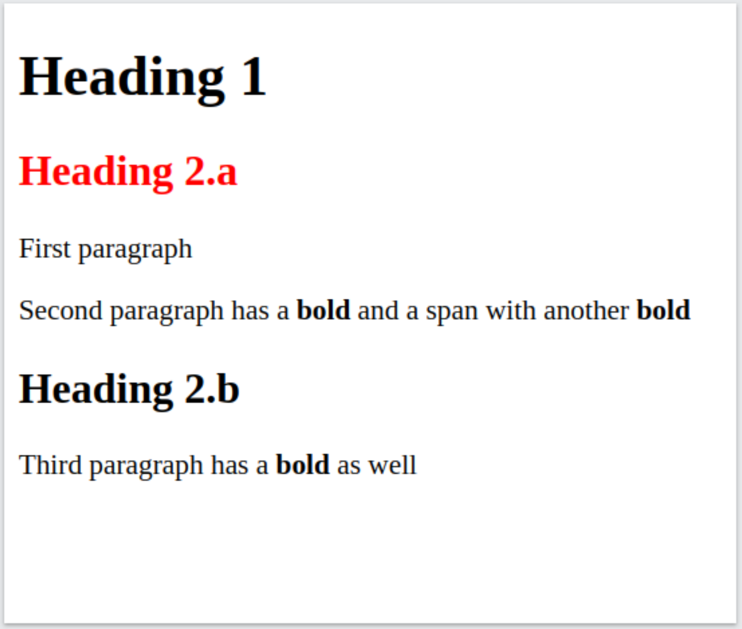
- The direct sibling selector `s1+s2` selects anything that matches s2 that is the next child along of something that matches s1 in the tree.

- For example, matching on `h1+h2` results in every h2 element directly to the right of a h1 element being selected.





```
<!DOCTYPE html>
<html>
  <head>
    <style>
      h1+h2 {color:red}
    </style>
  </head>
  <body>
    <h1 title="The first heading"> Heading 1 </h1>
    <h2 id="heading1"> Heading 2.a </h2>
    <p class="important"> First paragraph </p>
    <p> Second paragraph has a <b>bold</b> and
        a <span>span with another <b>bold</b></span>
    </p>
    <h2 class="important"> Heading 2.b </h2>
    <p> Third paragraph has a <b>bold</b> as well</p>
  </body>
</html>
```



Heading 1

Heading 2.a

First paragraph

Second paragraph has a **bold** and a span with another **bold**

Heading 2.b

Third paragraph has a **bold** as well

Conflict resolution

- An element may be the subject of more than one rule because:
    1. A tag may be used twice as a selector.
    2. A tag may inherit a property *and* be used as a selector.

- This is often unavoidable as your page will invariably contain multiple style sheets with conflicting definitions:
    - User style sheets, (style sheets written by the user via the browser settings)
    - Author style sheets (style sheets used by the developer)
    - User agent style sheets (default style sheets provided by the browser)

- CSS priority can be overridden by the !important modifier.

(Simplified) precedence rules

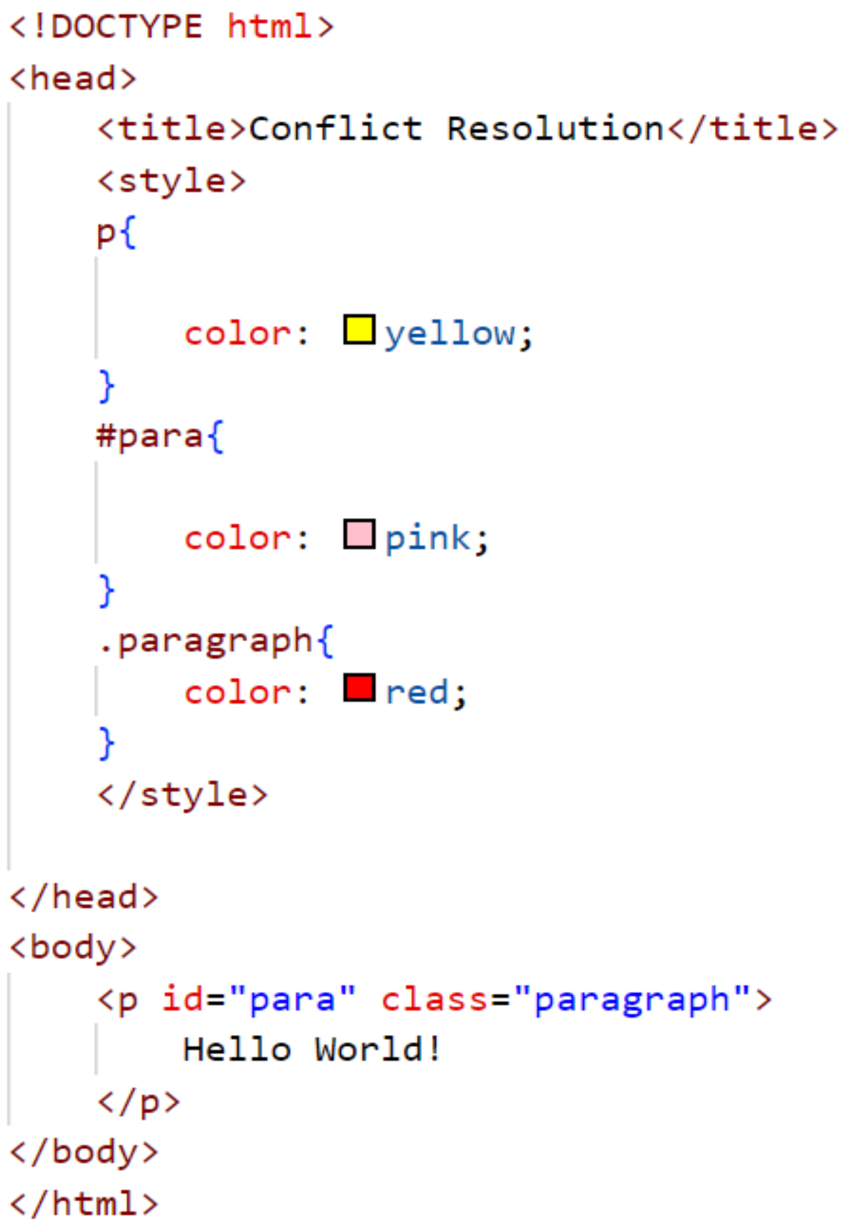
1. First break ties by origin and importance:
    - Transition declarations (used for animation, not covered)
    - Important user agent declarations
    - Important user declarations
    - Important author declarations
    - Normal author declarations
    - Normal user declarations
    - Normal user agent declarations

2. If still tied, then judge on specificity:
    - Inline styles > style sheet styles
    - Number of IDs in selector
    - Number of classes, attributes and pseudo-classes in selector
    - Number of type and pseudo-elements in selector

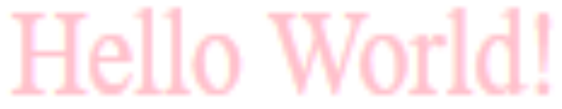
3. If still tied, then choose whichever selector appears last.

Full rules available at: https://www.w3.org/TR/css-cascade-3/

Conflict resolution (IDs vs. Classes vs. Elements)

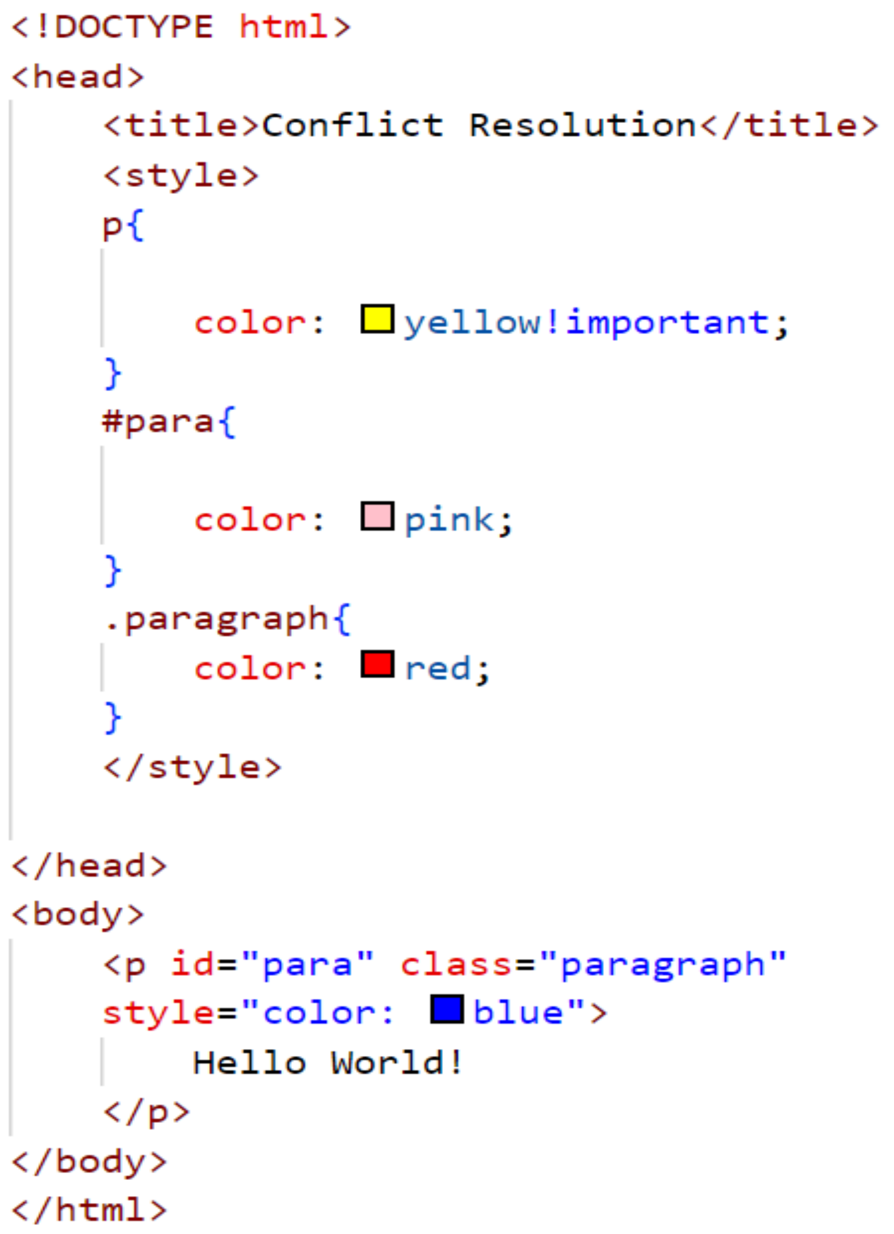


```
<!DOCTYPE html>
<head>
    <title>Conflict Resolution</title>
    <style>
    p{

        color: yellow;
    }
    #para{

        color: pink;
    }
    .paragraph{
        color: red;
    }
    </style>

</head>
<body>
    <p id="para" class="paragraph">
        Hello World!
    </p>
</body>
</html>
```



```
Hello World!
```

Conflict resolution (Inline vs. !Important)



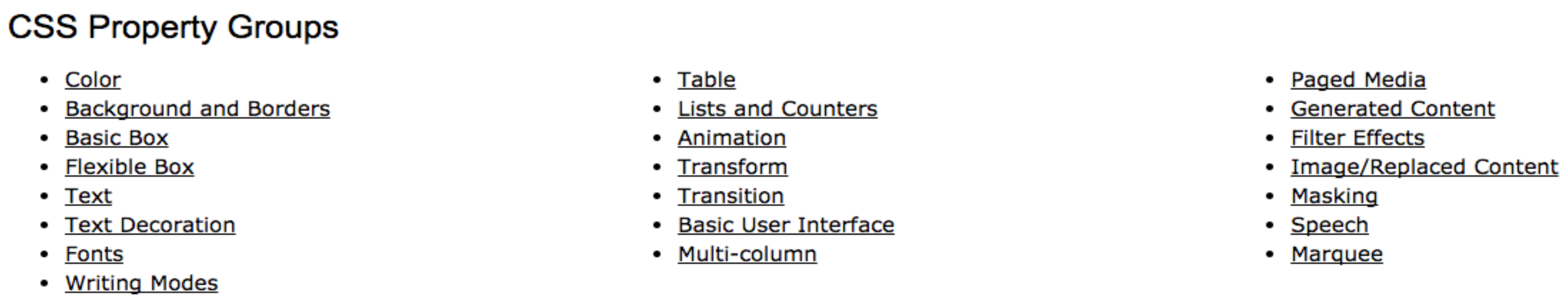
```
<!DOCTYPE html>
<head>
    <title>Conflict Resolution</title>
    <style>
    p{

        color: yellow!important;
    }
    #para{

        color: pink;
    }
    .paragraph{
        color: red;
    }
    </style>

</head>
<body>
    <p id="para" class="paragraph"
    style="color: blue">
        Hello World!
    </p>
</body>
</html>
```

Hello World!

CSS properties

Property groups

- There are many, *many* CSS properties and the list is continually growing.



CSS Property Groups

- Color
- Background and Borders
- Basic Box
- Flexible Box
- Text
- Text Decoration
- Fonts
- Writing Modes
- Table
- Lists and Counters
- Animation
- Transform
- Transition
- Basic User Interface
- Multi-column
- Paged Media
- Generated Content
- Filter Effects
- Image/Replaced Content
- Masking
- Speech
- Marquee

- The basic ones we will cover are:
  - text
  - background
  - borders
  - the box model
  - colors
  - tables
  - lists

Font properties

- `font-size` – values: a number or a name, such as `smaller`, `xx-large`, etc.
- `font-style` – values: `italic`, `normal`
- `font-weight` – degrees of boldness
  - can specify as one of: `bolder`, `lighter`, `bold`, `normal`
  - can specify as a multiple of 100 (100 – 900)
- `font`
  – For specifying a list of font properties at the same time
  – `font: bolder 14pt Arial Helvetica`
  – Order must be style, weight, size, name(s)
- `text-decoration` – values: `line-through`, `overline`, `underline`, `none`
- `letter-spacing` – values: any number

Text alignment

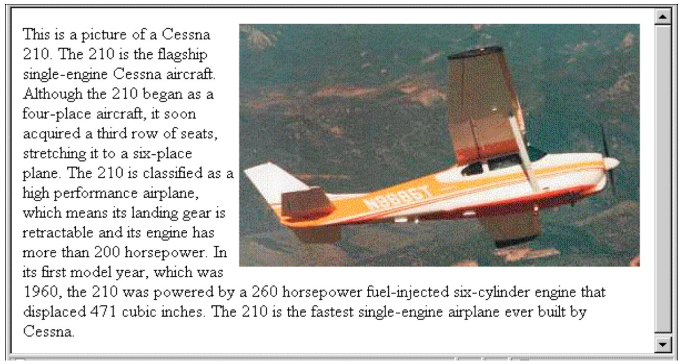
- The `text-indent` property allows indentation:
  - Takes either a length or a % value

- The `text-align` property has 4 possible values:
  - `left` (the default), `center`, `right`, or `justify`

- Sometimes we want text to flow around another element - the `float` property
  - The `float` property has the possible values, `left`, `right`, and `none` (the default)
  - If we have an element we want on the right, with text flowing on its left, we use the default `text-align` value (`left`) for the text and the `right` value for `float` on the element we want on the right

```
<img src="c210.jpg” style="float: right"/>
```

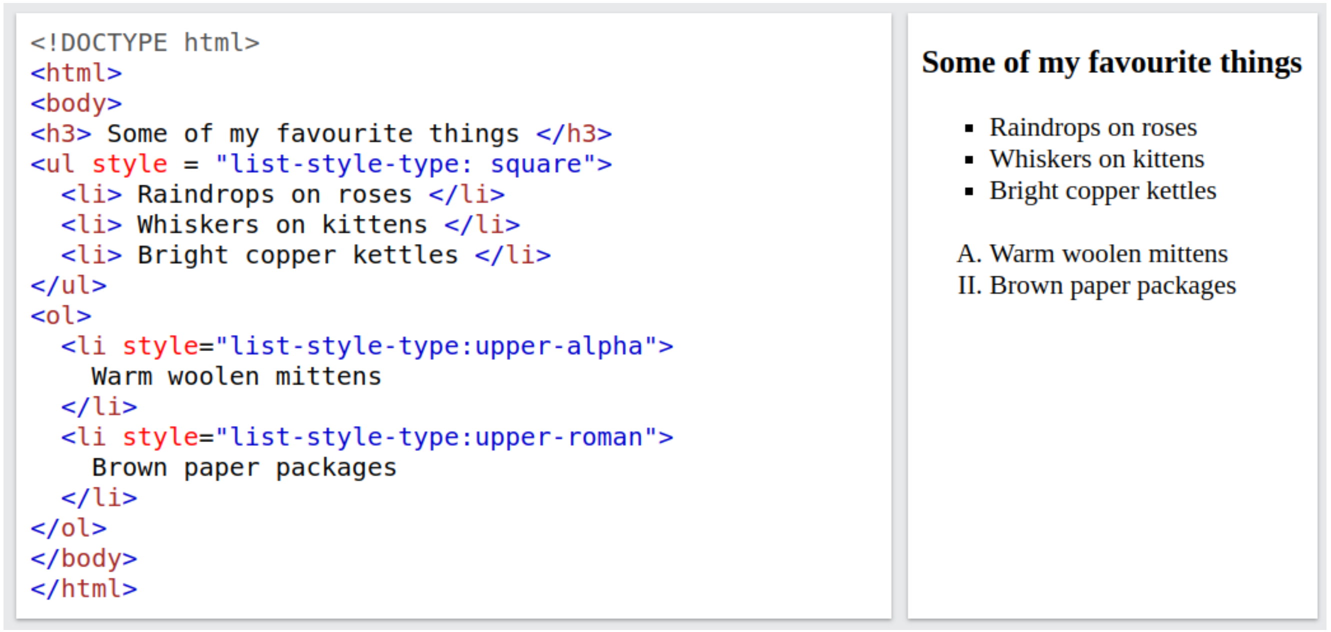
  - Some text with the default alignment - `left`



This is a picture of a Cessna 210. The 210 is the flagship single-engine Cessna aircraft. Although the 210 began as a four-place aircraft, it soon acquired a third row of seats, stretching it to a six-place plane. The 210 is classified as a high performance airplane, which means its landing gear is retractable and its engine has more than 200 horsepower. In its first model year, which was 1960, the 210 was powered by a 260 horsepower fuel-injected six-cylinder engine that displaced 471 cubic inches. The 210 is the fastest single-engine airplane ever built by Cessna.

List properties

- The `list-style-type` property sets the marker type used for `<li>` elements.
- When set on the `<ul>`/`<ol>` element, the style applies to all `<li>` elements in the list.
- When set on the individual `<li>` element it applies only to that element.
- Possible values:
  - Unordered list: `disc`, `square`, `circle`, `none` etc.
  - Ordered list: `decimal`, `upper-alpha`, `upper-roman`, etc.



```
<!DOCTYPE html>
<html>
<body>
<h3> Some of my favourite things </h3>
<ul style = "list-style-type: square">
  <li> Raindrops on roses </li>
  <li> Whiskers on kittens </li>
  <li> Bright copper kettles </li>
</ul>
<ol>
  <li style="list-style-type:upper-alpha">
    Warm woolen mittens
  </li>
  <li style="list-style-type:upper-roman">
    Brown paper packages
  </li>
</ol>
</body>
</html>
```

Some of my favourite things

▪ Raindrops on roses
▪ Whiskers on kittens
▪ Bright copper kettles

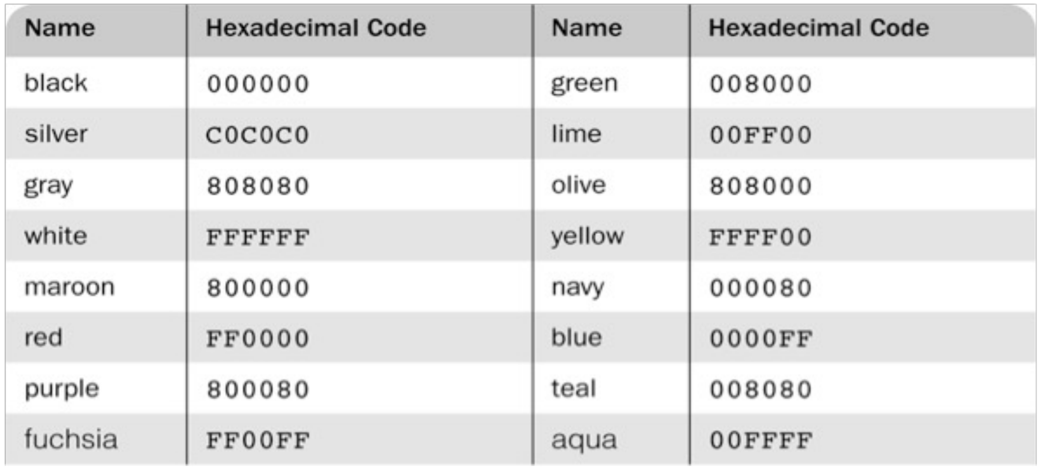
A. Warm woolen mittens
II. Brown paper packages

Colours

- The color property specifies the foreground colour of element

- There are three colour collections:

1. There is a set of 16 colours that are guaranteed to be displayable by all graphical browsers on all colour monitors:



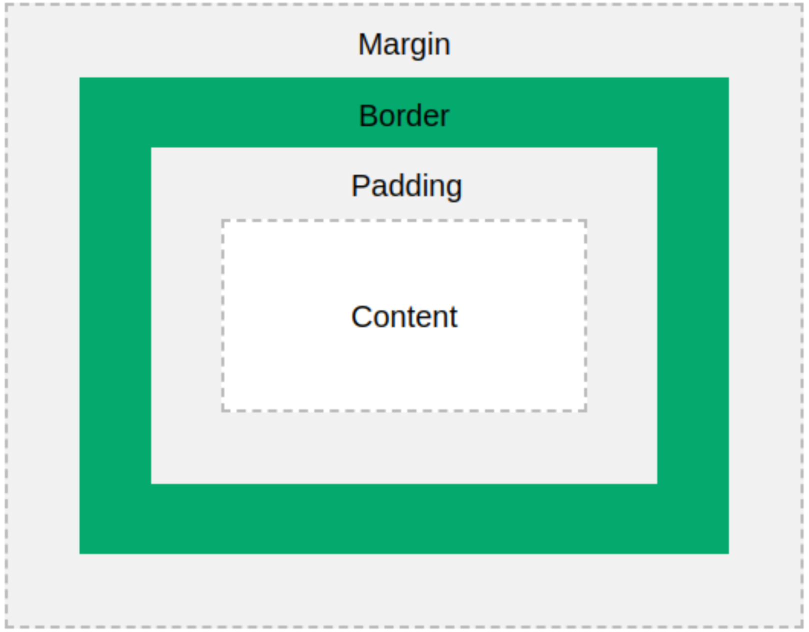
Name	Hexadecimal Code	Name	Hexadecimal Code
black	000000	green	008000
silver	C0C0C0	lime	00FF00
gray	808080	olive	808000
white	FFFFFF	yellow	FFFF00
maroon	800000	navy	000080
red	FF0000	blue	0000FF
purple	800080	teal	008080
fuchsia	FF00FF	aqua	00FFFF

2. There is a much larger set of 140 named colours supported by all major browsers:
   https://www.w3schools.com/cssref/css_colors.php
3. Any one of 16 million different colours
   - #000000, #000001, #000002, . . . , #FFFFFE, #FFFFFF

CSS layout

The Box model

- Every element is essentially laid out as four boxes which is known as the box model.
- It is used to control the spacing and borders around an element on the page.

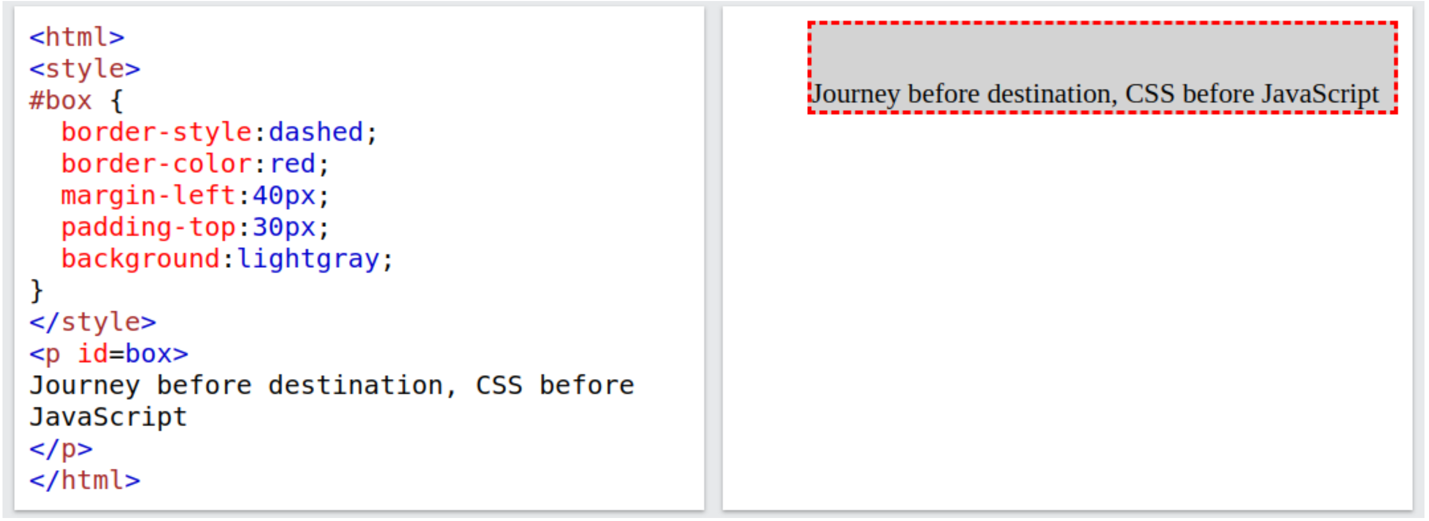


- The content is where the actual element is rendered.

The Box model – margin and border

- The margin of an element is the space between its border and its neighboring element.
  - The margin is always transparent, and its size can be set with margin, margin-left, margin-top, …

- The border of an element can be set using the following properties:
  - border-style: none (default), dotted, dashed, etc.
  - border-width: thin, medium (default), thick, or a length value in pixels
  - border-color: any colour

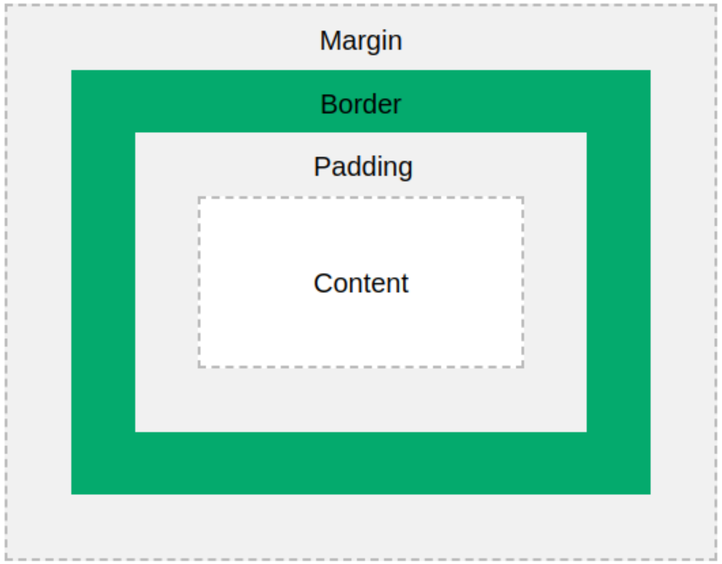


```
<html>
<style>
#box {
  border-style:dashed;
  border-color:red;
  margin-left:40px;
  padding-top:30px;
  background:lightgray;
}
</style>
<p id=box>
Journey before destination, CSS before
JavaScript
</p>
</html>
```

Journey before destination, CSS before JavaScript

The Box model - padding and sizes

- The `padding` of an element is the space between its border and its content.
  - The margin is always transparent, and its size can be set with `padding`, `padding-left`, `padding-top`, …
- When you set the `width` and `height` properties of an element with CSS, you (normally) just set the width and height of the content area.
- To calculate the *total* width and height of an element, you must also include the padding and borders.
- The margin property also affects the total space that the box will take up on the page, but the margin is not included in the actual size of the box. The box's total width and height stops at the border.

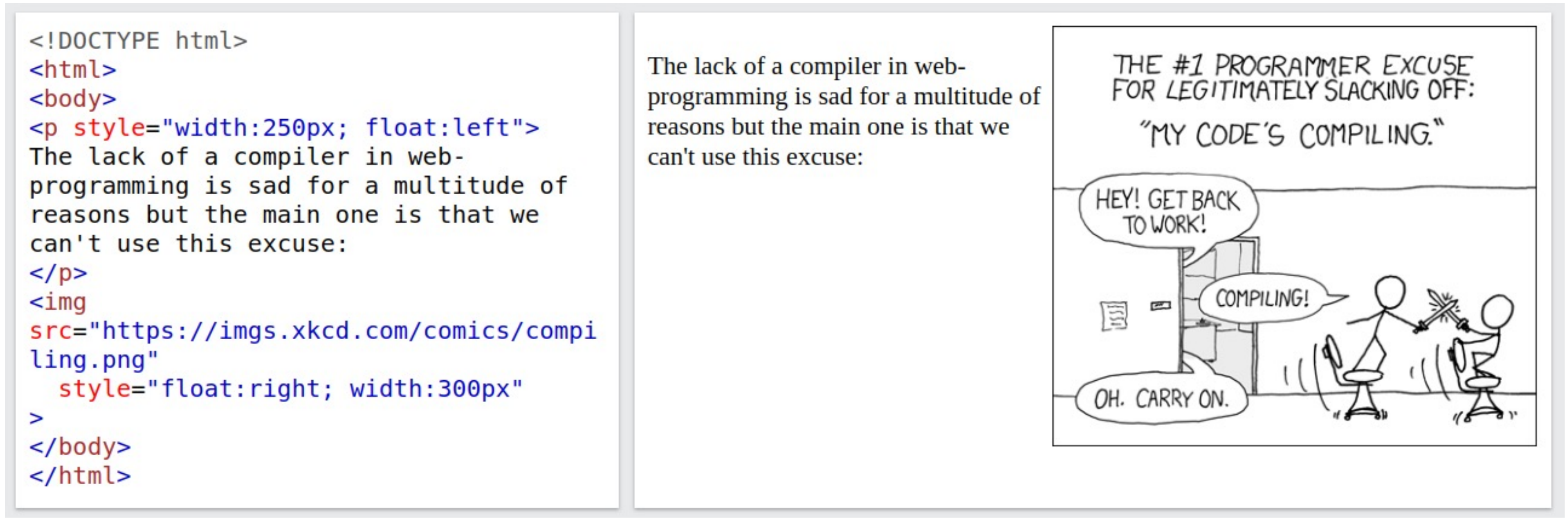


Document flow and the float property

- By default, elements are laid out according to the normal flow of the document – they appear sequentially above each other in the order they are declared.

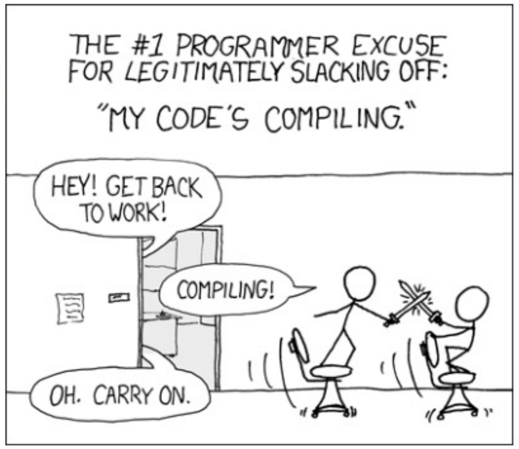
- There are three ways of overriding the normal flow.

1. The `float` property
   - The element is laid out according to the normal flow, then shifted to lie to the left or right of the previous element.
   - There are three values: `none`, `left`, `right`, `inherit`
   - Used to e.g., have images and text on the same line.



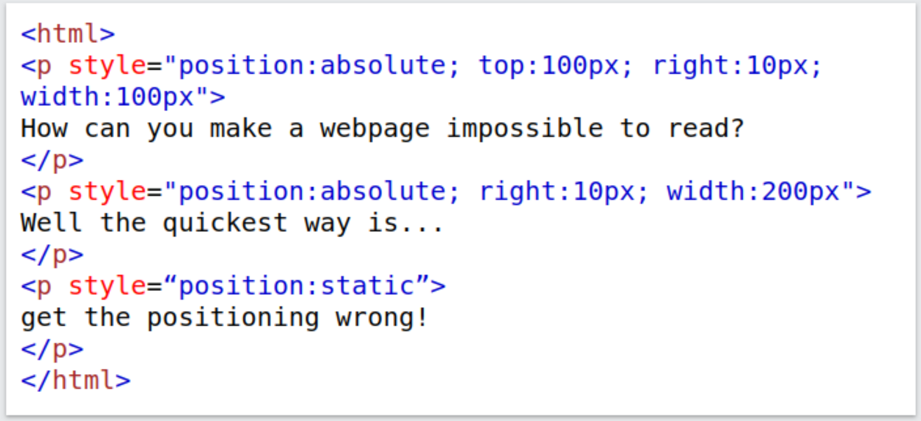
```
<!DOCTYPE html>
<html>
<body>
<p style="width:250px; float:left">
The lack of a compiler in web-
programming is sad for a multitude of
reasons but the main one is that we
can't use this excuse:
</p>
<img
src="https://imgs.xkcd.com/comics/compi
ling.png"
  style="float:right; width:300px"
>
</body>
</html>
```

The lack of a compiler in web-programming is sad for a multitude of reasons but the main one is that we can't use this excuse:

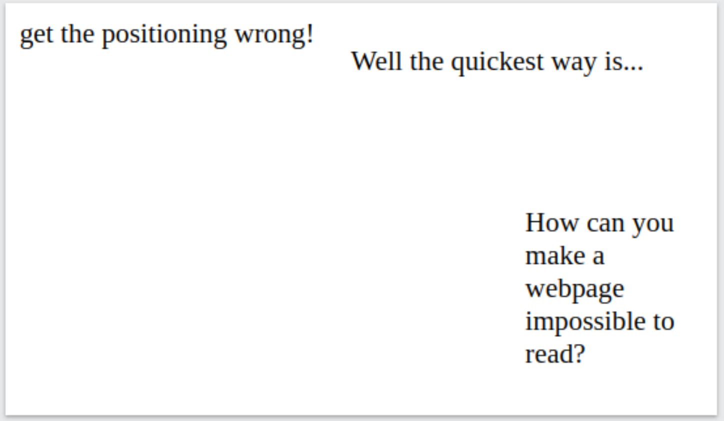


Position property

2. The `position` property
   - Uses the *offset* properties: `top`, `left`, `right`, `bottom`
   - There are five value:
     - `static` - (default) - go with the normal flow
     - `relative` – the element is offset relative to its normal flow position, but the element is not removed from the flow.
     - `absolute` – the element is offset relative to its most recently positioned ancestor. The element is removed entirely from normal flow.
     - `fixed` - the element is offset relative to the fixed viewport. The element is removed entirely from normal flow.
     - `sticky` - switches between relative and fixed depending on the scroll position.



```
<html>
<p style="position:absolute; top:100px; right:10px;
width:100px">
How can you make a webpage impossible to read?
</p>
<p style="position:absolute; right:10px; width:200px">
Well the quickest way is...
</p>
<p style="position:static">
get the positioning wrong!
</p>
</html>
```



get the positioning wrong!
Well the quickest way is...
How can you make a webpage impossible to read?

Position property and frameworks

3. The `display` property determines how the element is treated in normal flow
   - There are ~20 possible values:
     - `block`, `inline`, `flex`, `grid`, `inline-flex`, etc.
     - Won't go into detail here but see various tutorials online.

Laying out your webpage manually with raw CSS is hard!
Furthermore you must consider how the webpage looks across many different screen types and sizes.

- A much more scalable way is to use a CSS framework which are libraries of CSS code allowing you to quickly build a visually appealing and responsive website.

- See next Lecture for more details.

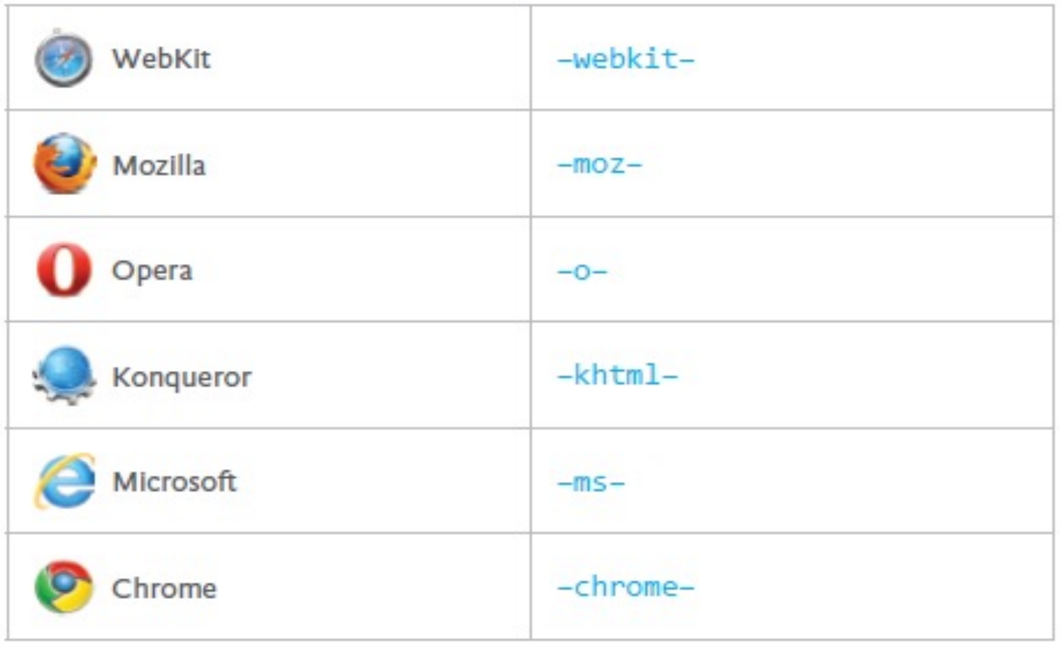
Other

Vendor prefixes

A positive catalyst for the evolution to exciting technologies

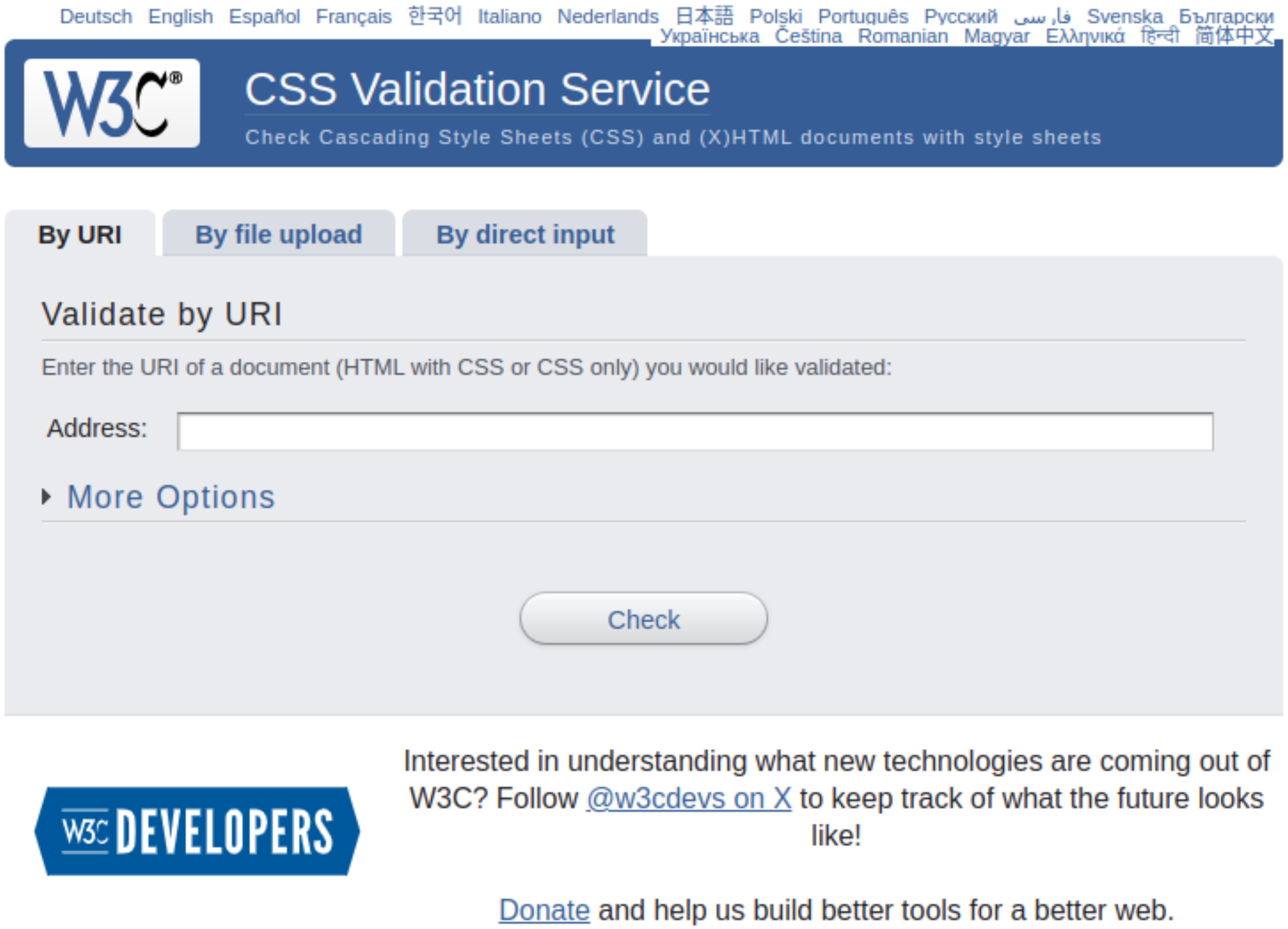
"… force the vendors and the Working Group to work together to devise the tests necessary to determine interoperability. Those tests can then guide those who follow, helping them to achieve interoperable status much faster. They could literally ship the prefixed implementation in one public beta and drop the prefix in the next."

```
.foo {
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  border-radius: 10px;
}
```



WebKit	-webkit-
Mozilla	-moz-
Opera	-o-
Konqueror	-khtml-
Microsoft	-ms-
Chrome	-chrome-

Validating CSS



Deutsch English Español Français 한국어 Italiano Nederlands 日本語 Polski Português Русский فارسی Svenska Български Українська Čeština Romanian Magyar Ελληνικά हिन्दी 简体中文

CSS Validation Service

Check Cascading Style Sheets (CSS) and (X)HTML documents with style sheets

Validate by URI

Enter the URI of a document (HTML with CSS or CSS only) you would like validated:

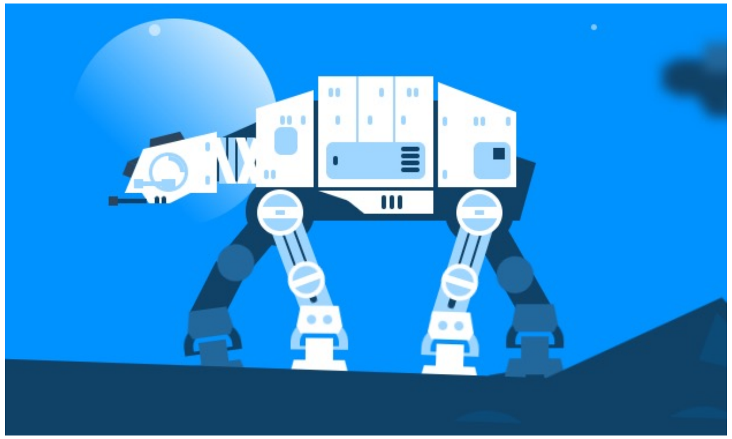
Address:

Interested in understanding what new technologies are coming out of W3C? Follow @w3cdevs on X to keep track of what the future looks like!

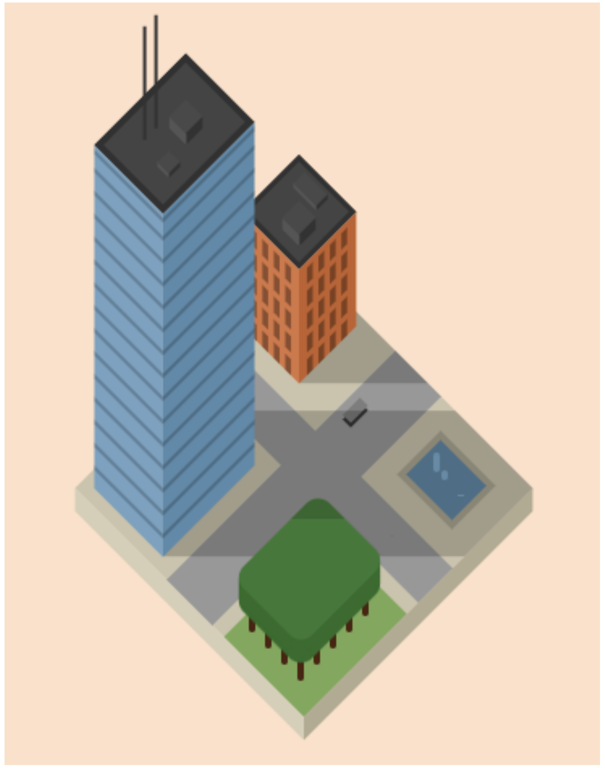
Donate and help us build better tools for a better web.

CSS is boring...

You can build some amazing things with pure CSS:



https://codepen.io/r4ms3s/pen/gajVBG



https://codepen.io/stepan/pen/NWmqdW