Web Fundamentals

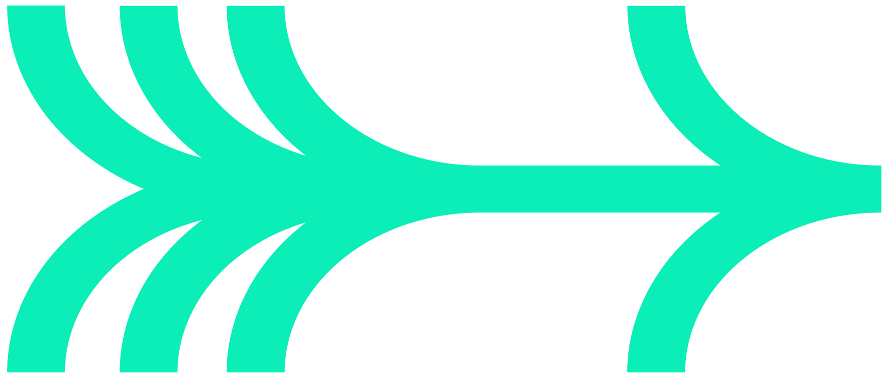
CSS

LEARNING OBJECTIVES

- Understand what CSS is
- Understand how CSS can be applied to web pages
- Understand the syntax of writing CSS rules
- Be able to select elements to apply CSS to
- Be able to work with Text, Colours, and Images
- Be able to work with the Box Model and position elements
- Be able to style lists and tables
- Be able to add CSS animations, transforms, and transitions to elements

Cascading Style Sheets

- CSS application and syntax
- CSS and the DOM
- Inheritance and Selecting Elements
- Text and Colours
- Measurement Units
- Images and Backgrounds
- The Box Model
- Lists and Tables
- Animations, Transitions, and Transformations



CSS Application and Syntax

CSS Fundamentals

Cascading Style Sheets

CSS stands for Cascading Style Sheets

It describes how HTML elements are to be displayed

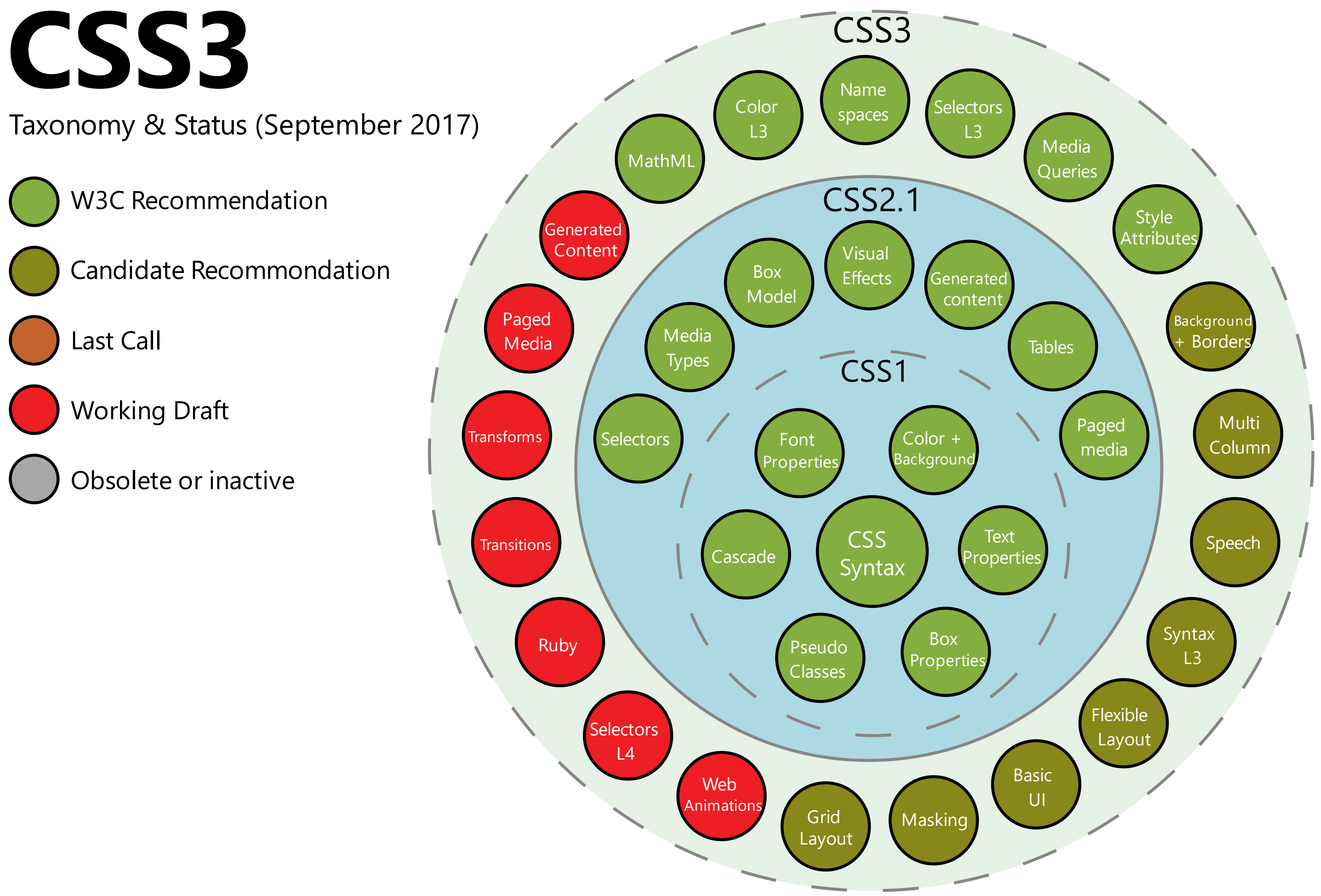
- This could be on the screen, on other media, or even how it should be printed on paper

Can control the layout of multiple web pages from a set of rules

Styling can be applied in one of four ways:

- Inline – defined in the actual element to style
- In an embedded stylesheet on the page – defined on a per-page basis
- In an external style sheet linked to the page – defined inside a separate .css file
- By linking in some existing CSS (almost never to be used)

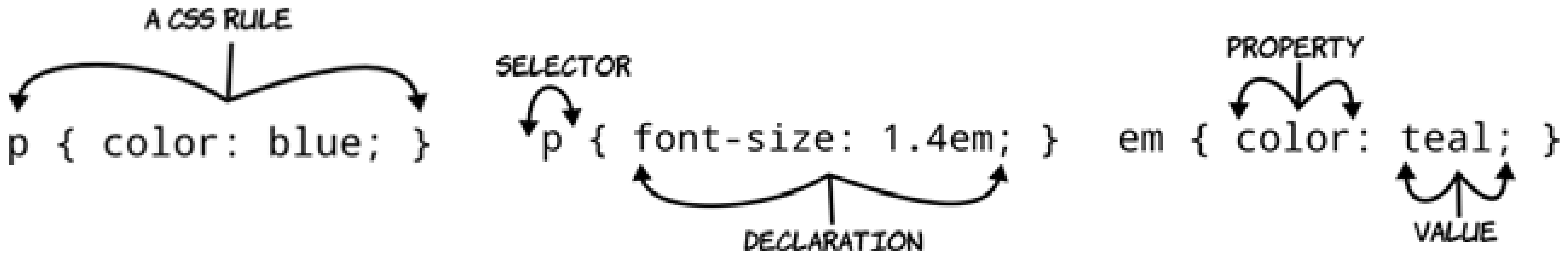
CSS Taxonomy and Status (2017)



- This is a popular diagram showing the history of CSS progression

CSS Basic Syntax

- Rules, selectors, properties, and values
- A CSS style sheet is made up of rules
- Here are three examples CSS rules:



```
p { color: blue; }   p { font-size: 1.4em; }   em { color: teal; }
```

Inline Styles

- `style` attribute can be used on any HTML tag
- Affects that HTML tag only

```
<p style="margin-left: 1in; margin-right: 1in; line-height:200%">
  This text will be shown with one-inch left and right margins, and
  double-spaced.
</p>
<p>
  This text is formatted as normal for &lt;p&gt; tags.
</p>
```

Embedded Style Sheets

Use `<style>` … `</style>` inside the `<head>` tag

A style sheet definition contains a list of

- HTML tags
- Associated format information for that tag

```
.. .. ..
<style>
  h1 { font-size: 15pt; font-weight:bold}
  p { font: bold italic 12pt/20pt times, serif}
</style>
.. .. ..
```

External Style Sheets

- **Put all CSS in separate file and then link to it from each page**
  - In same format that it appeared in the Internal Style Sheets
- **`<link>` element references an external style sheet**
- **Should appear in the head of the document**

```
<link href="styles.css" rel="stylesheet">
```

POWERING
POTENTIAL

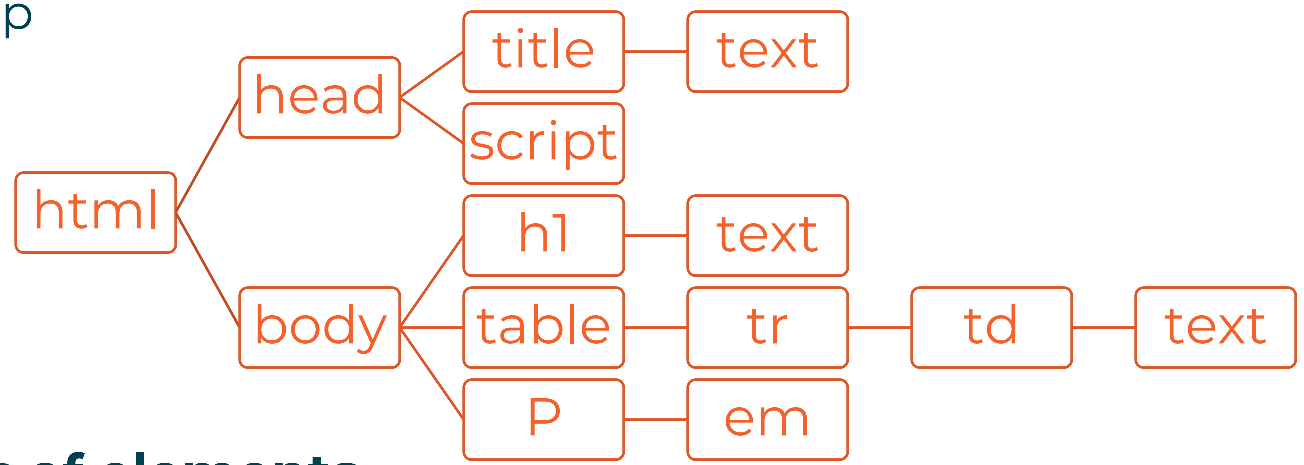
CSS and the DOM

CSS Fundamentals

Recall: The Document Object Model

HTML documents have a hierarchical structure that form the DOM

- Every element, except <html> is contained within another
- Creating a parent/child relationship



A DOM tree contains two types of elements

- Nodes
- Text

HTML markup to DOM object (1)

Consider the following HTML

```
<img id="myImage" src="image.gif" alt="An image" title="This is an image"/>
```

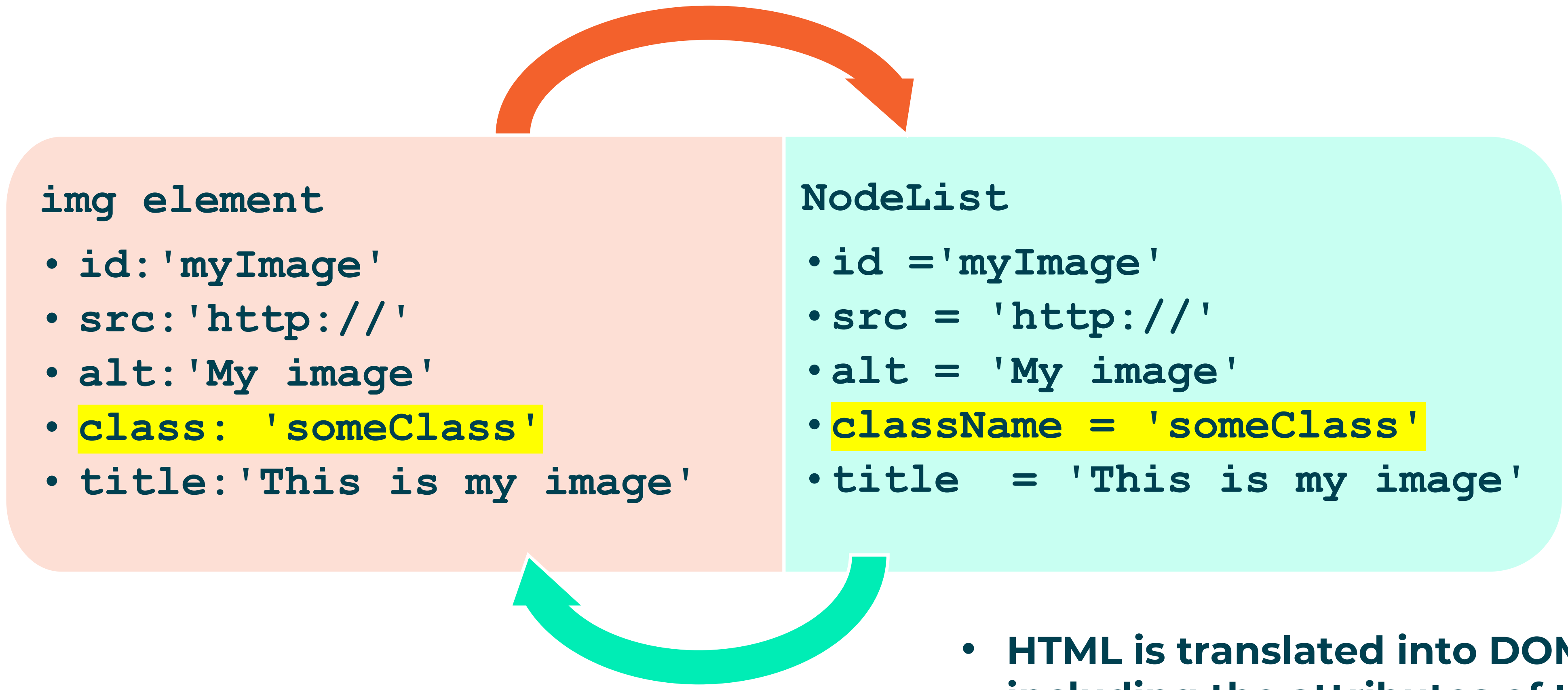
The tag has a type of `<img>` and four attributes

- `id`
- `src`
- `alt`
- `title`

The element is read and interpreted by the browser into a DOM

- Each element becomes a NodeList object
- Assigned a property based on the html attribute

HTML markup to DOM object (2)



```
img element
• id:'myImage'
• src:'http://'
• alt:'My image'
• class: 'someClass'
• title:'This is my image'
```

```
NodeList
•id ='myImage'
•src = 'http://'
•alt = 'My image'
•className = 'someClass'
•title  = 'This is my image'
```

- HTML is translated into DOM elements, including the attributes of the tag and the properties created from them
- These can be used to select and style elements

Inheritance and Selecting Elements

CSS Fundamentals

SELECTING ELEMENTS TO STYLE

- Select all tags of a particular type
- Select tags dependent on the relationship to others in the DOM
- Select tags based on their id or class attribute
- Select tags based on other attributes
- Select tags based on a combination of the above

Selecting all tags of a particular type

This is as simple as creating a rule for the tag name and nothing else.
All elements with this tag will be affected, regardless of where they are in the DOM.

- *Assuming that this is the only CSS applied to the page (more on that later…)*

```
p { color: green }
```

- Would make all text in *any* `p` element green

```
div { background-color: red }
```

- Would make the background colour of *any* `div` element red
- Multiple elements can be selected by putting a comma between them

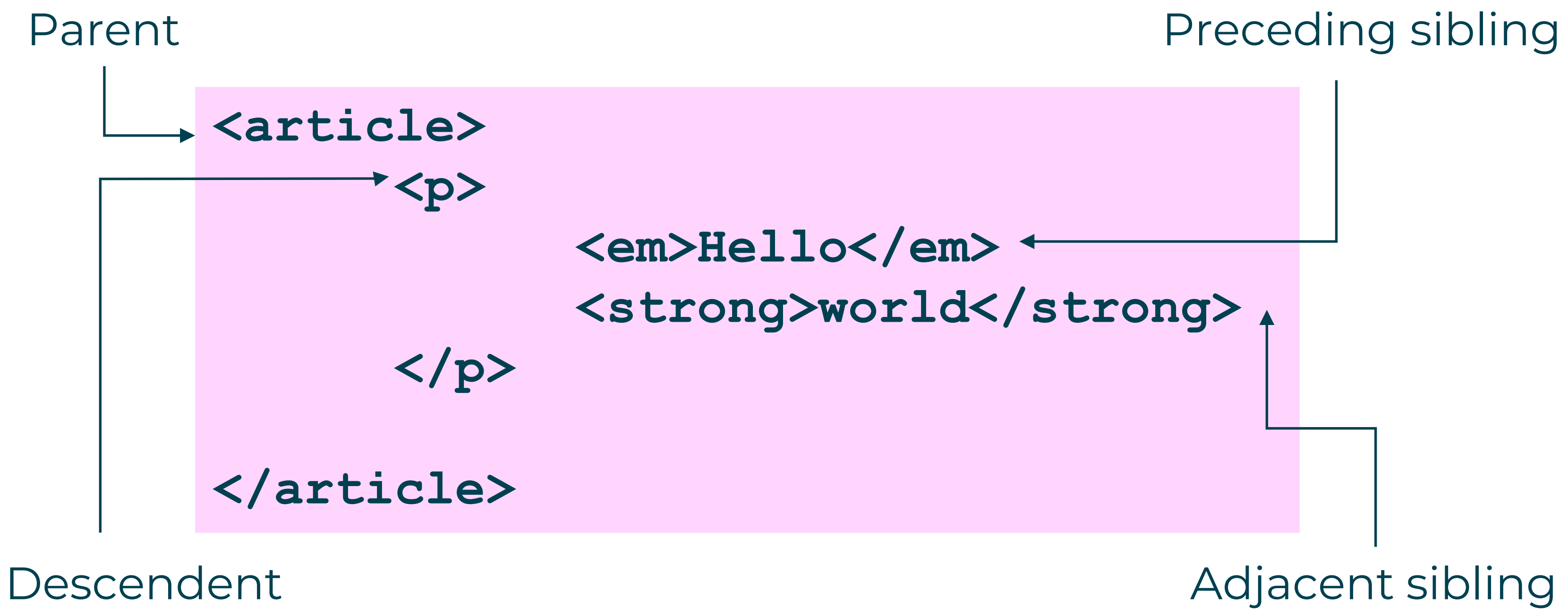
```
h1, a {color: pink }
```

- Would make the text of *any* `h1` element and *any* `a` element pink

Understanding Inheritance

HTML tags exist in a hierarchical tree from `<html>` root to text nodes.

- When a tag is surrounded by another tag, the tags are nested.



```
<article>
      <p>
            <em>Hello</em>
            <strong>world</strong>
      </p>

</article>
```

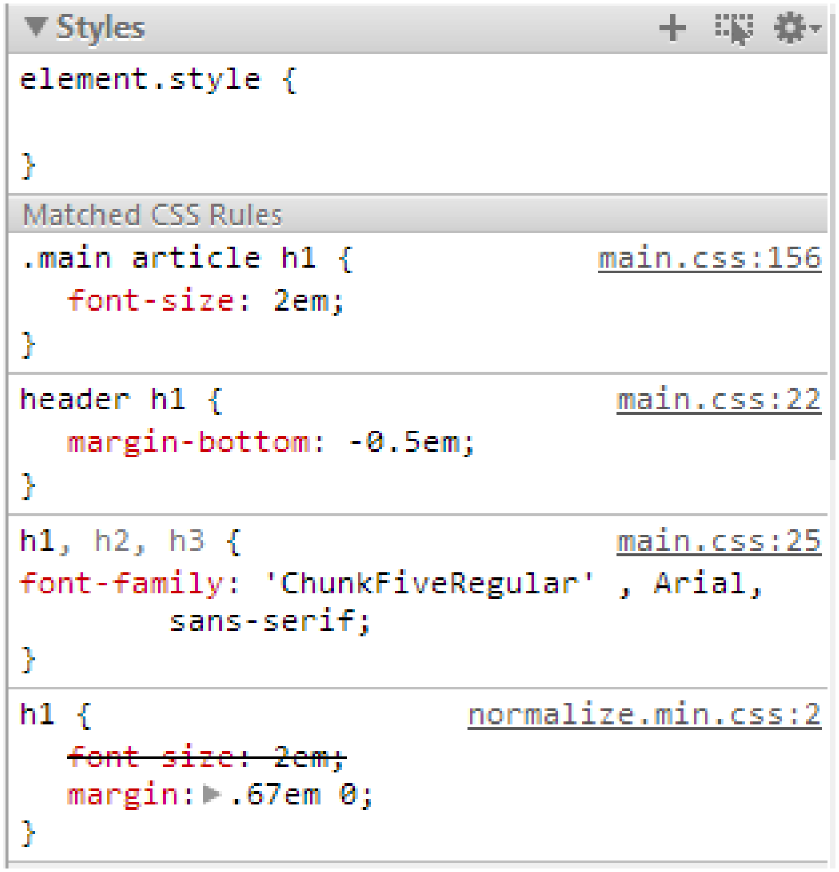
Hierarchical Inheritance

Elements inherit from containing parents

- So we only need to define a style rule at the highest level
- We can then override rules at descendent levels

Complex hierarchies are difficult to manage

- Chrome's developer tools help greatly
- Showing which styles are applied
- Where they come from
- If rules are being overridden
- The order in which they're applied



```
element.style {

}
```

```
.main article h1 {                main.css:156
   font-size: 2em;
}
header h1 {                        main.css:22
   margin-bottom: -0.5em;
}
h1, h2, h3 {                       main.css:25
font-family: 'ChunkFiveRegular' , Arial,
        sans-serif;
}
h1 {                      normalize.min.css:2
   ~~font-size: 2em;~~
   margin:▶.67em 0;
}
```

Select tags dependent on the relationship to others in the DOM

Descendant selector – put a space between the parent and child – all descendants will be styled

```
ul li { color: purple }
```

- Would make all text in *any* `li` that is a *descendant* of a `ul` purple

Child selector – put a > between the parent and child – any direct child will be styled

```
section > p { color: brown }
```

- Would make all text in *any* `p` that is a *direct descendant* of a `section` brown

Adjacent selector – put a + between the siblings – last sibling listed will be styled

```
h2 + p { color: black }
```

- Would make all text in any `p` that *immediately follows* a `h2` element black

Sibling selector – change the + for a ~ to select *any following element*

Select tags based on their id or class attribute

id selector – put a # before the name of the id to be styled

```
#myChosenId { color: purple }
```

- Would make all text in *any* **element** that has an **`id`** attribute of **`myChosenId`** **purple**
- **Note:** An **id** should be *unique* within a page, if *more than one* is needed a **class** should be used

class selector – put a . Before the name of the class to be styled

```
.myChosenClass { color: brown }
```

- Would make all text in *any* **element** that has a **`class`** attribute *of* of a **`myChosenClass`** **brown**

```
<p id="myChosenId">A paragraph with myChosenId</p>
<p class="myChosenClass">A paragraph with myChosenClass</p>
```

Selecting sets of elements with pseudo-classes

Selecting first and last element

```
ul li:first-child { background-color: red; }
ul li:last-child { background-color: red; }
```

Selecting an element by its ordering

```
li:nth-child(3), li:nth-child(5) { background-color: red; }

li:nth-child(2n + 1) { background-color: red; }

li:nth-child(odd) { background-color: blue; }

li:nth-child(even) { background-color: green; }
```

Selecting sets of elements with pseudo classes

More selection patterns

```
ul:last-child { background-color: red; }
```

```
ul:first-child:last-child { background-color: red; }
```

Selecting by types of element

```
article:first-of-type { background-color: red; }
```

```
p:last-of-type { background-color: red; }
```

Choosing what isn’t

```
input:not([type=checkbox]):not([type=radio]) {
    display: block; width: 12em;
}
```

Pseudo classes and elements

(Dynamic) pseudo classes, elements…

- Applying to user actions (pseudo classes)

```
:active { color: blue; }
:hover { color: blue; }
:focus { color: blue; }
:link { color: blue; }
:visited { color: blue; }
```

- Applying to placement (pseudo elements)

```
::after { color: blue; }
::before { color: blue; }
::first-letter { color: blue; }
::first-line { color: blue; }
```

- Selection pseudo element

```
::selection { color: blue; }
```

Pseudo classes specificity

Recall: In the cascade styles are sorted by specificity

- Latter rules are more specific that earlier rules

Hence for link pseudo classes to work use this order

```
a               {color: black; }
a:link          {color: blue; }
a:visited       {color: red; }
a:hover         {color: green; }
a:active        {color: orange; }
```

before: and after:

Used to insert content before or after an element

- Can be specific content, counters, or values of attributes

Specify style and content of inserted content

- content: normal | none | <string> | <uri> | <counter> | attr(<identifier>) | open-quote | close-quote | no-open-quote | no-close-quote | inherit

```
p.note:before { font-weight: bold; content: "Note: "; }

h1:before {
  content: "Chapter " counter(chapter) ".";
  counter-increment: chapter;
}
```

Choosing elements by their attribute

- = operator finds attributes whose value exactly matches

```
a[href="http://www.qa.com"] { color: blue; }
```

- ^= operator finds attributes starting with a value

```
a[href^="http:"] { color: blue; }
```

- $= operator finds any element attributes ending with a value

```
[src$=".png"] { color: red; }
```

- *= operator finds attributes containing the value

```
[id*="stuff"] { color: red; }
```

Quick Lab Chapter 7 – CSS Selectors

Apply selectors to style rules to apply styling to particular elements