Lesson 2 - Notes

Nathan Stilwell
Feb 4, 2013

Review

See Lesson 1 Notes

Links

The Anchor Tag - a tag used by the browser to send the user to a different location.

```
<a href="" title="" target="">Link</a>
```

The **href** attribute is used to define a location

```
<a href="[location]">Link</a>
```

Links can redirect a user to a location in the same document (on the same page) by referencing the value in an ID attribute.

```
<section id="section-1">
    This is section one.

    <a href="#section-2">Go to Section 2</a>
</section>

<section id="section-2">
    This is section two.

    <a href="#section-1">Go to Section 1</a>
</section>
```

The **ID** attribute can be referenced by using the **#** symbol. So in the example above `href="#section-2"` points to `id="section-2"`

Anchor tags can direct the user to a different page on the same domain by using a **relative path**.

Relative paths

Given the following site structure

```
▼ some-site.com
  ▼ about
      about-me.html
  ▼ blog
      blog-post-1.html
    page-1.html
    page-2.html
```

http://some-site.com/page–1.html

```
<!doctype html>
<body>
    <h1>This is Page 1</h1>
    <a href="page-2.html">Go to Page 2</a>
</body>
</html>
```

http://some-site.com/page–2.html

```
<!doctype html>
<body>
    <h1>This is Page 2</h1>
    <a href="page-1.html">Go to Page 1</a>
</body>
</html>
```

Because **Page 1** and **Page 2** are both in the same folder (some-site.com), we can infer the folder and just specify the name of the file.

http://some-site.com/blog/blog-post–1.html

```
<!doctype html>
<body>
    <h1>Blog Post 1</h1>
    ...
    <a href="../about/about-me.html">About Me</a>
</body>
</html>
```

http://some-site.com/about/about-me.html

```
<!doctype html>
<body>
    <h1>About Me</h1>
    ...
    <a href="../blog/blog-post-1.html">Best Blog Post</a>
</body>
</html>
```

When specifying relative paths, each "folder" in the directory structure has two directories for navigation, `.` which specifies *this folder* and `..` which specifies the *the folder above*.

In the given structure, on page **blog-post–1.html**, the folder `./` is the same as `blog/`. And `../` is the same as `some-site.com/`. And so the location `../about/about-me.html` is the same as `some-site.com/about/about-me.html`.

Absolute Paths

An absolute path is a fully formed URL (*universal resource locator*). When linking to different domain (a different web site), you must use an absolute path.

http://some-site.com/blog/blog-post–1.html

```
<!doctype html>
<body>
    <h1>Blog Post 1</h1>
    ...
    <a href="http://some-tumblr.tumblr.com">My Tumblr</a>
</body>
</html>
```

Anatomy of a URL

`{protocol}://{subdomain}.{domain}.{top-level-domain}/{path}/{file}#{anchor}?{query-string}`

Example

http://store.apple.com/us/browse/home/shop_mac/family/macbook_pro#product-selection–3

Email links

Emails can be composed from links (weird, I know) by specifying `mailto` as the protocol in the href. You can also do some neat tricks with the query string.

```
<a href="mailto:your@mom.com?subject=Hi Mom&body=I'm having fun at school">Ema
```

Open links in a new tab

To have links open in a new tab (or new window, depending on the browser), provide `_blank` as the `target` attribute.

```
<a href="http://f.oo/" target="_blank">Link!</a>
```

CSS Continued

Where to define Style

Inline, using the `style` attribute

```
<div class="some-div" style="color : tomato;"></div>
```

A **style tag**, which should be placed in the head of the document.

```
<html>
  <head>
    <style>
      p { color : tomato; }
    </style>
  </head>
  ...
</html>
```

A **linked stylesheet**, which are styles in a seperate file loaded into the page with a `link` tag (also in the head)

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

Selectors (Continued)

Type Selector - select by tag name

html

```
<h1>Some title</h1>
```

css

```
h1 {
  font-size : 100px;
  font-weight : bold;
}
```

Class Selector - select by class attribute

html

```
<div class="my-div">blah blah blah </div>
```

css

```
.my-div {
  font-size : 100px;
  font-weight : bold;
}
```

An **ID selector** - select by id attribute

html

```
<div id="my-div">blah blah blah </div>
```

css

```
#my-div {
  font-size : 100px;
  font-weight : bold;
}
```

Descentant Selectore - select tags inside of tags

html

```
<div class="my-div">
  <p>
    some stuff
  </p>
</div>
```

css

```
.my-div p {
  color : purple;
}
```

Adjacent sibling - The + combinator matches the second element only if it is immediately following the first element

html

```
<div class="my-div">
  <p class="test-p">
    some stuff
  </p>

  <p>
    this is purple
  </p>

  <p>
    this is not purple
  </p>
</div>
```

css

```
.my-div .test-p + p {
  color : purple;
}
```

General sibling - The ~ combinator separates two selectors and matches the second element only if it is preceded by the first, and both share a common parent.

html

```
<div class="my-div">
  <p class="test-p">
    some stuff
  </p>

  <p>
    this is purple
  </p>


  <p>
    and this is too
  </p>
</div>
```

css

```
.my-div .test-p ~ p{
  color : purple;
}
```

Child Selector - The > combinator separates two selectors and matches only those elements matched by the second selector that are direct children of elements matched by the first.

html

```
<div class="my-div">
   <p>
     <!-- child of p -->
     <span class="child">purple</span>
     <div>
       <!-- grandchild of p -->
       <span class="child">not purple</span>
     </div>
  </p>
</div>
```

css

```
.my-div p > .child {
  color : purple;
}
```

Specificity

(W3C specificity)[http://www.w3.org/TR/css3-selectors/#specificity] rules state that types of selectors have different orders of precedence. In order they are:

Inline - 1000
ID - 100
Class - 10
Type - 1

So a selector can have a calculated specificity by assigning precedence values to it's definition. For example:

```
.my-div p > .child {}
```

would get 21 "points", while

```
#h1 {}
```

would get 100 "points". Therefore the id selector will override the class and type selector.

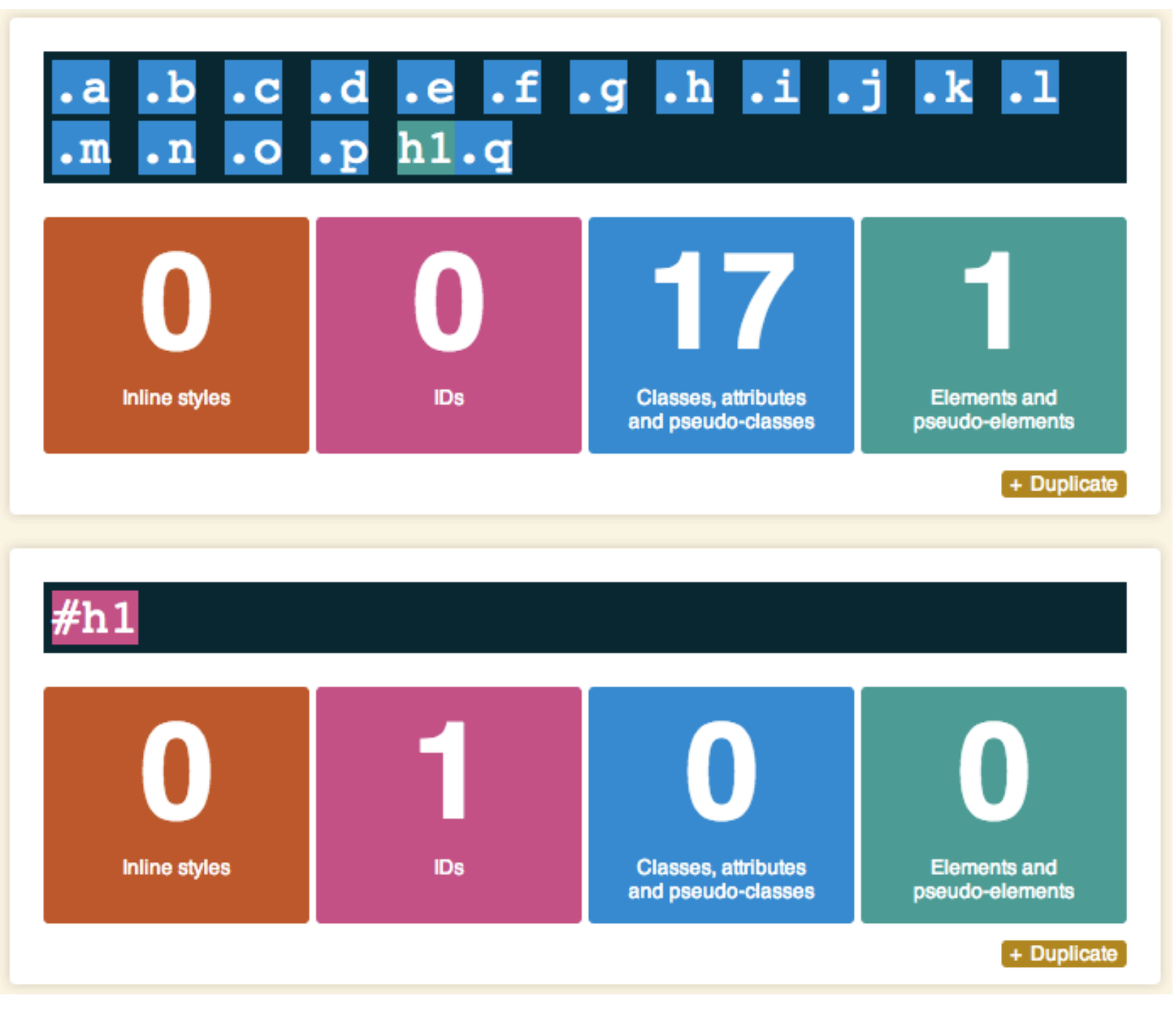
This theory falls apart when trying to override an id selector with class selectors, for instance:

```
.a .b .c .d .e .f .g .h .i .j .k .l .m .n .o .p h1.q {}
```

won't beat

```
#h1 {}
```

even though based on our points system, the first example has 171 points and the second has 100. Using Keegan's Specificity calculator, we can see this more clearly.



So a category of lower precedence can not override a category of higher precidence. Keegan's Specificity Calculator can be found here.

The Cascade

On top of the weight system defined by CSS specificity, selectors with equal weight can be overridden based on where the style was loaded from. If selectors are equal weight, they're precedence is defined by the what was "read" last. The order in which styles are "read" are:

Linked Styles, top to bottom
Internal Styles (`<style> tag`), top to bottom
Inline Styles, top to bottom

So after determining a selectors specificity, we must determine in which order it was loaded into the document.