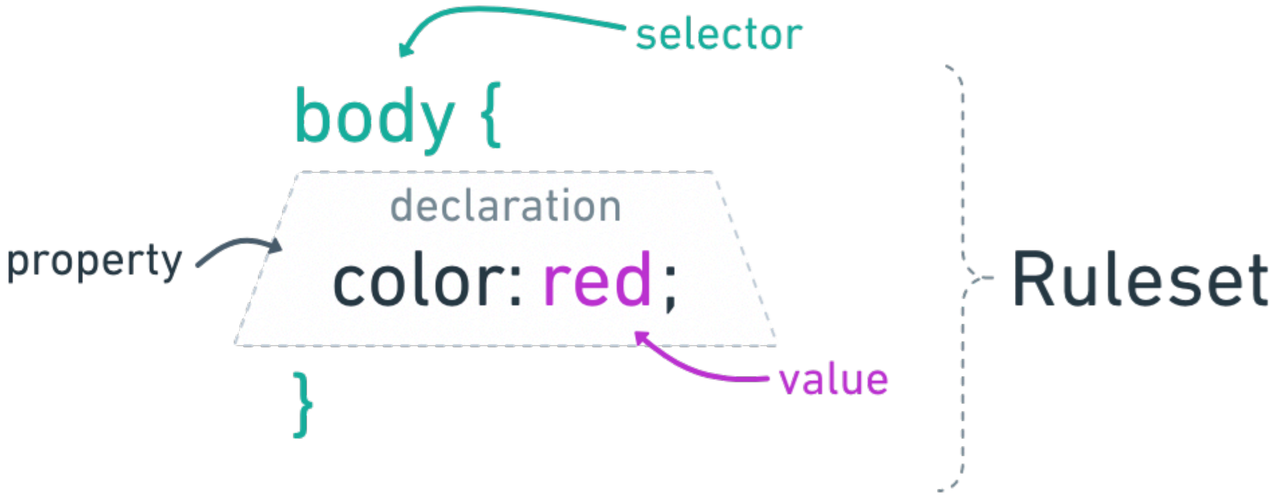
CSS Refresher

Selectors, Properties, and Values



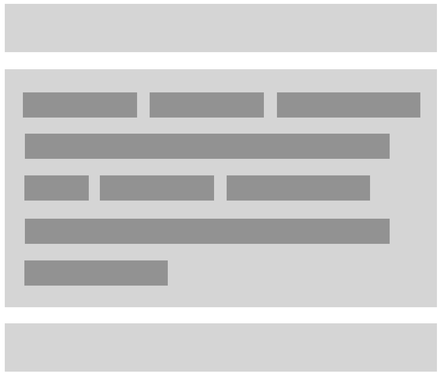
CSS Defaults

Block Elements

<address>	<article>	<aside>	<blockquote>	<canvas>	<dd>
	<dl>	<dt>	<fieldset>	<figcaption>	<figure>
<footer>	<form>	<h1>–<h6>	<header>	<hr>	
<main>	<nav>	<noscript>		<p>	<pre>
<section>	<table>	<tfoot>		<video>	

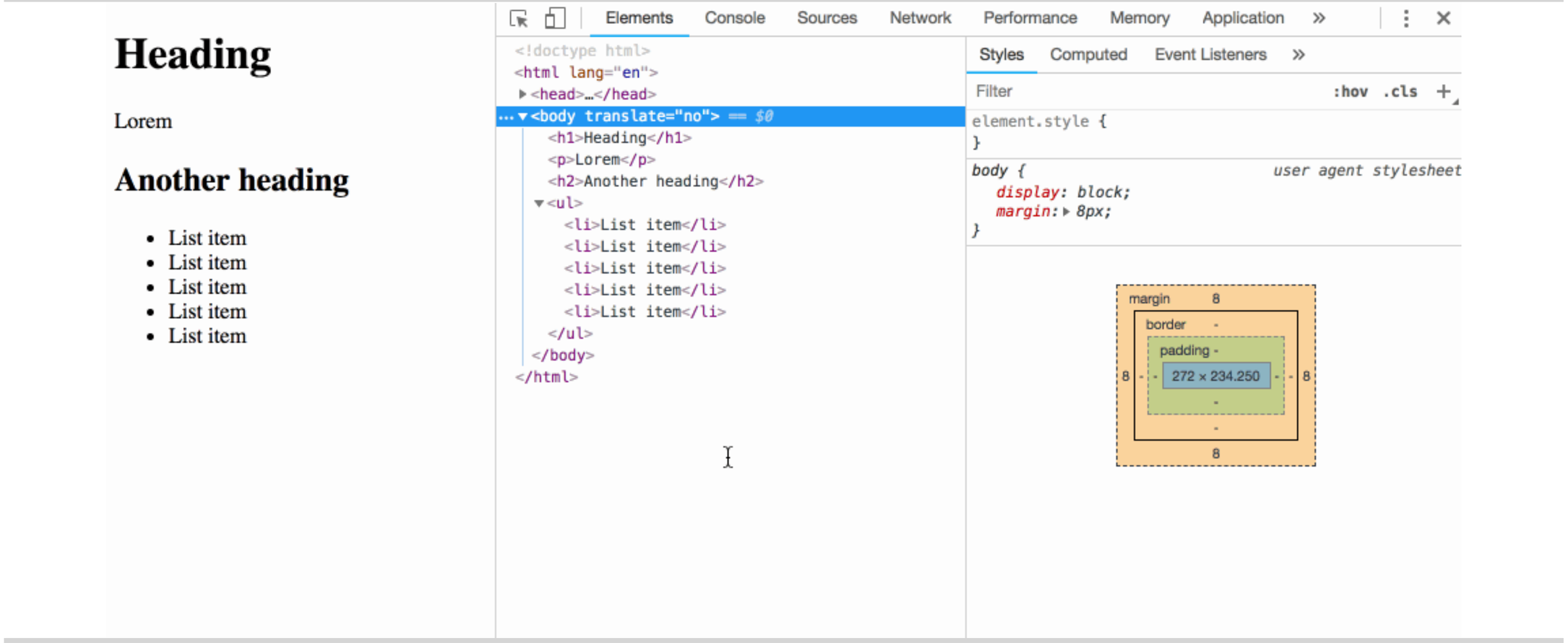
Inline Elements

<a>	<abbr>	<acronym>	<b>	<bdo>	<big>
 	<button>	<cite>	<code>	<dfn>	
<i>	<img>	<input>	<kbd>	<label>	<map>
<object>	<output>	<q>	<samp>	<script>	<select>
<small>			<sub>	<sup>	<textarea>
<time>	<tt>	<var>			



CSS Defaults

Typography



Class & ID

Use class names over bare elements or IDs

```
<h1 class="page-title cursive"></h1>
```

Class name selectors start with a **period**

```
.page-title {
  text-transform: uppercase;
}
```

Don’t forget:

- Rules must end with a semicolon
- Use the `text-transform` property rather than setting the case in HTML

Class & ID

Use an ID only if the element is truly one-of-a-kind

```
<p id="copyright-notice">© 1920</p>
```

ID selectors start with a **hash** character

```
#copyright-notice{
    font: italic 9px/1.3 Helvetica, sans-serif;
    color: #999;
}
```

Comments and Organization

Group rulesets by type and label them using `/*` and `*/` for comments

```
/*
 * === SITE HEADER ===
 */

header h1 { … }
header h2 { … }
header p { … }

/*
 * --- Header nav ---
 */

header nav button { … }
header nav label { … }
header nav img { … }
```

Things to Avoid

Don’t use empty divs to achieve your styles

```
<div class="something-fancy">
    <div class="a-normal-element"></div>
</div>
```

Only use !important as a last resort (and probably not even then)

```
.body-text{
    color: black;
}

p{
    color: red !important;
}
```

Things to Avoid

You don’t need to include units for measurements of zero

```
div {
  margin: 0px;
  padding: 0px;
  width: 0px;
}
```

```
div {
  margin: 0;
  padding: 0;
  width: 0;
}
```

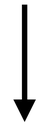
Things to Avoid

Don’t go overboard with selectors and specificity

```
<header class="site-header">
  <nav class="nav-bar">
    <ul class="nav-menu">
      <li>…</li>
      <li>…</li>
      <li>…</li>
    </ul>
  </nav>
</header>
```

✗

```
header.site-header > nav.nav-bar >
ul.nav-menu > li.nav-item{
  …
}
```



✓

```
.site-header .nav-menu li{
    …
}
```

Things to Avoid

Make as few assumptions about the markup as possible

X

```
.highlight a { background: yellow; }
```

```
<span class="highlight">
  <a href="#">Highlighted link</a>
</span>
```

✓

```
.highlight { background: yellow; }
```

```
<a href="#" class=“highlight">
Highlighted link</a>
```

Things to Avoid

Don’t create rules that you immediately override

X

```
li {
  display: inline-block;
  margin-left: 14px;
}

li:first-of-type { margin-left: 0; }
```

✓

```
li { display: inline-block; }
li + li { margin-left: 14px; }
```

Things to Avoid

Don’t make HTTP requests for resources that can be replicated with simple CSS

X

```
li::before {
  content: url(green-circle.svg);
}
```

✓

```
li::before {
  background: green;
  border-radius: 50%;
  content: "";
  display: block;
  height: 20px;
  width: 20px;
}
```

CSS Structure

Your CSS file should have 4 distinct sections in the following order:

1. Cross-browser reset
2. Utilities & variables
3. Base styles
4. Custom styles

CSS Structure

1. Cross-browser reset

Exercise: Learn about Resets vs Normalization

- Break into groups and learn about one or the other, then report back
- Details and links in Slack…

CSS Structure

2. Utilities and Variables

Use classes to bundle modular rules

```
.u-text-center { text-align: center; }
.u-text-left { text-align: left; }
.u-text-right { text-align: right; }
```

```
<p class="u-text-center">…</p>
```

```
.u-bg-contain { background-size: contain; }
.u-bg-cover { background-size: cover; }
```

https://gist.github.com/samizdatco/251c0d5513aaba478fcb6082ac9e6e62

CSS Structure

2. Utilities and Variables

Set variables to use consistent colors and sizes throughout

```
:root {
  --text-color: #222;
  --title-color: red;
}

h1, h2, h3 {
  color: var(--text-color);
}

p{
  color: var(--text-color);
}
```

```
:root {
  --page-width: 1024px;
  --col-width: 720px;
}

main{
  max-width: var(--page-width);
}

article{
  width: var(--col-width);
}
```

CSS Structure

3. Base Styles

```
body {
  background: whitesmoke;
  color: #1b203a;
  font-family: "Zilla Slab", serif;
  margin: 0;
}

h1, h2, h3, h4, h5, h6, p, li {
  line-height: 1.5625em;
  margin: 0;
}

a:hover{
  text-decoration: none;
  color: skyblue;
}
```

CSS Structure

4. Custom Styles

Use classes, IDs, and nested selections to style non-standard parts of the page

```
#email-contact-info { … }
.product-page header img { … }
ul > li ol li { … }
footer ~ * { … }
p + p { … }
code.javascript { … }
```

Naming Things

Choosing names is a fundamental activity in all forms of coding. When picking class- and ID-names, try to make them:

- Understandable
- Obvious
- Functional
- Consistent
- Non-redundant