IRON
HACK

HTML & CSS BEST PRACTICES

HTML - Follow Standard Markup

- Nest and close all elements correctly
- Use ids and classes appropriately
- Always close your tags

Take a look at this code:

```
<p id="intro        rang    f th
Wall to investiga          idings:
and Gared</p></str
<p id="intro">T      ound       mber of wilding corpses dismembered.
```

Can anyone find the 3 mistakes?

HTML - Follow Standard Markup

```
<p id="intro">Three rangers of the Night’s Watch depart from the
Wall to investigate the wildings: <strong>Ser Waymar Royce, Will
and Gared </p></strong

<p id="intro">They found a number of wilding corpses
dismembered.
```

?

HTML - Follow Standard Markup

Solution:

```
<p class="intro">Three rangers of the Night’s Watch depart from the
Wall to investigate the wildings: <strong>Ser Waymar Royce, Will
and Gared</strong></p>
```

```
<p class="intro">They found a number of wilding corpses
dismembered.</p>
```

HTML - Use Semantic Elements

- **Each HTML tag has a meaning**, talks about the purpose of that element in your code.
- Use them to describe the content and improve your code's organization.
- The readability will increase and styling will be much easier.

HTML - Use Semantic Elements

```html
<div>
  The Night’s Watch Blog
</div>
<div>
  <a href="about_us.html">About us</a>
</div>
<div>
  Last news from the Wall
</div>
<div>
  Copyleft 7 Kingdoms 2015 ;)
</div>
```

```html
<header>
  The Night’s Watch Blog
</header>
<nav>
  <a href="about_us.html">About us</a>
</nav>
<article>
  Last news from the Wall
</article>
<footer>
  Copyleft 7 Kingdoms 2015 ;)
</footer>
```

IRON HACK

HTML - Keep the syntax organized

As pages grow, managing HTML can become a hard task.

To help you with this, we recommend you to follow the next simple rules:

HTML - Keep the syntax organized

Use lowercase for all your elements

```
<DIV>
  <H1 HIDDEN='HIDDEN'>Upcase is bad</H1>
  <P>Don't do this!</P>
</DIV>
```

```
<div>
  <h1 hidden='hidden'>Lowcase is great</h1>
  <p>Do this instead</p>
</div>
```

HTML - Keep the syntax organized

- Strictly use double quotes, not single or completely omitted quotes

```html
<div id='wrong-div'>
  <p class=wrong-p>Don't do this!</p>
</div>
```

```html
<div id="correct-div">
  <p class="correct-p">Do this!</p>
</div>
```

HTML - Keep the syntax organized

- Boolean attributes (hidden, disabled, selected, checked...) don't need a value. Their presence is enough.

```
<p hidden>Hidden text</p>


<form>
  <input type="text" name="disabled_field" disabled>
</form>
```

HTML - Practical use of ID & Class Values

ID and classes values need to be related to the content itself, not the style of the content.

Take a look at this code:

```
<p class="red">The wildlings are coming!</p>
```

HTML - Practical use of ID & Class Values

The HTML here assumes that this alert message will be red.

Using a value of red to describe red text isn’t ideal, as it describes the presentation of the content.

If the style of the alert changes to orange, the class name of red will no longer make sense, forcing you to change both HTML and CSS.

```html
<p class="red">The wildlings are coming!</p>
```

HTML - Practical use of ID & Class Values

So we change the class name to be a description of the content's purpose.

Later in css we will take care of giving it the correct color.

```html
<p class="alert">The wildlings are coming!</p>
```

HTML - Images ALT attribute

Images should always include the alt attribute. Screen readers, google spiders and other accessibility software rely on the alt attribute to provide context for images

```
<img src="jonsnow.jpg"> ✗
```

```
<img src="jonsnow.jpg" alt="Jon Snow"> ✓
```

HTML - Images ALT attribute

The alt attribute value should be very descriptive of what the image contains.

```
<img src="jonsnow.jpg" alt="Jon Snow is
eating a waffle while he watches The Wall">
```

HTML - `<img>` tag vs. background image

If an image isn’t a meaningful part of the content—perhaps it’s just there as a user interface detail, for example—it should be included as a CSS background image, not as an <img> element.

HTML - `<img>` tag vs. background image

For example, on the Ironhack Web site many images aren’t part of the content. They are there to support the content: the text. Those image should probably be background images set in the CSS.

By contrast, on an image site like Instagram user images are a major part of the content. They should be <img> tags.

Exercise

Create markup for a blog post with a comment form and a couple comments:

- Use different and appropriate containers for the blog post, the comment form and the comments.
- The blog post must have: title, subtitle, date, author (with a link to their website), text, and image, a list, and some pull quotes.
- The comment form must have: name field, email field, space to write, a "remember my data" option and a submit button.

CSS - Organize with Comments

- CSS files can become quite extensive, spanning hundreds of lines. This makes finding and editing our styles nearly impossible.
- Let’s keep our styles organized in logical groups.
- Before each group, let’s provide a comment noting what the following styles pertain to.

CSS - Organize with Comments

```css
/* Primary header */

header { ... }

/* Featured article */

article { ... }

/* Buttons */

.btn { ... }
```

CSS - Build Proficient Selectors

- The longer a selector is and the more qualifiers it includes, the higher specificity it will contain.
- And the higher the specificity the more likely a selector is to break the CSS cascade and cause undesirable issues.

```
#aside #featured ul.news li a { ... }

#aside #featured ul.news li a em.special { ... }
```

CSS - Build Proficient Selectors

- Don’t use IDs within selectors.
- They are overly specific, quickly raise the specificity of a selector, and quite often break the cascade within our CSS files

```
#aside #featured ul.news li a { ... }

#aside #featured ul.news li a em.special { ... }
```

CSS - Build Proficient Selectors

- Let's use shorter and primarily direct selectors. Nest them only two to three levels deep, and remove as many location-based qualifying selectors as possible.

```
.news a { ... }

.news .special { ... }
```

CSS - Use Specific Classes

- There are times when a CSS selector is so long and specific that it no longer makes sense.
- It creates a performance lag and is strenuous to manage.

```
section aside h1 em { ... }
```

CSS - Use Specific Classes

- In this case, using a class alone is advised.
- While applying a class to the targeted element may create more code within HTML, it will allow the code to render faster and will remove any managing obstacles.

`section aside h1 em { ... }` $\longrightarrow$ `.text-offset { ... }`

CSS - Indent your CSS

- This makes your code far easier to read for fellow coders

```
selector,
.selector-secondary,
.selector[type="text"] {
··padding: 15px;
··margin-bottom: 15px;
··background-color: rgba(0,0,0,0.5);
··box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;
}
```

CSS - Each selector in a single line

- When grouping selectors, keep individual selectors in a single line

```css
selector,
.selector-secondary,
.selector[type="text"] {
  padding: 15px;
  margin-bottom: 15px;
  background-color: rgba(0,0,0,0.5);
  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;
}
```

CSS - Use of Space

- Include one space before the opening brace of declaration blocks for legibility.
- Place closing braces of declaration blocks on a new line. Include one space after : for each declaration.

```css
selector,
.selector-secondary,
.selector[type="text"]·{
  padding:·15px;
  margin-bottom:·15px;
  background-color:·rgba(0,0,0,0.5);
  box-shadow:·0 1px 2px #ccc, inset 0 1px 0 #fff;
}
```

CSS - End all Declarations with a Semicolon

The last declaration's is optional, but your code is more error prone without it.

```css
selector,
.selector-secondary,
.selector[type="text"] {
  padding: 15px;
  margin-bottom: 15px;
  background-color: rgba(0,0,0,0.5);
  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;
}
```

CSS - Comma-separated Property Values

They should include a space after each comma (e.g., `box-shadow`). But don't separate values inside `rgb()`, `rgba()`, `hsl()` or `hsla()`.

```css
selector,
.selector-secondary,
.selector[type="text"] {
  padding: 15px;
  margin-bottom: 15px;
  background-color: rgba(0,0,0,0.5);
  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;
}
```

CSS - Use of 0

When a value is 0, don’t add a unit. This reduces clutter and improves readability.

```css
selector,
.selector-secondary,
.selector[type="text"] {
  padding: 15px;
  margin-bottom: 15px;
  background-color: rgba(0,0,0,0.5);
  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;
}
```

IRON HACK

CSS - HEX Color Values

Lowercase all hex values, e.g., `#fff`. Lowercase letters are much easier to discern when scanning a document. Also, use the shorthand form when possible (e.g., `#fff` instead of `#ffffff`).

```css
selector,
.selector-secondary,
.selector[type="text"] {
  padding: 15px;
  margin-bottom: 15px;
  background-color: rgba(0,0,0,0.5);
  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;
}
```

CSS - Quote Attribute Values in Selectors

They're the only way to guarantee code renders the same in any environment.

```
selector,
.selector-secondary,
.selector[type="text"] {
  padding: 15px;
  margin-bottom: 15px;
  background-color: rgba(0,0,0,0.5);
  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;
}
```

Exercise

Clone the HTML file here: CSSPlayground

- Follow the directions for each div.
- Use the correct selector for each div to change their background color, #red should be red, .blue should be blue.
- Add a margin of 10px to all of the boxes.
- Add a red border to blue divs.
- Add a blue border to red divs.
- Give the direction H3 a padding of 20% on all sides using short hand notation. See what happens.