Udacity Frontend Nanodegree Style Guide

Introduction

This style guide acts as the official guide to follow in your projects. Udacity evaluators will use this guide to grade your projects. There are many opinions on the "ideal" style in the world of Front-End Web Development. Therefore, in order to reduce the confusion on what style students should follow during the course of their projects, we urge all students to refer to this style guide for their projects.

General Formatting Rules

Capitalization

Use only lowercase.

All code has to be lowercase. This applies to CSS selectors, properties and property values (with the exception of strings).

Not Recommended:

```
color: #E5E5E5;
```

Recommended:

```
color: #e5e5e5;
```

Trailing Whitespace

Remove trailing white spaces.

Trailing white spaces are unnecessary and can complicate diffs.

Not Recommended:

```
border: 0;__
```

Recommended:

```
border: 0;
```

If using Sublime Text, this can be done automatically each time you save a file by adding the following to your User Settings JSON file (you should be able to find this within Sublime Text's menu):

```
"trim_trailing_white_space_on_save": true
```

Indentation

Indentation should be consistent throughout the entire file. Whether you choose to use tabs or spaces, or 2-spaces vs. 4-spaces - just be consistent!

General Meta Rules

Encoding

Use UTF-8 (no BOM).

Make sure your editor uses UTF-8 as character encoding, without a byte order mark. Do not specify the encoding of style sheets as these assume UTF-8.

Comments

Explain code as needed, where possible.

Use comments to explain code: What does it cover, what purpose does it serve, and why is the respective solution used or preferred?

Action Items

Mark todos and action items with `TODO:`.

Highlight todos by using the keyword `TODO` only, not other formats like `@@`. Append action items after a colon like this: `TODO: action item`.

Recommended:

```
/* TODO: add button elements */
```

CSS Style Rules

CSS Validity

Use valid CSS.

Using valid CSS is a measurable baseline quality that ensures proper CSS usage and allows you to spot CSS code that may not have any effect and can be removed.

ID and Class Naming

Use meaningful or generic ID and class names.

Instead of presentational of cryptic names, always use ID and class names that reflect the purpose of the element in question or that are otherwise generic.

Names that are specific and reflect the purpose of the element should be preferred as these are most understandable and the least likely to change.

Generic names are simply a fallback for elements that have no particular meaning different from their siblings. They are typically needed as helpers.

Not Recommended:

```
.p-998 { … }
.btn-green { … }
```

Recommended:

```
.gallery { … }
.btn-default { … }
```

Type Selectors

Avoid qualifying ID and class names with type selectors.

Unless necessary (for example, with helper classes), do not use element names in conjunction with IDs or classes. Avoiding unnecessary ancestor selectors is useful for performance reasons.

It is also considered bad practice to use IDs in your CSS files. There are no situations where IDs provide a benefit over classes. If you need to use a unique name for an element, use a class. (The only benefit IDs provide is speed, and is only beneficial on pages with thousands of similar elements.)

Not Recommended:

```
ul#example { … }
div.error { … }
```

Recommended:

```
.example { … }
.error { … }
```

🔗 Shorthand Properties

Shorthand should be used.

CSS offers a variety of shorthand properties (like `padding` rather than explicitly setting `padding-top`, `padding-bottom`, etc.) that should be used whenever possible, with the exception of font properties and properties intended to override properties of the same name in a framework such as Bootstrap.

Using shorthand properties is useful for code efficiency and understandability. The `font` shorthand property is recommended when setting all font related properties but is not required when making minor modifications. When using the font shorthand property, keep in mind that if font size and family are not included browsers will ignore entire font statement.

Not Recommended:

```
border-top-style: none;
font-family: palatino, georgia, serif;
font-size: 100%;
line-height: 1.6;
```

```
padding-bottom: 2em;
padding-left: 1em;
padding-right: 1em;
padding-top: 0;
```

Recommended:

```
border-top: 0;
font: 100%/1.6 palatino, georgia, serif;
padding: 0 1em 2em;
```

0 and Units

Omit unit specification after `0` values.

Not Recommended:

```
margin: 0em;
padding: 0px;
```

Recommended:

```
margin: 0;
padding: 0;
```

Leading 0s

Include leading `0`s in decimal values for readability.

Not Recommended:

```
font-size: .8em;
```

Recommended:

```
font-size: 0.8em;
```

Hexadecimal Notation

Use 3-character hexadecimal notation where possible.

Not Recommended:

```
color: #eebbcc;
```

Recommended:

```
color: #ebc;
```

ID and Class Name Delimiters

Separate words in ID and class names by a hyphen (`-`).

Use hyphens to concatenate words and abbreviations in selectors in order to improve understanding and scannability.

One exception: underscores (`_`) are also acceptable separators when writing BEM style CSS selectors.

Not Recommended:

```
.demoimage { … }
.errorStatus { … }
```

Recommended:

```
.demo-image { … }
.error-status { … }
```

Hacks

Avoid user agent detection as well as CSS "hacks"—try a different approach first.

It's tempting to address styling difference over user agent detection or special CSS filters, workaround and hacks. Both approaches should be considered an absolute last resort in order to achieve and maintain an efficient and manageable code base. Consider if the intended style is absolutely critical to the functionality of your application or can the "offending" user agent "live without it".

CSS Formatting Rules

Block Content Indentation

Indent all block content, that is rules within rules as well as declarations to reflect hierarchy and improve understanding.

Recommended:

```
@media screen, projection {
    html {
        background: #fff;
        color: #444;
    }
}
```

Declaration Stops

Use a semicolon after every declaration for consistency and extensibility reasons.

Not Recommended:

```
.test {
    display: block;
    height: 100px
}
```

Recommended:

```
.test {
    display: block;
    height: 100px;
}
```

Property Name Stops

Always use a space after a property name's colon, but no space between property and colon, for consistency reasons.

Not Recommended:

```
font-weight:bold;
padding : 0;
margin :0;
```

Recommended:

```
font-weight: bold;
padding: 0;
margin: 0;
```

Declaration Block Separation

Always use a single space between the last selector and the opening brace that begins the declaration block.

Not Recommended:

```
.video-block{
    margin: 0;
}

.audio-block
{
    margin: 0;
}
```

Recommended:

```
.video-block {
    margin: 0;
}

.audio-block {
    margin: 0;
}
```

Selector and Declaration Separation

Always start a new line for each selector and declaration.

Not Recommended:

```
h1, h2, h3 {
    font-weight: normal; line-height: 1.2;
}
```

Recommended:

```
h1,
h2,
h3 {
    font-weight: normal;
```

```
    line-height: 1.2;
}
```

Rule Separations

Always put a blank line (two line breaks) between rules.

Recommended:

```
html {
    background: #fff;
}

body {
    margin: auto;
    width: 50%;
}
```

CSS Quotation Marks

Use double quotation marks for attribute selectors or property values. Do not use quotation marks in URI values (`url()`).

Not Recommended:

```
@import url("css/links.css");

html {
    font-family: 'Open Sans', arial, sans-serif;
}
```

Recommended:

```
@import url(css/links.css);

html {
    font-family: "Open Sans", arial, sans-serif;
}
```

CSS Meta Rules

Section Comments

If possible, group style sheet sections together by using comments. Separate sections with new lines.

Recommended:

```
/* Header */
.header {
    …
}

.header-nav {
    …
}
```

```
/* Content */
.gallery {
    …
}

.gallery-img {
    …
}


/* Footer */
.footer {
    …
}

.footer-nav {
    …
}
```