AlarmBuddy Style Guide

Team Members:
Joshua Ringling: ring7885@stthomas.edu
Hannah Balut: balu2673@stthomas.edu
Alex Notch: notc6377@stthomas.edu
Mitzi Bustamante Chicaiza: bust6340@stthomas.edu

Background:

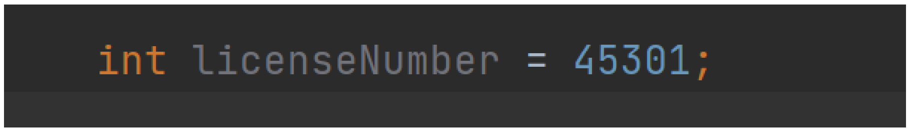
This style guide is to be used as a reference for all work being done related to the AlarmBuddy project. The attached guides were heavily inspired by style guidelines provided by Google under the CC-By 3.0 License, which encourages the sharing of these and the original source documents. The original style guides provided by Google can be found at https://google.github.io/styleguide/

Documentation Guidelines:

- Any services requiring accounts being used should have detailed instructions on how to create and login to an account.
- Any software or tools used in production such as IDEs should have detailed instructions on downloading and installing the software such that settings are the same used while making the project.

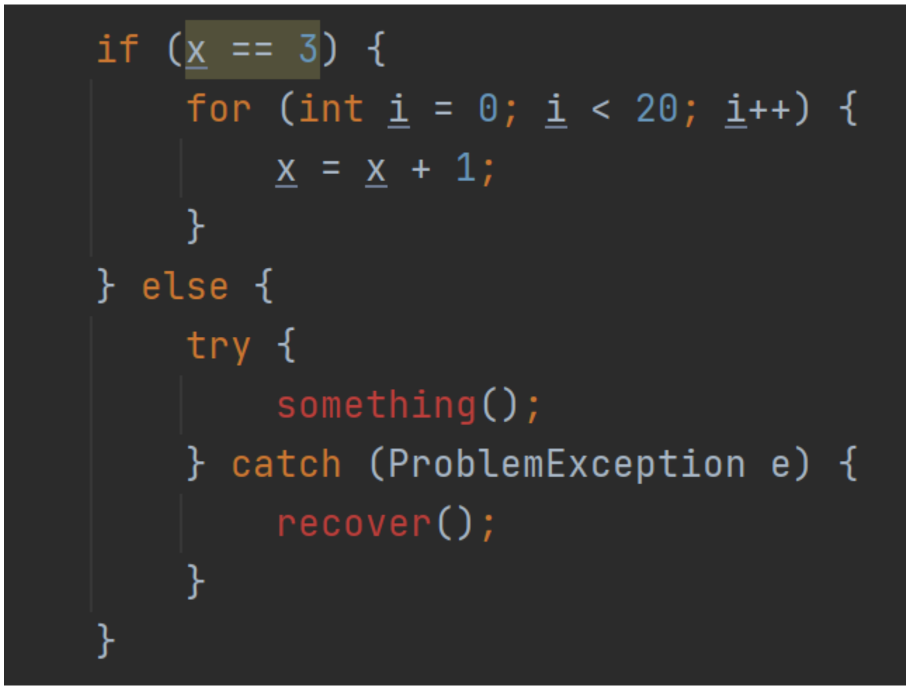
General Rules:

- Variables start with lower case letters and use context appropriate names. Use camel-case for variables that use multiple words.



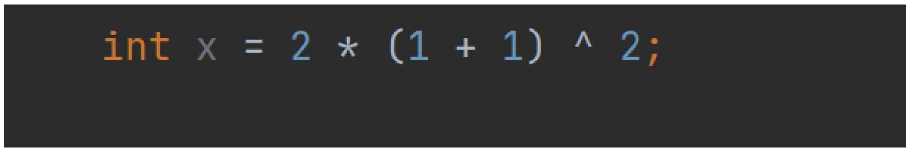
```
int licenseNumber = 45301;
```

- Indent using tab to properly show scope. Examples: if statements, for loops, while loops, and functions.
- Use curly braces on all if-else statements, loops, and functions even when they are empty or only contain one line.
- Curly braces follow the Kernighan and Ritchie style (or "Egyptian brackets") for code blocks:
    - No line break before the opening brace.
    - Line break after the opening brace.
    - Line break before the closing brace.
    - Line break after the closing brace *only if* that brace terminates the current body of a method, constructor, or named class. For example, there is no line break after a brace if an else is being used.



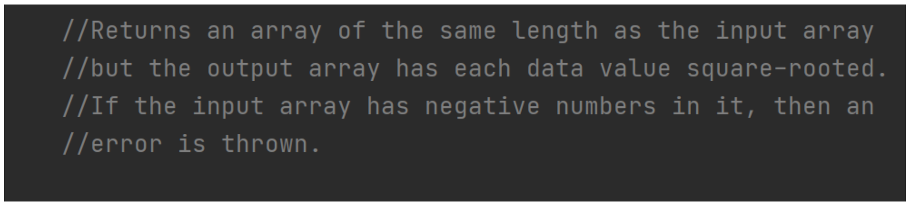
```java
if (x == 3) {
    for (int i = 0; i < 20; i++) {
        x = x + 1;
    }
} else {
    try {
        something();
    } catch (ProblemException e) {
        recover();
    }
}
```

- In mathematical operations, have spaces between all operators and operands *except* for parenthesis and use parenthesis to show clear mathematical steps.



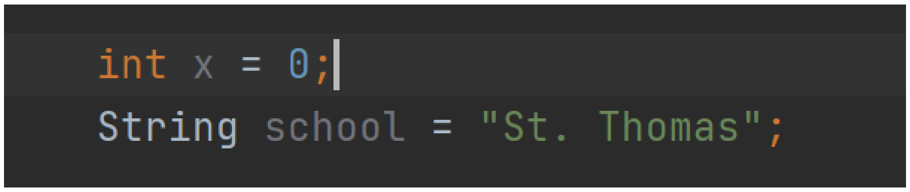
```java
int x = 2 * (1 + 1) ^ 2;
```

- Comment the use of functions as well as any special exceptions that may happen.
- Avoid making comments longer than 120 characters long. Try to wrap excessively long sections onto the next line.



```java
//Returns an array of the same length as the input array
//but the output array has each data value square-rooted.
//If the input array has negative numbers in it, then an
//error is thrown.
```

- Try to initialize variables when declaring them. However, this is not always possible.



```java
int x = 0;
String school = "St. Thomas";
```

HTML/CSS Style Guide

Joshua Ringling: ring7885@stthomas.edu

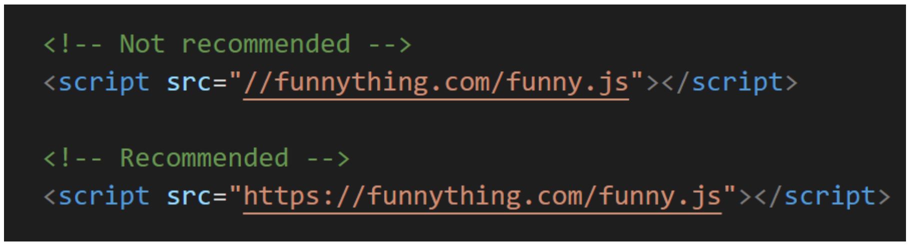
Background:

This document defines the formatting and style rules for HTML and CSS. It applies to raw, working files that use HTML and CSS coding.

General Rules:

Protocol:

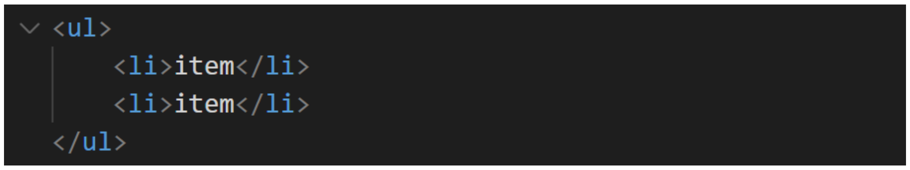
Use HTTPS for embedded resources where possible. Always use HTTPS for images, style sheets, and scripts except for when not available over HTTPS.



```
<!-- Not recommended -->
<script src="//funnything.com/funny.js"></script>

<!-- Recommended -->
<script src="https://funnything.com/funny.js"></script>
```

Indentation:

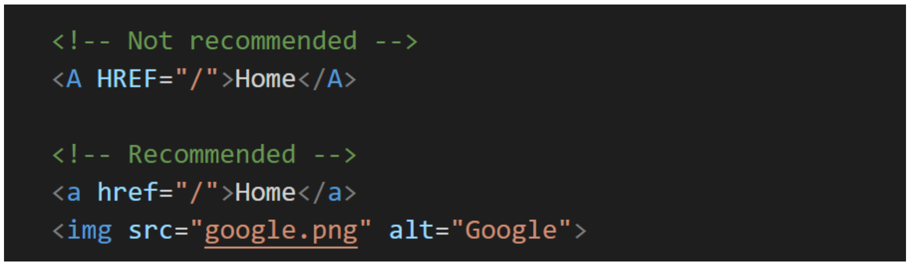
Indent using a single tab. Don't mix tabs and spaces for indentation.



```
<ul>
	<li>item</li>
	<li>item</li>
</ul>
```

Capitalization:

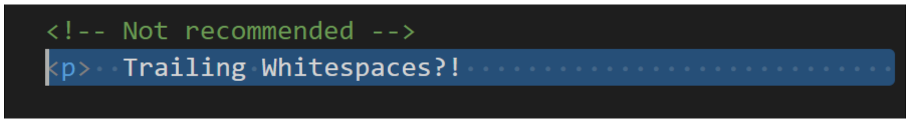
Use lowercase.
All HTML and CSS code should be lowercase including element names, attributes, attribute values, CSS selectors, properties, and property values except for strings, comments, and paragraph text.



```
<!-- Not recommended -->
<A HREF="/">Home</A>

<!-- Recommended -->
<a href="/">Home</a>
<img src="google.png" alt="Google">
```

Trailing Whitespace:

Remove any trailing white spaces.
Trailing white spaces are unnecessary and can complicate code.



```
<!-- Not recommended -->
<p>  Trailing Whitespaces?!
```

Encoding:

Use UTF-8.

Make sure your IDE editor is using UTF-8 as character encoding.

Specify the encoding in HTML templates and documents via

`<meta charset="utf-8">`

Comments:

Explain code as needed when possible.

Difficult sections of code should be documented explaining its function.

HTML Style Rules:

Document Type:

Use HTML 5

HTML syntax is to be used for all HTML documents: <!DOCTYPE html>.

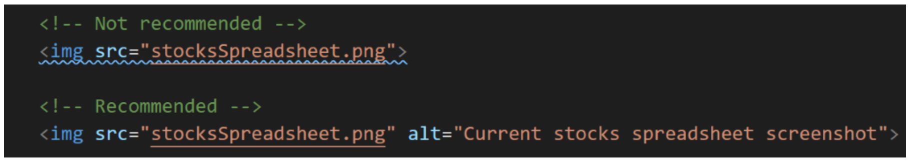
Semantics:

Use HTML elements for what they have been created for. For example, use heading elements for headings, p elements for paragraphs, and a elements for anchors.

Multimedia Fallback

Provide alternative contents for any multimedia.

Providing alternative contents is important for accessibility: A blind user using a reading application will have no cue without an alt text.



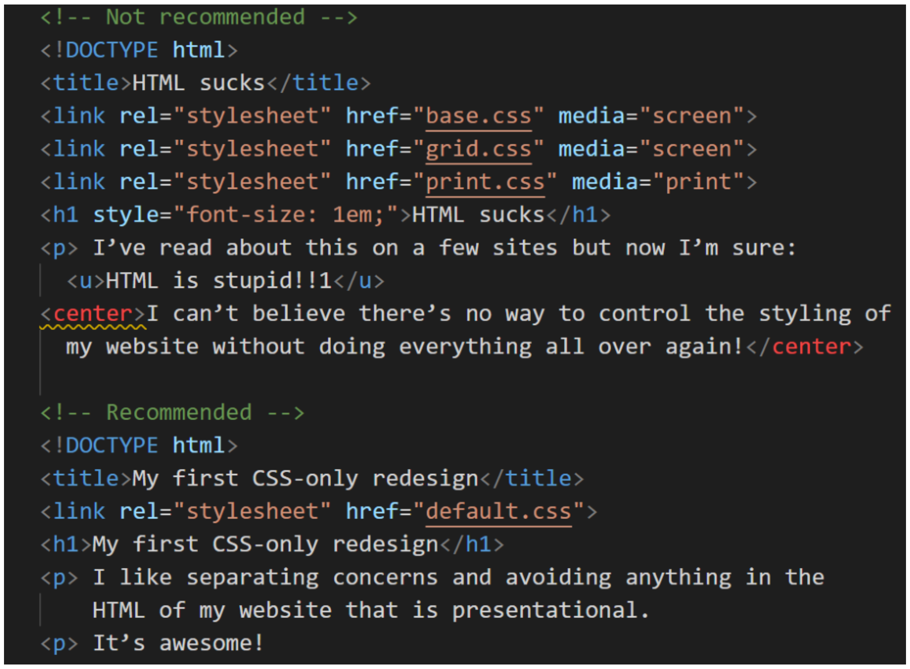
```
<!-- Not recommended -->
<img src="stocksSpreadsheet.png">

<!-- Recommended -->
<img src="stocksSpreadsheet.png" alt="Current stocks spreadsheet screenshot">
```

Separation of Code:

Strictly keep structure (markup), presentation (styling), and behavior (scripting) apart, and try to keep the interaction between them to an absolute minimum.

For example, make sure documents and templates contain only HTML that is solely serving structural purposes. Move everything used for presentation into style sheets, and everything behavioral into scripts.

In addition, keep contact as small as possible by linking as few style sheets and scripts to a single HTML document as possible.



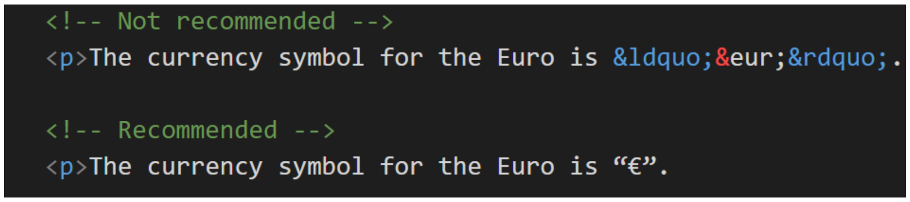
```html
<!-- Not recommended -->
<!DOCTYPE html>
<title>HTML sucks</title>
<link rel="stylesheet" href="base.css" media="screen">
<link rel="stylesheet" href="grid.css" media="screen">
<link rel="stylesheet" href="print.css" media="print">
<h1 style="font-size: 1em;">HTML sucks</h1>
<p> I’ve read about this on a few sites but now I’m sure:
  <u>HTML is stupid!!1</u>
<center>I can’t believe there’s no way to control the styling of
  my website without doing everything all over again!</center>

<!-- Recommended -->
<!DOCTYPE html>
<title>My first CSS-only redesign</title>
<link rel="stylesheet" href="default.css">
<h1>My first CSS-only redesign</h1>
<p> I like separating concerns and avoiding anything in the
    HTML of my website that is presentational.
<p> It’s awesome!
```

Entity References:

Do not use any entity references.

UTF-8 is used among all teams, files, and editors. There is no need to use entity references such as &mdash;, &rdquo;, or &eur.

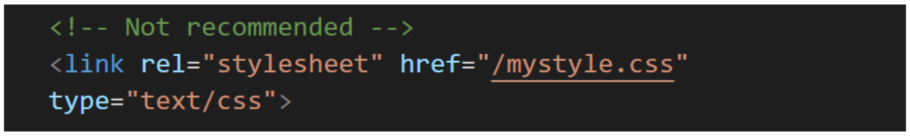
The only exceptions to this rule are characters with special meaning in HTML like < and & as well as control characters like no-break spaces.



```html
<!-- Not recommended -->
<p>The currency symbol for the Euro is &ldquo;&eur;&rdquo;.

<!-- Recommended -->
<p>The currency symbol for the Euro is “€”.
```

Type Attributes:

Do not include type attributes for style sheets and scripts as HTML5 implies them automatically.



```html
<!-- Not recommended -->
<link rel="stylesheet" href="/mystyle.css"
type="text/css">
```

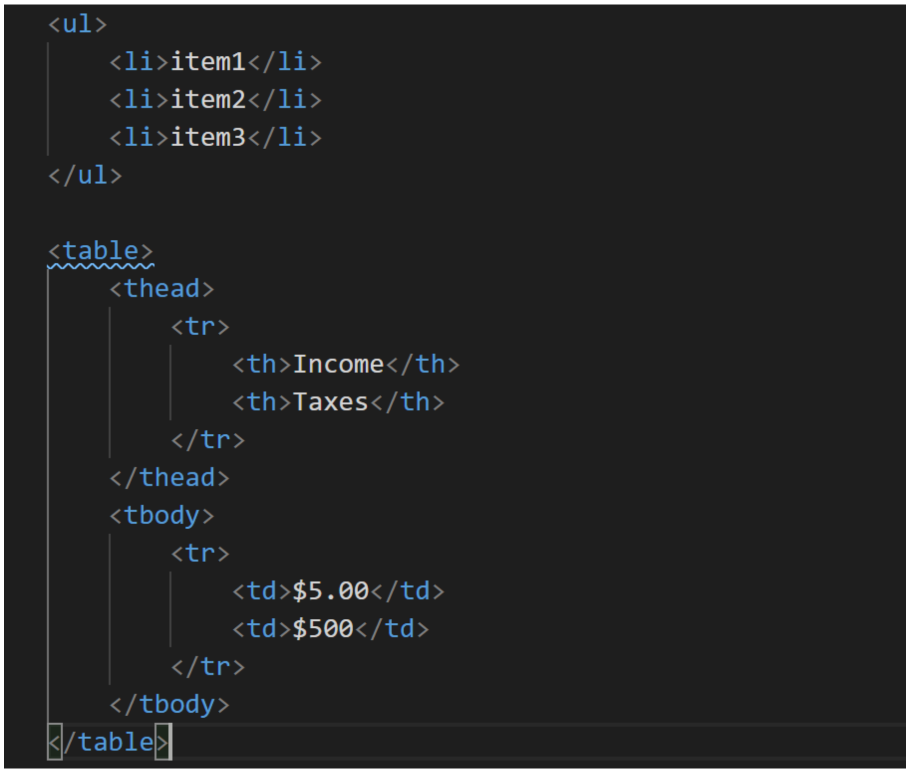
HTML Formatting Rules:

General Formatting:

Use a new line for every block, list, or table element, and indent every such child element.

Place every block, list, or table element on a new line.

Indent any element that is the child element of a block, list, or table element.



```html
<ul>
    <li>item1</li>
    <li>item2</li>
    <li>item3</li>
</ul>

<table>
    <thead>
        <tr>
            <th>Income</th>
            <th>Taxes</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>$5.00</td>
            <td>$500</td>
        </tr>
    </tbody>
</table>
```

HTML Line-Wrapping:

(optional) Break long lines.
When possible, consider wrapping long lines if it significantly improves readability.
When line-wrapping, each continuation line should be indented by 1 tab from the original line.

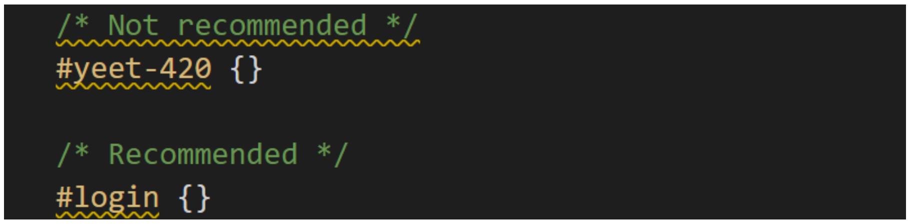
HTML Quotation Marks

When quoting attribute values, use double quotation marks (“”).

CSS Style Rules:

ID and Class Names:

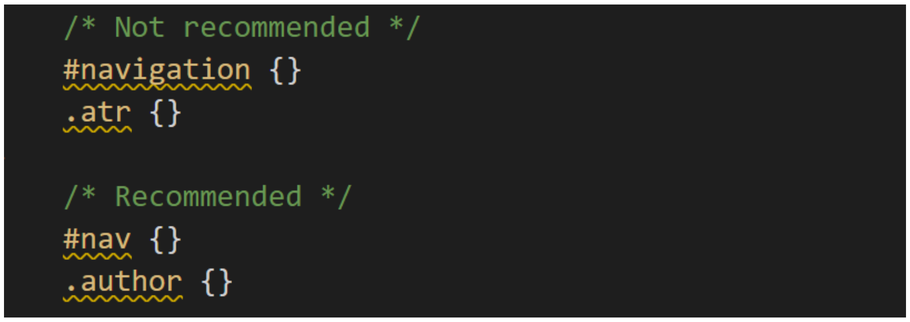
Use meaningful or generic ID and class names.
Don’t use cryptic names. Always use ID and class names that reflect the purpose of the element or ones that are generic.
Meaningful ID and class names are preferred. Generic names are a fallback for elements that have no meaning different from their siblings.



```css
/* Not recommended */
#yeet-420 {}

/* Recommended */
#login {}
```

ID and Class Names Length:

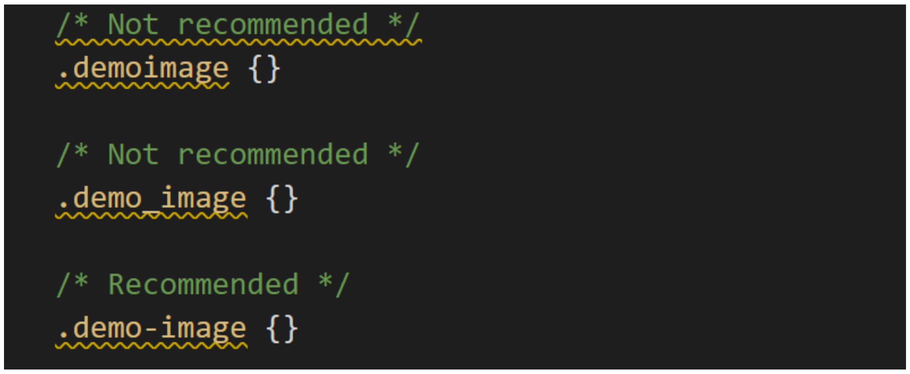
Use ID and class names that are as short as possible but as long as necessary.
Try to convey what an ID or class is while being as brief as possible.



```
/* Not recommended */
#navigation {}
.atr {}

/* Recommended */
#nav {}
.author {}
```

ID and Class Name Delimiters:

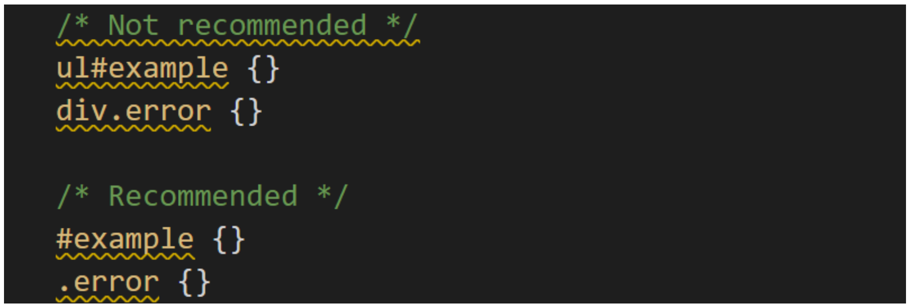
Separate multiple words in ID and class names by a hyphen.



```
/* Not recommended */
.demoimage {}

/* Not recommended */
.demo_image {}

/* Recommended */
.demo-image {}
```

Type Selectors:

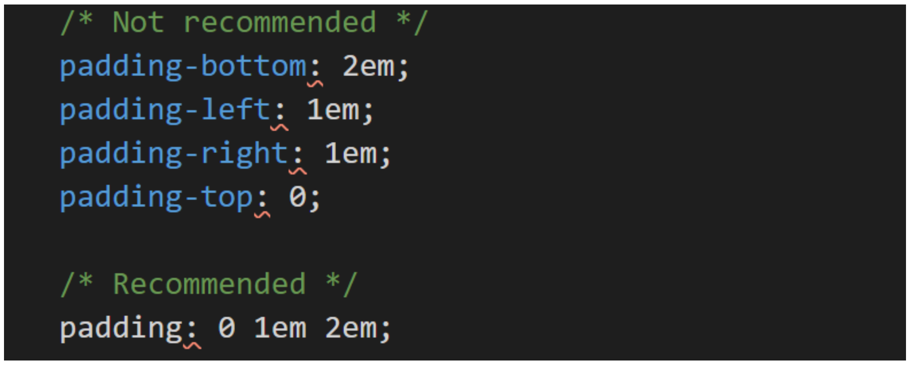
Avoid using ID and class names with type selectors.
Using unnecessary ancestor selectors hinders performance and load times.



```
/* Not recommended */
ul#example {}
div.error {}

/* Recommended */
#example {}
.error {}
```

Shorthand Properties:

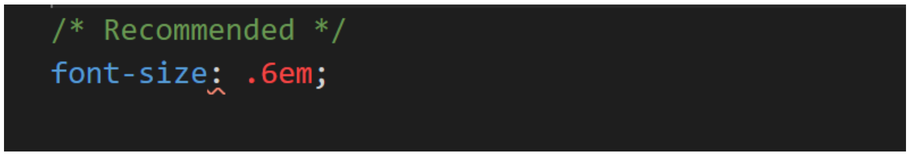
(optional) Use shorthand properties when possible.
Using shorthand properties is useful for code efficiency and understandability.



```
/* Not recommended */
padding-bottom: 2em;
padding-left: 1em;
padding-right: 1em;
padding-top: 0;

/* Recommended */
padding: 0 1em 2em;
```

Leading 0s:

Omit leading “0”s in values that are between -1 and 1.

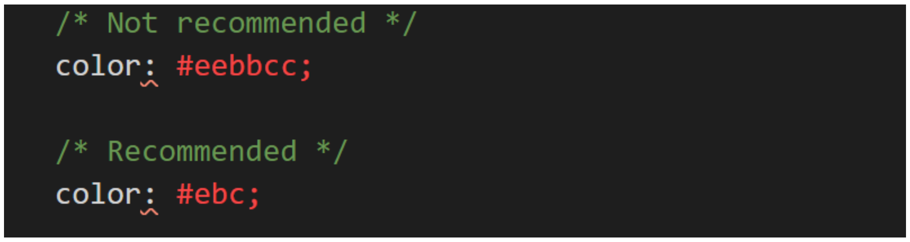


```
/* Recommended */
font-size: .6em;
```

Hexadecimal Notation:

Use 3 character hexadecimal notation where possible.

For the majority of color values, 3 character hexadecimal notation is shorter and more succinct.



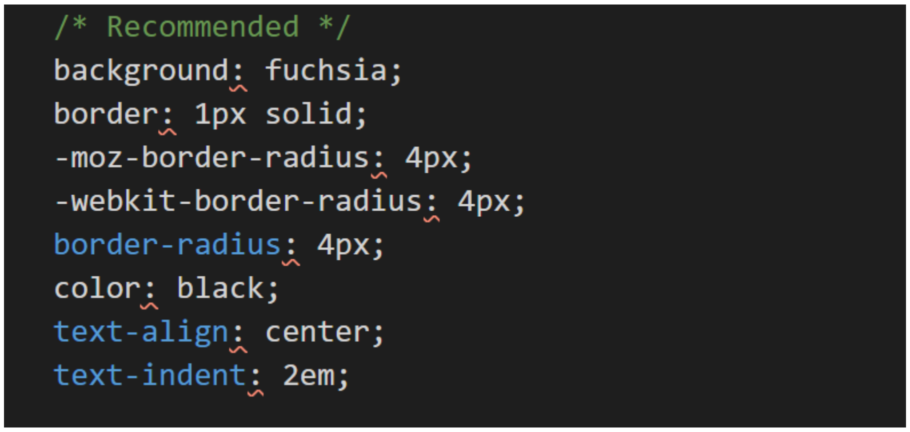
```
/* Not recommended */
color: #eebbcc;

/* Recommended */
color: #ebc;
```

CSS Format Rules:

Declaration Order:

Alphabetize declarations.

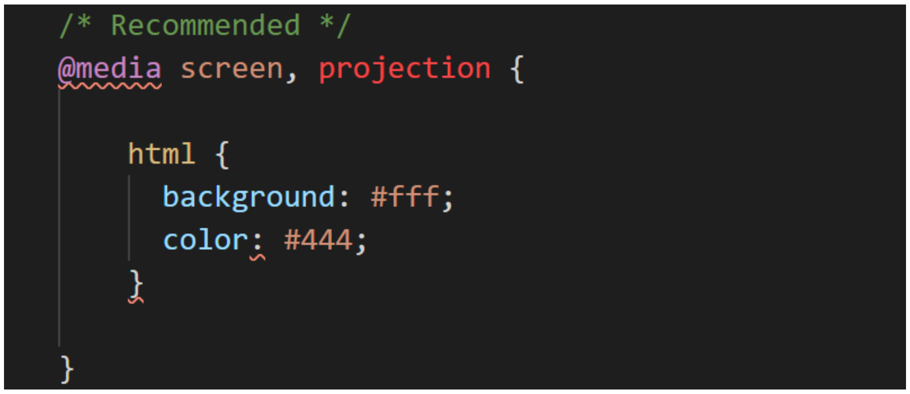
Put declarations in alphabetical order to achieve consistent code that is easy to remember and maintain.



```
/* Recommended */
background: fuchsia;
border: 1px solid;
-moz-border-radius: 4px;
-webkit-border-radius: 4px;
border-radius: 4px;
color: black;
text-align: center;
text-indent: 2em;
```

Block Indentation:

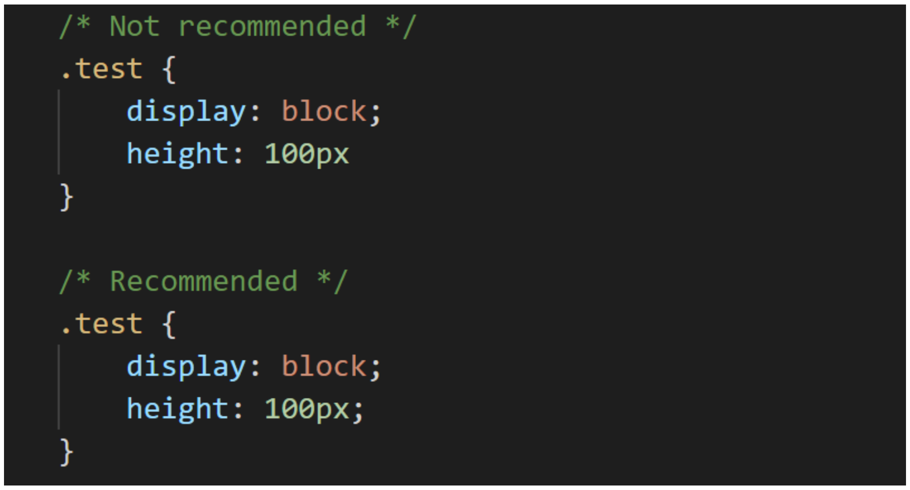
Indent all block content such as rules within rules as well as their declarations to properly reflect scope.



```
/* Recommended */
@media screen, projection {

    html {
      background: #fff;
      color: #444;
    }

}
```

Declaration Stops:

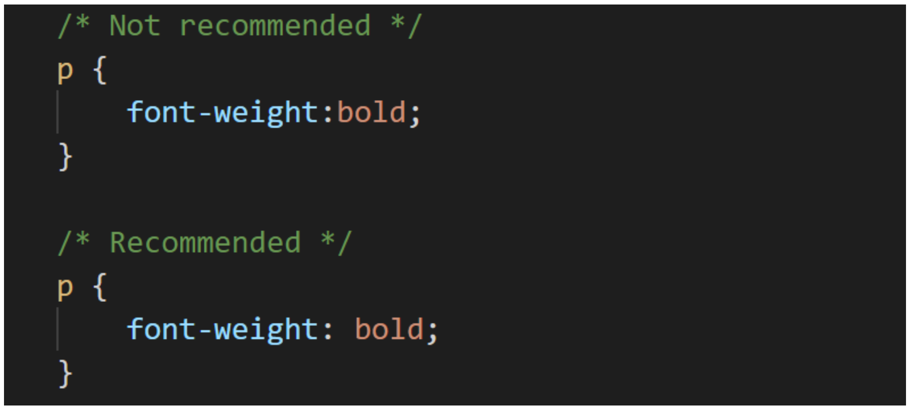
Use a semicolon after every declaration for consistency and clarity.



```
/* Not recommended */
.test {
    display: block;
    height: 100px
}

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

Property Name Stops:

Always use a single space after a property name's colon.

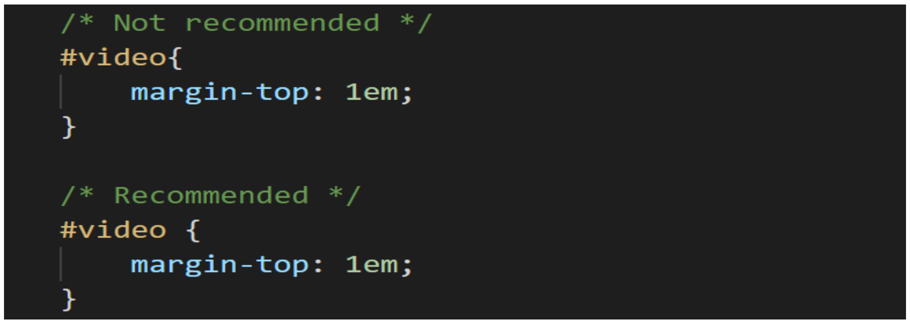


```
/* Not recommended */
p {
    font-weight:bold;
}

/* Recommended */
p {
    font-weight: bold;
}
```

Declaration Block Separation:

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

The opening brace should be on the same line as the last selector.

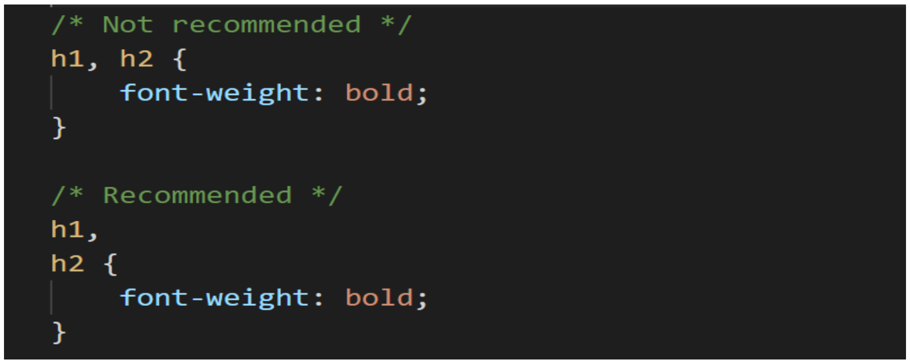


```css
/* Not recommended */
#video{
    margin-top: 1em;
}

/* Recommended */
#video {
    margin-top: 1em;
}
```

Selector and Declaration Separation:

Separate selectors and declarations by new lines.

Always start a new line for each new selector and declaration.

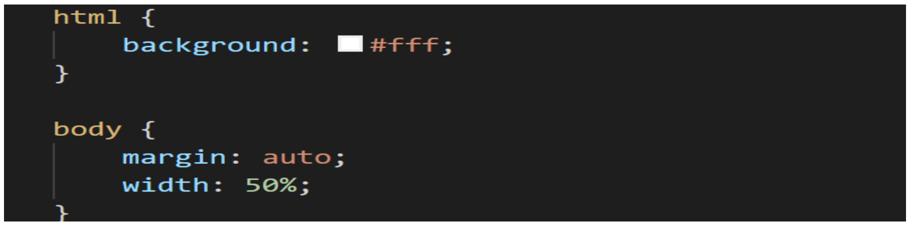


```css
/* Not recommended */
h1, h2 {
    font-weight: bold;
}

/* Recommended */
h1,
h2 {
    font-weight: bold;
}
```

Rule Separation:

Separate rules by new lines.

Always put a blank line between rules.

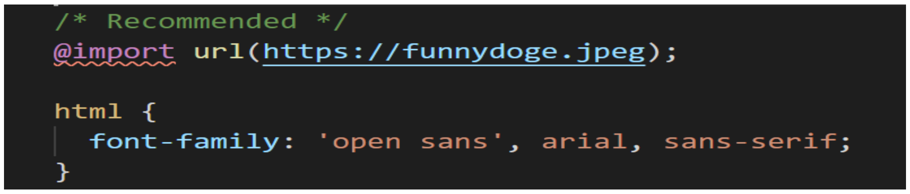


```css
html {
    background: #fff;
}

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

CSS Quotation Marks:

Use single (‘’) quotation marks for attribute selectors and property values.

Do not use quotation marks in URL values.



```css
/* Recommended */
@import url(https://funnydoge.jpeg);

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