

---

# **Coding Standards**

for

## **NINSHIKI**

Version 0.1.0 - Demo #2 Draft

---

A detailed set of guidelines dictating the best practices, programming styles and conventions that all developers of this project will adhere to. Please note that these guidelines were heavily influenced by the style guides released by Google<sup>1</sup> under the Creative Commons Licence<sup>2</sup>.

Compiled By

### **The Software Sharks Team**

Coetzer MJ

Bester TJ

Orrie O

Bekker L

Lew J

Matodzi MC

*Department of Computer Science  
The University of Pretoria*

For

**Bramhope International School of Inovation**

---

<sup>1</sup><http://google.github.io/styleguide/>

<sup>2</sup><https://creativecommons.org/licenses/by/3.0/>

COS 301 Team: Software Sharks

Department of Computer Science, University of Pretoria

<https://github.com/OrishaOrrie/SoftwareSharks>

This coding standards document was drafted under the supervision of involved lecturers according to the assessment guidelines of the final year Computer Science module: COS 301 - Software Engineering, presented by the Department of Computer Science in the faculty of Engineering, Built Environment and Information Technology at the University of Pretoria during the first semester of the year 2018.

**Disclaimer:** This document is a modification of the Google Style Guides licensed under the Creative Commons 3.0 Attribution - Allowing total reuse and modification and as such the Software Sharks, Department of Computer Science nor the University of Pretoria claim this entire document to be their original work.



First release, April 2018

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>HTML/CSS</b>	<b>4</b>
2.1	General . . . . .	4
2.1.1	General Style Rules . . . . .	4
2.1.2	General Formatting Rules . . . . .	4
2.1.3	General Meta Rules . . . . .	5
2.2	HTML . . . . .	6
2.2.1	HTML Style Rules . . . . .	6
2.2.2	HTML Formatting Rules . . . . .	7
2.3	CSS . . . . .	8
2.3.1	CSS Style Rules . . . . .	8
2.3.2	CSS Formatting Rules . . . . .	9
2.3.3	CSS Meta Rules . . . . .	10
<b>3</b>	<b>JavaScript</b>	<b>11</b>
3.1	Source File Basics . . . . .	11
3.1.1	File Name . . . . .	11
3.1.2	File Encoding . . . . .	11
3.1.3	Special Characters . . . . .	11
3.2	Source File Structure . . . . .	11
3.3	Formatting . . . . .	12
3.3.1	Braces . . . . .	12
3.3.2	Block indentation: +2 spaces . . . . .	13

## **1 Introduction**

## 2 HTML/CSS

### 2.1 General

#### 2.1.1 General Style Rules

##### Protocol

- Never omit the protocol when embedding a resource, image or script.
- Always make use of the HTTPS protocol for embedding resources, images or scripts when available.

```
<!-- Recommended -->  
<script src="https://example.com/example/someJava.min.js"></script>
```

#### 2.1.2 General Formatting Rules

##### Indentation

- Indent with two spaces.
- Do not make use of tabs or various spacing other than the above for indentation.

```
<!-- Recommended -->  
<ul>  
  <li>Item 1  
  <li>Random Item 2  
</ul>
```

##### Capitalisation

- All code has to be in lower-case with the exception of strings.

```
<!-- Recommended -->  

```

##### Trailing White Space

- Ensure that there are no trailing white spaces in any line.

### 2.1.3 General Meta Rules

#### Encoding

- Make use of UTF-8 encoding that does not include a byte order mark.
- Specify the encoding in HTML templates and documents via

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

- Do not specify the encoding of style sheets as these assume UTF-8.

#### Comments

- Where ever possible explain code as needed.
- Use comments to explain code: What does it cover, what purpose does it serve, why is respective solution used or preferred?

#### Action Items

- Mark todos and action items with **TODO**.
- Highlight todos by using the keyword **TODO** only, not other common formats like @@.
- Append a contact (username or mailing list) in parentheses as with the format **TODO**(contact).
- Append action items after a colon as in **TODO:** action item.

```
<!-- TODO: remove optional tags -->
<ul>
  <li>Tag 1</li>
  <li>Tag 2 (Optional)</li>
</ul>
```

## 2.2 HTML

### 2.2.1 HTML Style Rules

#### Document Type

- Use HTML5.
- HTML5 (HTML syntax) is preferred for all HTML documents: `<!DOCTYPE html>`.
- Do not close void elements, i.e. write `<br>`, not `<br />`.

#### HTML Validity

- Use valid HTML where possible according to the W3C<sup>3</sup>.
- Use valid HTML code unless that is not possible due to otherwise unattainable performance goals regarding file size.

#### Semantics

- Use elements (sometimes incorrectly called “tags”) for what they have been created for. For example, use heading elements for headings, `<p>` elements for paragraphs, `<a>` elements for anchors, etc.

#### Multimedia Fall-back

- For multimedia (such as images, videos and animated objects via `<canvas>`) make sure to provide meaningful alternative text via `alt="Descriptive alt text"` to enable global accessibility.

```
<!-- Recommended -->  

```

#### Separation of Concerns

- Separate structure (HTML), presentation (CSS) and behaviour (Scripts) into their respective files and file-types.

#### Entity References

- Do not make use of entity references except for special HTML characters (`<` and `&`).

---

<sup>3</sup><https://validator.w3.org/nu/>

### type **Attributes**

- Do not use type attributes for style sheets (unless not using CSS) and scripts (unless not using JavaScript).

```
<!-- Recommended -->  
<link rel="stylesheet" href="https://www.google.com/css/maia.css">  
<script src="https://www.google.com/js/gweb/analytics/autotrack.js"></script>
```

## 2.2.2 HTML Formatting Rules

### General Formatting

- Use a new line for every block, list or table element.
- Indent every such child element.

### HTML Line-Wrapping

- It is optional to break long lines in order to improve readability.
- If lines are broken then the continuation line should be indented at least 4 additional spaces from the original line.

### HTML Quotation Marks

- When quoting attribute values, make use of double quotation marks instead of single quotation marks.

```
<!-- Recommended -->  
<a class="maia button-secondary">Sign in</a>
```



## 2.3 CSS

### 2.3.1 CSS Style Rules

#### CSS Validity

- Use valid CSS where possible according to the W3C<sup>4</sup>.

#### ID and Class Naming

- Use meaningful or generic ID and class names that clearly reflect the purpose of the element instead of presentational or cryptic names.

```
/* Recommended: specific */
#gallery {}
#login {}
.video {}

/* Recommended: generic */
.aux {}
.alt {}
```

#### ID and Class Name Style

- Use ID and class names that are as short as possible but as long as necessary.

#### Type Selectors

- Avoid qualifying ID and class names with type selectors.

#### Shorthand Properties

- Use shorthand properties where possible for code efficiency and understandability.

#### Leading Os

- Omit leading "0"s in values or lengths between -1 and 1.

```
/* Recommended */
font-size: .8em;
```

---

<sup>4</sup><https://jigsaw.w3.org/css-validator/>

### Hexadecimal Notation

- Use 3 character hexadecimal notation where possible.

```
/* Recommended */  
color: #ebc; /* Instead of #eebbcc */
```

### ID and Class Name Delimiters

- Separate words in ID and class names by a hyphen only.

```
/* Recommended */  
#video-id {}  
.ads-sample {}
```

## 2.3.2 CSS Formatting Rules

### Declaration Order

- Alphabetize declarations in order to achieve consistent code in a way that is easy to remember and maintain.
- Ignore vendor-specific prefixes.

### Block Content Indentation

- Indent all block content so to reflect hierarchy and improve understanding.

### Declaration Stops

- End every declaration with a semicolon for consistency and extensibility reasons.

### Property Name Stops

- Use a space after a property name's colon but not before.

### Declaration Block Separation

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

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

## Selector and Declaration Separation

- Separate selectors and declarations by new lines.

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

## Rule Separation

- Separate rules by a blank line.

## CSS Quotation Marks

- Use single ( ' ' ) rather than double ( " " ) quotation marks for attribute selectors and property values.
- Do not use quotation marks in URI values (`url()`).

```
/* Recommended */
@import url(https://www.google.com/css/maia.css);

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

### 2.3.3 CSS Meta Rules

#### Section Comments

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

```
/* Recommended */
/* Header */

#adw-header {}

/* Footer */

#adw-footer {}
```

## 3 JavaScript

### 3.1 Source File Basics

#### 3.1.1 File Name

- File names must be all lower-case and may include underscores (\_) or dashes (-), but no additional punctuation.
- Follow the convention that your project uses.
- File name extensions must be '.js'.

#### 3.1.2 File Encoding

- All source files must be encoded in UTF-8.

#### 3.1.3 Special Characters

##### Whitespace Characters

- Aside from the line terminator sequence, the ASCII horizontal space character (0x20) is the only whitespace character that appears anywhere in a source file.
- All other whitespace characters in string literals are escaped.
- Tab characters are not used for indentation.

##### Special escape sequences

- Use special escape sequences: " \', \", \\, \b, \f, \n, \r, \t, \v ".
- Do not use numeric or octal escapes.

### 3.2 Source File Structure

A source file consists of, in order:

1. Licence or copyright information if available.
2. `// @fileoverview` JSDoc.
3. File implementation.

Exactly one blank line separates each section that is present.

### 3.3 Formatting

#### 3.3.1 Braces

##### Braces are Used for all Control Structures

- Braces are required for all control structures.
- The first statement of a non-empty block must begin on its own line.

##### Non-empty blocks: K&R style

Braces follow the Kernighan and Ritchie style ("Egyptian brackets") for non-empty blocks and block-like constructs:

- 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 if that brace terminates a statement or the body of a function or class statement, or a class method.

```
class InnerClass {  
    constructor() {}  
  
    /** @param {number} foo */  
    method(foo) {  
        if (condition(foo)) {  
            try {  
                // Note: this might fail.  
                something();  
            } catch (err) {  
                recover();  
            }  
        }  
    }  
}
```

##### Empty blocks: may be concise

- An empty block or block-like construct may be closed immediately after it is opened, with no characters, space, or line break in between.

**3.3.2 Block indentation: +2 spaces****Array literals: optionally "block-like"**