Style Guide

A styling guide of rules to keep in mind when styling a web page.

We'll cover the following

Style guide {`#style-guide`}

Here are the coding rules and principles used throughout the course:

> This chapter is by nature subjective and opinionated. Feel free to make your own choices.

Naming

Naming things right goes a long way into making code cleaner and easier to understand. Some general naming rules are presented below.

1. Choose Meaningful Names

The most important rule is to give each element (variable, function, class, etc) a specific name that reflects its role. A variable holding the value of a circle radius should be named `radius` rather than `num` or `myVal`. Brevity should be

limited to short-lived elements, like loop counters.

2. Don't Use Reserved Words

Each JavaScript keyword is a reserved name. They should not be used as variable names. Here's the list of reserved words in JavaScript.

3. Follow a Naming Convention

It can take several words to describe precisely the role of certain elements. This course adopts the popular camelCase naming convention, based on two main principles:

- All names begin with a *lowercase* letter.
- If a name consists of several words, the first letter of each word (except the first word) is *uppercase*.

In addition, this course uses the following naming rules:

- Functions and method names include an *action verb*: `computeTotal()`, `findFirstParent()`, `attackTarget()`, etc.
- To be consistent with other programming languages, class names start with an *uppercase* letter: `User` instead of `user`.
- Since they may contain multiple elements, arrays are named *plurally* or suffixed with `List`: `movies` or `movieList`, but not `movie`.
- To distinguish them from other variables, DOM elements are suffixed with `Element` (or `Elements` for array-like variables): `divElement` rather than simply `div`.

> Like many other languages, JavaScript is *case sensitive*. For example, `myVariable` and `myvariable` are two different variable names. Be careful!

Code Formatting

This is a subject of many debates in the JavaScript community: using spaces or

This is a subject of many debates in the JavaScript community: using spaces or tabulations for indenting, omitting semicolons, simple vs double quotes for strings, and so on.

A simple and efficient solution is to rely on a tool to automate the low-level task of formatting code, so that you can concentrate on more high-level work.

Code Quality

Since JavaScript is a dynamically typed language, a number of errors don't show up until execution: misnaming a function, loading a nonexistent module, etc. In addition, many other mistakes like declaring a variable without ever using it won't affect execution outcome, but make your code harder to read and lower its overall quality.

Fortunately, specialized tools called linters can check your code against rules during edition and warn about potential defects. By allowing to fix many bugs before they happen, linters greatly enhance developer productivity.