Experimentation and Evaluation **2024**

Students: Davide Frova, Costanza Rodriguez Gavazzi

Project 2 **Due date:** Thursday, 19 December 2024, 06:00 PM

---

1. Abstract

This study investigates whether using `camelCase` or `kebab-case` as a naming convention affects response time when reading and understanding code identifiers. A total of 36 participants, with and without programming backgrounds, completed timed tasks involving both case styles. Descriptive statistics showed that participants were faster and more consistent with `camelCase`, especially those without programming experience. Inferential statistics using a paired samples t-test revealed no significant difference between the two styles, suggesting that while `camelCase` appeared slightly faster, the difference was not statistically significant.

2. Introduction

The readability of code is extremely important for developer productivity and collaboration. Part of it is the style of naming conventions used for identifiers. Two commonly used styles are `camelCase` (e.g., `myVariableName`) and `kebab-case` (e.g., `my-variable-name`). While every developer has their preference (including us), we wanted to investigate whether one style is more effective than the other. In this study, we conducted a controlled experiment to compare response times for tasks involving `camelCase` and `kebab-case`. We also comment on descriptive statistics with respect to accuracy, measured as correctness of the answers. Both participants with and without programming backgrounds were involved.

1. Hypotheses

Null Hypothesis: There is no significant difference in speed between `camelCase` and `kebab-case`.

Alternative Hypothesis: Users perform better with `kebab-case` compared to `camelCase`.

3. Method

1. Variables

1.1. Independent Variable

Case Style: `camelCase`, `kebab-case`

1.2. Dependent Variables

Response Time: in milliseconds;
Correctness: true/false

1.3. Control Variables

Questions: the **same** questions were presented to the participants, although the order of presentation was randomly determined to avoid any possible learning effects.

1.4. Blocking Variable

Previous programming experience: true/false.

2. Design

Single-Factor Experiment

Within-Subject Design: Participants experienced both case styles in random order to avoid potential learning effects.

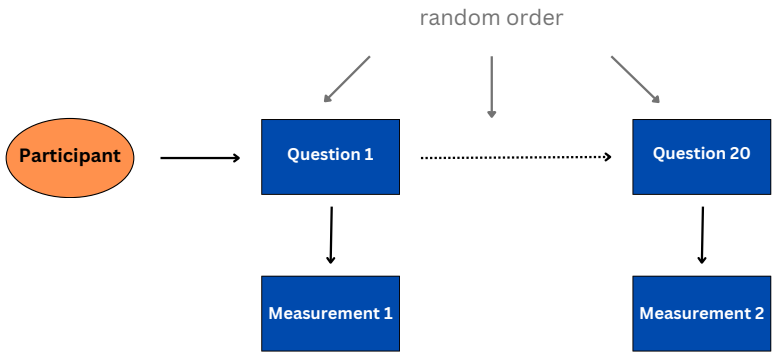

Figure 1: Design of the experiment

3. Participants

Total participant: 36
Age range: 20 - 58
Programming background: 25 participants had programming experience, 11 did not.

4. Apparatus and Materials

Web App: A custom web application was used to present the questions and record response times. The app was designed to be simple and intuitive, with a clean interface to minimize distractions. It was developed using the framework Next.js, which provided an easy and fast way to create a web application with React that met our requirements. We also took care of the responsiveness of the app, so that it could be used on both desktop and mobile devices. This was possible thanks to the styling and layout system provided by Material UI.

The app was hosted on the service Vercel, which allowed us to easily deploy and share the experiment with participants without any expenses.
We think that deploying the app on Vercel was a good choice, as it allowed us to easily share the URL to the experiment with anyone just by sending a link.
The link is the following: `https://code-comprehension.vercel.app`

Data Persistence: The app stored the data in a MongoDB database, which allowed us to easily export the data in CSV and JSON format for further analysis. We used the free tier of MongoDB Atlas, which was sufficient for our needs and easy to configure with Next.js and Vercel.

Here is an example of the interface of the web app while asking the user a question:

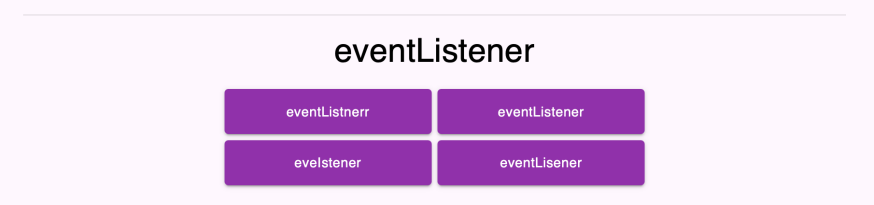

Figure 2: Web App Interface

5. Procedure

After reading a brief explanation of the experiment participants were asked to:

- Provided their age and whether they had programming background.
- Completed a warm-up tutorial.
- Answer 10 `camelCase` and 10 `kebab-case` questions, randomly shuffled.

We recorded response times, correctness, age of participant and whether they had a programming background or not.

4. Results

These results are produced using a Jupyter Notebook with Python and JASP.

1. Visual Overview

Looking at how people performed with `camelCase` in Figure 3 and Figure 4 and `kebab-case` in Figure 5 and Figure 6, we can see that both styles were correct at around 90%.

In particular about 91.4% of answers were correct in the group of people who had programming experience and 94% in the group of people who did not have any experience. We hypothesize that the difference in accuracy between the two groups is due to the fact that people with programming experience were quicker to answer - which is later discussed -, and therefore more prone to make mistakes.

Figure 7 shows the boxplot of the elapsed time for `camelCase` and `kebab-case`. We can see that the median response time (the center line in each box) for `camelCase` is slightly lower, meaning participants answered faster with this naming style. Furthermore, `kebab-case` has a wider spread of response times, as we can notice in the number of outliers (dots) above the box.
The distributions in Figure 8 and Figure 9 exhibit a right-skewed shape, where most correct answers occur at a shorter elapsed time. In Figure 8, the `camelCase` distribution peaks slightly

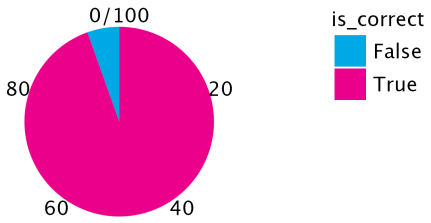

Figure 3: Correct Answers for `camelCase` with No Programming Background

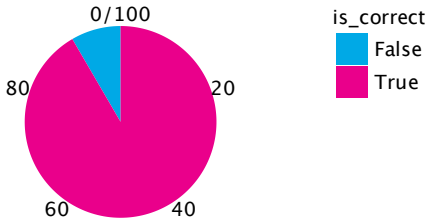

Figure 4: Correct Answers for `camelCase` with Programming Background

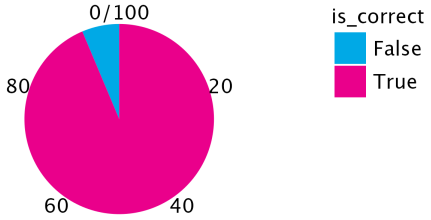

Figure 5: Correct Answers for `kebab-case` with No Programming Background

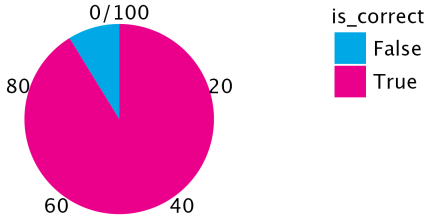

Figure 6: Correct Answers for `kebab-case` with Programming Background

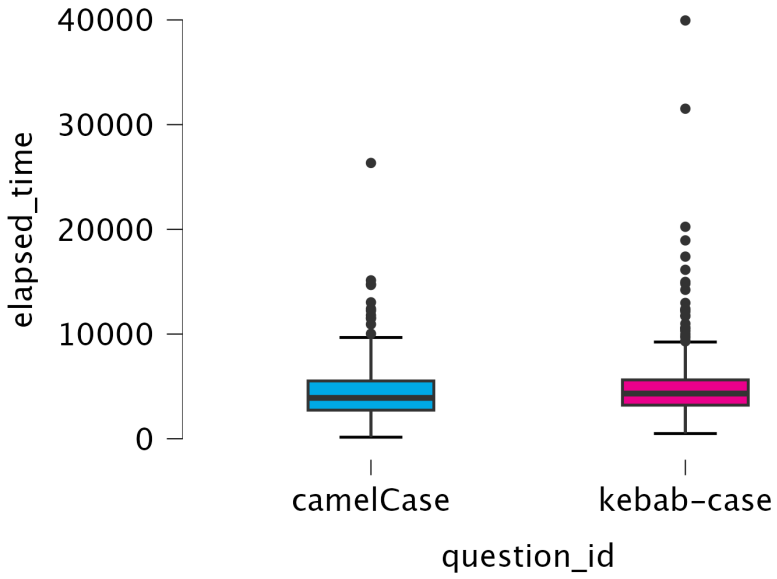

Figure 7: `camelCase` vs `kebab-case`: Elapsed Time

earlier, meaning faster response times. Figure 9 shows a longer tail, suggesting that some participants took more time to answer.

The data shows that `camelCase` was slightly faster, while the `kebab-case` times were a bit more

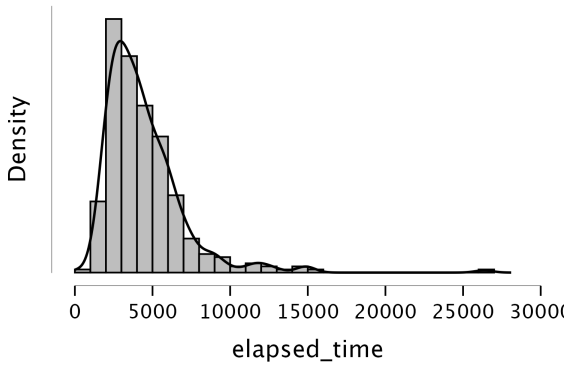

Figure 8: Distribution of Correct Answers for `camelCase`

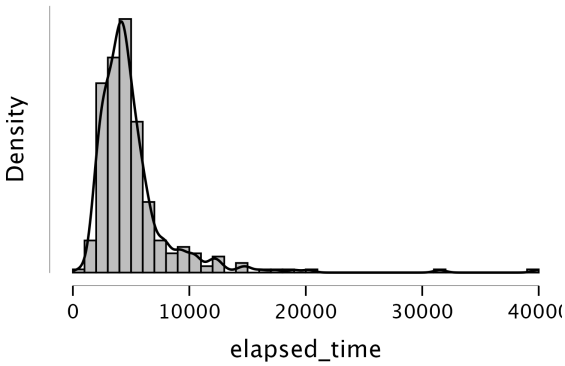

Figure 9: Distribution of Correct Answers for `kebab-case`

spread out, but followed the same basic shape. The long tails may indicate that some identifiers were harder to understand, or that some participants were more familiar with one case style over the other.

These observations suggests that `camelCase` led to faster responses on average and more consistent performance. In contrast, `kebab-case` resulted in more variation, with some participants taking much longer to respond.

Overall, while both naming conventions were readable, `camelCase` seems to be on average faster for participants.

2. Descriptive Statistics

Table 1: Descriptive Statistics of All Participants

	Mean	Std. Deviation	Minimum	Maximum	25th percentile	50th percentile	75th percentile
`camelCase`	4442.718	2609.916	150.000	26341.000	2734.000	3904.000	5522.000
`kebab-case`	5119.937	3691.977	497.000	39951.000	3209.500	4319.000	5624.000

Table 1, which includes all participants, shows that the average response time for `camelCase` is 4442 ms, compared to 5119 ms for `kebab-case`. The smaller standard deviation for `camelCase` (2609 ms) suggests that responses were more consistent.

Table 2: Descriptive Statistics of Participants with a Programming Background

		Mean	Std. Deviation	Minimum	Maximum	25th percentile	50th percentile	75th percentile
elapsed_time	`camelCase`	3969.640	2113.221	150.000	15059.000	2548.250	3511.000	4987.750
	`kebab-case`	4567.916	3290.778	328.000	39951.000	3007.250	4007.500	5118.750

In Table 2, participants with programming experience appear to respond faster on average. For `camelCase`, the mean response time is 3969 ms, while `kebab-case` shows a higher mean of 4567 ms. This may suggest that `camelCase` could be slightly easier or quicker to process for programmers. The smaller standard deviation for `camelCase` (2113 ms) compared to `kebab-case` (3290 ms) indicates more consistent response times, in line with the trends for all participants.

Table 3: Descriptive Statistics of Participants without a Programming Background

		Mean	Std. Deviation	Minimum	Maximum	25th percentile	50th percentile	75th percentile
elapsed_time	`camelCase`	5296.818	3376.998	1744.000	26341.000	3200.250	4419.500	6334.250
	`kebab-case`	6310.291	4330.106	1771.000	31522.000	3637.000	4952.500	7584.250

In Table 3, participants without programming experience seem to take longer overall. The average response time for `camelCase` is 5296 ms, while for `kebab-case` it is 6310 ms. This could mean that `camelCase` might be processed more quickly by non-programmers as well.

Overall this indicates that `camelCase` seems to be faster than `kebab-case`.

3. Inferential Statistics

Table 4: Paired Samples T-Test

Measure 1		Measure 2	t	df	p	Cohen's d	SE Cohen's d
`kebab-case`	-	`camelCase`	3.515	359	1.000	0.185	0.064

Note. For all tests, the alternative hypothesis specifies that `kebab-case` is less than `camelCase`.

Table 4 shows the results of a paired samples t-test. We compares response times for `kebab-case` and `camelCase`. The test revealed that the observed difference in response times was not statistically significant ($t(359) = 3.515$, $p = 1.000$). The small Cohen's d value of 0.185 tells us that even when people were faster with `camelCase`, the difference is not statistcally relevant.

Even though the descriptive statistics analysis showed that people were generally faster and more consistent with `camelCase`, these inferential statistic results suggest that this effect is not

significant. This outcome supports the null hypothesis that there is no significant difference in reading speed between the two case styles.

5. Discussion

1. Comparison of Hypotheses with Results

The results support the null hypothesis: there was no significant difference between `kebab-case` and `camelCase` in terms of response times. Although `camelCase` appeared slightly faster in descriptive statistics, this difference was not statistically significant.

For participants with programming experience, the mean response time for `camelCase` was about 600 ms faster than for `kebab-case`. Non-programmers showed a larger descriptive difference of around 1000 ms, with `camelCase` being faster. However, the inferential statistics suggest these differences could be due to random variation, as the effect size was small. Furthermore, we observed that non programmers were more accurate than programmers, which could be due to the fact that they took more time to answer.

2. Implications

While `camelCase` showed a slight advantage in speed and consistency, the lack of statistical significance means that neither style can be declared superior for reading comprehension tasks. This is consistent with the idea that naming conventions are mainly a matter of preference, familiarity or other factors.

3. Limitations and Threats to Validity

This study has several limitations:

- The sample size of 36 participants may have been too small to detect subtle differences.
- The participants were mainly Informatics students, meaning that the sample was not representative of the general population of programmers.
- Variations in participants' familiarity and preferance with naming conventions could have impacted response times.
- The experiment doesn't really replicate real-world coding environments, where identifier complexity may differ.
- The study does not consider the impact of the participants' native language.

4. Conclusion

The findings indicate that both `camelCase` and `kebab-case` are highly readable, with no significant performance difference in terms of speed or accuracy. While `camelCase` appeared slightly faster and more consistent, this effect was small and not statistically significant. Future research with larger sample sizes and more complex tasks could provide further insights into how naming conventions influence code readability and comprehension.

6. Appendix

1. Reproduction Package

All the code and data used in this study is available on GitHub at:
`https://github.com/costanza1234/USI-Exp-Eval-24/tree/main/experiment2`
The github repository contains the following elements:

- `report/`: This report in LaTeX format.
- `results/`: Contains the raw and processed data from the experiment and the analysis.ipynb file used to analyze the data. In particular, the `results/code-comprehension.runs.csv` and `results/code-comprehension.runs.json` files contain the raw data exported from the MongoDB database, they are provided in both CSV and JSON format.
  The `results/questions.json` file contains the questions used in the experiment, they are divided in two categories: `camelCase` and `kebab-case`. Foreach question we have the `identifier` and the `distractors`.
- `code-comprehension/`: Is a git submodule that contains the code for the web app used in the experiment.

The link to the web app is the following: `https://code-comprehension.vercel.app`