camelCase vs kebab-case

Group 5

May 16, 2022

Erick Garro Elizondo
Cindy Guerrero Toro

Contents

Abstract

Introduction: A proper identifier is a set of words that can provide several layers of useful information into a programming language. The current research was carried out to investigate the correlation between multiple word identifiers and the facility for people to recognize and comprehend those.
Methods: The main hypothesis considers that the case styles, kebab-case or camelCase, influences the response correctness and speed. The readability skills of 69 participants were considered for either style. A deeper analysis was based on 1) academic background (High-School, HS; Computer Science field, CS; Other Fields, OF) and 2) facilitated readability by color case (monochromatic or chromatic). The participants were instructed to use the web application *EExp2*, where they completed a tutorial and then a twenty-task experiment. A web application was chosen over a desktop application to make it easier for users to utilize the tool and minimize error and for researchers to collect results.
Results: A general overview shows no difference in the correctness between kebab-case or camelCase. Participants with HS education showed a lower responding speed than those within CS for kebab-case. The speed responses to camelCase monochromatic were faster than its chromatic pair. CS participants had a better time response to camelCase than kebab-case when monochromatic.
Conclusions: The correctness was not affected by either case style, whereas the speed showed to be faster for camelCase, whereas the response time of camelCase is better for trained individuals in programming.

Acknowledgements

The beneficiaries have confirmed that they have taken all necessary precautions to avoid potential conflicts of interest that could affect the impartial and objective conduct of the current research. Such a conflict of interest may arise for a variety of reasons, including economic interests, political or national affiliation, family or emotional ties, or other common interests. Any circumstance that caused or could cause a conflict of interest during the conduct of the project was disclosed in writing in this report and communicated to the responsible parties. The authors confirmed that there was no economic or sociopolitical conflict of interest that would have affected the development of this project. The authors initially recruited participants by approaching friends, relatives, colleagues, or close acquaintances who may have recruited new volunteers through social media. Volunteers received no economic or social compensation. The authors collected a minimal amount of information (age, academic background, and gender) that was treated as private via a token ID. Neither the authors nor other sources could identify an individual if they had a set of responses or the token ID. The authors allowed the option 'n/a' if an individual did not wish to provide personal information. These data were excluded for analysis due to the small number of participants. The authors sincerely thank all volunteers who took the time to participate in this study and invite new participants.

Erick Garro & Cindy Guerrero

Language is a process of free creation: its laws and principles are fixed, but the manner in which the principles of generation are used is free and infinitely varied. Even the interpretation and use of words involves a process of free creation.

Noam Chomsky

1 Introduction

An identifier is a word or group of words chosen by programmers as entity names that provide useful information [4]. These identifiers are important for high-level understanding of a program. A correct identifier in programs can not only be descriptive, but also improves the understandability and maintainability of programs [12]. The main characteristic of a correct identifier is that it gives a meaning to the program. To achieve this, according to Alexander (2018) [1], there are three properties that an identifier must always contain: 1) Demonstrates an understanding of the problem that the program is intended to solve. 2) Uses words that everyone can understand. 3) Be unambiguous in your code. The *first characteristic* introduces the concept of self-explanatory identifiers. This refers to the ability to choose an identifier that can accurately describe the task to be performed. The *second characteristic* points to readability. This indicates the ability to empathize with others and avoid ubiquitous language. In this case, users (developers, designers, or customers) can avoid getting lost in translation by using common language terms. A clear example of this is acronyms or abbreviations whose meaning is not clear to everyone. The *third characteristic* concerns a certain organization of the code. Not all programmers are able to describe a process eloquently, certainly not when a hierarchical structure is involved (i.e., class, subclass, methods, etc.). The size of a function or method also affects your ability to name things. By limiting its size to a small value, you force the function to perform a single task. This level of abstraction makes it easier to come up with a name.

To better understand the meaning of an identifier, consider a program with two given functions `save_category(string category)` and `foo(string cat)`. Each of them contains exactly the same body. The body contains the specifications for storing the information in the database that corresponds to a category (a well-defined subdivision in a categorization system). It can be seen that the 1 example contains more information than the 2 example. In the first case, there is an identifier `save_category`, from which the user can already conclude that it is a function to *"save a category"* is involved. This function receives a `String category` as a parameter. So when `save_category` is executed, the specified category will be saved to the database.

Example 1 (Proper identifier)**.**

```
public void save_category (String
    category) {
    // The {body} of the function
       that saves category into
       database
    // ...
}
```

Unlike `save_category( )`, the method `foo( )` has an ambiguous identifier name, `foo`. One cannot tell at first glance what `foo` can do. Also, `foo` has a parameter `String cat`, which refers to the category. However, `cat` is also an ambiguous term, as it could refer to either `cat`, the animal, `cat`, the catalogue, `cat`, the category, `cat`, the cathering, or other `cat`. Thus, the user cannot directly determine what `foo( )` does without first fully understanding the body or even the entire programme.

Example 2 (Improper identifier)**.**

```
public void foo (String cat) {
    // The {body} of the function
       that saves category into
       database
    // ...
}
```

As mentioned earlier, an identifier should not be presented with a ubiquitous language, but as part of a natural language (NL). A natural language is any language that arose spontaneously and evolved in humans through the constant use and repetition of sounds or gestures without intentional design [11].

There are two main forms of NL, spoken language and sign language. These two forms play a role in the development and evolution of writing, an expressive language characterized by a system of markings. Following the era of proto-writing (*c.* 6,000 BCE), which gave rise to basic ideographic symbols, one of the oldest and best-known forms of established writing is cuneiform (*c.* 3,200 BCE). Cuneiform was a logosyllabic representation used to represent goods. Around 2600 BC, cuneiform began to represent syllables of the Sumerian language, a language spoken in Mesopotamia (attested from *c.*3,500 BC) [14]. During the development of writing three main forms of writing have emerged: 1) *image writing system*, with simplified images representing objects or concepts (Mnemonic, glyphs used as memory aids; Pictographic, glyphs directly representing an object or idea; Ideographic, graphemes[1] representing an idea or concept). 2) *transitional system*, with graphemes referring not only to the object/idea, but also to his/her own name. 3) *phonetic system*, with graphemes referring to sounds or spoken symbols not necessarily associated with a particular meaning (verbal, grapheme representing a whole word; syllabic, grapheme representing a syllable; alphabetic, grapheme representing a basic sound). [15].

Nowadays, the picture-writing system is associated with Sino-Tibetan or Indo-Iranian languages such as Chinese or Bengali, while the phonetic system is associated with European languages such as English or German.

Between the 4th and 8th centuries BC, Latin and Greek scribes used uncial, a type of majuscule script (capital letters) with highly detailed decorative surfaces [7]. This type of script persisted until the early Middle Ages, when paleographers tracked down a new type of script known as semi-uncial, which resembled today's lowercase. When Johannes Gutenberg invented letterpress printing in 1439, the technique became popular throughout Europe. Letterpress printing included a system for sorting letters and symbols and sped up the process of copying text [9]. By the 18th century, letterpress printing was a common practice for books and newspapers. Majuscules were used infrequently and were sorted into wooden boxes (or cases) located on the upper shelves, while minuscules were used more often and were located in wooden boxes on the table for easy access. This is the etymological reason for the use of uppercase for majuscules and lowercase letters for minuscules [10]. The traditional use of capitalization of letters in natural language is easily demonstrated in languages such as German, where all nouns or the formal pronoun for you, Sie, must begin with a capital letter [6]. With the influence of printing methods, the capitalization of nouns begins to follow German typography. In the 18th$\hat{}th$ century, more uniform rules for the use of capital letters in English were established [17], covering not only the capitalization of the first letter in a paragraph. Other items such as chemical formulas (e.g., nitric oxide, NO or sodium sulfate, Na_2SO_4) or abbreviations (e.g., PhD or MSc or MD).

In the mid-1900s, advances in science and technology led to a new era in which identifiers with multiple words were needed. At that time, computers were limited in capacity and functionality, as was the programming language on which they were based. The technology was very limited to cover an entire world of orthographic rules that were grammatically and syntactically correct. In the early days of programming languages, capitalization was ignored (flatcase). Decades later, capitalization in higher programming languages serves a purpose. For example, languages like FORTRAN, which are case-insensitive, have very limited features. However, case-sensitivity in languages like C added more features. Nowadays, we continue to use the C language, while FORTRAN is considered almost obsolete.

To create functional identifiers, programmers used efficient characters (such as hyphens or underscores) and/or capitalization of letters to combine multiple words into one. Matching capitalization in chemical formulas is now a common practice for identifiers, called medial capitalization (also called CamelCase). CamelCase is a letter case that omits the spaces between words and separates words with a single capitalized letter (e.g., commonCase or thisIsAnExample) [18]. Another common letter case in programming is kebab case, where the spaces between words are replaced with hyphens to give the impression that it is a kebab meal on a stick (e.g. common-case oder this-is-an-example). Another popular letter case similar to kebab case is snake case, with the

[1] abstract symbol

difference that underscores are used instead of hyphens to replace spaces (i.e. common_case or this_is_an_example). Thanks to technological advances, it is now possible to expand the upper and lower case letters used for identifiers. New uppercase and lowercase letters or a combination of existing uppercase and lowercase letters have emerged as identifiers for identifiers of words with multiple-word identifiers (i.e., PascalCase, camel_Snake_Case, SCREAMING_SNAKE_CASE, TRAIN-CASE, Pascal_Snake_Case, HTTP-Header-Case, etc.). Because of these emerging new styles, a naming convention was created that provides rules for specifying the string required for an identifier used for variables, methods, types, and others in any programming language [16].

Hypotheses

- Case type: participants will respond faster and possibly more accurately if they read letter cases closer to a natural language. Taking into consideration the academic background of the participants:

  Participants with a computer science background (CS) will respond faster to camel case than to that of kebab case.

  Participants with other field (OF) than computer science will respond faster to the kebab case than to the camel case.

- Color case: For a color-coded case in which each word has a different color, participants would respond more quickly to a color distinction (chromatic cases) than to a single-color (monocromatic cases), regardless of whether the case is the camel case or the kebab case.

2 Materials and Methods

All experiments were performed following the instructions from the course Experimentation and Evaluation (Spring Semester 2022) as part of the curriculum program for BSc in Informatics at the Università della Svizzera italiana, Lugano, Switzerland.

2.1 Variables

This study focuses on the following two letter case styles: *Camel case* contains a collection of words separated by an uppercase character instead of a space. The capitalization of the first letter per word starts from the second word (e.g., thisIsCamelCase). *Kebab case* comprises a collection of lowercase words separated by a hyphen (e.g., this-is-kebab-case). As part of our hypotheses, we wanted to test the effect of color on the readability of both cases, which we called *Color case*.

Hence, we had two sets of cases, monochromatic (single color) and chromatic[2] (multiple colors). The monochromatic case contains a multiple-word identifier with black-colored text. Finally, the chromatic case contains a multi-colored text, where each word and the hyphen (for the kebab case) have a different color. All the text was displayed over white background.

Although we did not request medical information to declare whether participants have any deficient color vision conditions or not, for the Color case style, all experiments were treated within color-blind-safe palette, as described on Paul Tol's notes [13]. This palette is shown in figure 1 and it is composed of four contrasting colors as described on table 3.

Figure 1: Color blind safe palette.

2.1.1 Independent variables

Variable	Levels
Multi-letter case	Camel case
	Kebab case
Color case	Monochromatic Camel case
	Chromatic Camel case
	Monochromatic Kebab case
	Chromatic Kebab case

Table 1: Independent variables. Three major independent variables where included in the study. Each case must contain at least two different words. Monochromatic refers to black color font and chromatic to multiple blind-color safe colors.

[2]The authors use the words 'chromatic' and 'color' interchangeably

2.1.2 Dependent variables

The two main dependent variables are *response time* and *correctness*.

Response time is calculated as the time it takes in milliseconds (ms) for a participant to answer a question. A non-displayed timer mechanism is used as we wanted to keep the user interface as clean as possible and avoid causing stress that could add noise to the experiment.

Before every question appears, a "continue" button is shown on the screen. When the user presses it, the start time is registered, and when they select their answer, the time difference between the end time and the start time is computed.

The JavaScript `Date().getTime()` function is used for this purpose. Technically, when a question begins, the function sets the value of a variable named `startTime` and when the user chooses an answer, the *respone time* is calculated as `new Date().getTime() - startTime` and the result is assigned to the corresponding question in the responses object.

The total *response time* includes the time for reading a question and selecting one of the four possible given options.

Correctness is determined when the participant answers a question correctly. Each question contains a single correct answer and three incorrect ones, which may contain one or more distractors[3]. The *correctness* is calculated as the ratio between correct answers over total number of responses during the evaluation of the results and presented as a percentage.

Variable	Measurement scale
Response time	milliseconds (ms)
Correctness	true-false ratio (%)

Table 2: Dependent variables. The response time is calculated as the time for reading and selecting an answer per question. The correctness is calculated as the ratio of correct answers per total answers.

[3]A distractor is a word that looks similar to another word (demi-homograph) used to add controlled noise. This may cause participants to be confused and choose the incorrect answer

2.1.3 Control variables

Variable	Fixed value
Font family *(in fallback order)*	Open Sans *(preferred)*, Helvetica Neue, Helvetica, Arial, and sans-serif
Font size	question: 1.5em words to find: 2em options: 3rem
Font weight	question: bold words to find: bold options: 700
Font color *(general)*	question: #a6a6 words to find: #4b4b4b options: #4b4b4b
Font color *(color case)*	question: #a6a6 words to find: #66cc33, #ccbb44, and #ee6677 hyphen: #228833
Background color	#f5f5f5
Identifier position	upper half of the screen, vertical middle and centered
Number of options	4 (per identifier)
Options' position & alignment	lower half of the screen, centered
Options' displayed & alignment	2 per line (desktop) 4 per line (mobile)
Action buttons color	#ffd300
Action buttons position & alignment	bottom, centered

Table 3: Control variables: styling. All colors in hexadecimal representation. *See* section 5. An option is considered as a possible answer, given a question. An identifier is the initial given comparable set of words that the user needs to find among the given options. An action button is used by the user to proceed to next available question.

Variable	Fixed value
Machine *(programming)*	Apple MacBook Pro
CPU	2.6 GHz 6-Core Intel Core i7 CPU
Memory	16 GB 2400 MHz DDR4
Graphics	Radeon Pro 555X 4 GB Intel UHD Graphics 630 1536 MB
Operating System	macOS Catalina 10.15.7 (19H1715)
IDE	WebStorm 2021.3.3 Build #WS-213.7172.31
Programming language	JavaScript (ECMA-262)
Back-end technologies	Node.js v.17.8.0 & Express.js v.4.16.1
Front-end technology	React v.18.1.0
Email delivery service provider	Mailjet
Front-end deployment PaaS provider	Netlify
Back-end deployment PaaS provider	Digital Ocean
Back-end service	App Platform
Back-end specifications	1 vCPU & 512 MB RAM (basic plan)

Table 4: Control variables: software & hardware. This table shows the software and hardware used to create the experiment's applications and the services used to deploy and run the applications on the cloud. Abbreviations: IDE, Integrated development environment; PaaS: Platform as a service

2.2 Apparatus and Materials

To develop the experimentation suite (front-end and back-end applications), we used an Apple MacBook Pro with 2.6 GHz 6-Core Intel Core i7 CPU, 16 GB 2400 MHz DDR4, and Radeon Pro 555X 4 GB and Intel UHD Graphics 630 1536 MB, which runs on macOS Catalina 10.15.7 (19H1715). *See* table 4 for details.
The IDE used was WebStorm 2021.3.3 Build #WS-213.7172.31. The front-end application runs on Netlify's Edge CDN, while the back-end application runs on an App Platform instance of type "basic plan" with one virtual CPU and 512 MB of RAM.

2.3 Design

Type of study: Experiment
Number of factors: Multi-Factor Design
Within Subject Design
Sampling method: non-probabilistic by convenience

We were instructed to analyze two case styles, *Kebab case* and *Camel case*. We decided to add what we named *Color cases*, which is the *chromatic* or colored version of both previous *monocromtic* cases. The color was used to accentuate the separation between words.

The experiment applied to every participant was identical, a total of 69 participants were included in this experiment. When considering academic background, 67 participants were included, since two of them did not disclosed this information. The order of the tasks, the options for each task, and the colors were randomized.

Hence, the allocation of participants and experimental conditions were applied randomly to the experimental group.

2.4 Participants

We selected three main characteristics for our experimental group by background: computer science students, students of other fields, and high school students. The first two were subdivided into undergraduate and graduate students.

There was no control over who would participate as recruitment of participants happened through

the researchers' academic, family and friends networks.

2.5 Procedure

We developed a web application to extend our capacity to reach potential participants and automate the collection of results. Architectonically, we divided it into back-end and front-end to separate concerns and improve maintainability (*See* table 4 for further details).

2.5.1 Back-end

We created a RESTful API using Node.js version 17.8.0 and Express.js version 4.16.1.

It provides the following endpoints with their corresponding method denoted in capital letters and, if it applies, the respective parameters in the form `:paramName` (without including the colon):

- `GET /users/getId`

  Provides random 16-digit alphanumeric IDs

- `GET /tutorial/get/:userId`

  Generates a set of three pre-defined and pre-sorted questions with four options each in JSON format

- `GET /questions/:userId`

  Receives a userID and, with a helper function, queries the local hard disk for the user's pre-generated set of questions. If it does not find it, it produces a randomly sorted version of a predefined set of questions and their options which order is also randomized.

- `GET /responses/:userId`

  Given a user ID, read a JSON file on disk, and return it as a JSON object

- `POST /responses/submit/:userId`

  Receives the users' JSON responses, which come stringified in the body of the request. It then saves them on disk as JSON and CSV files with the userId as their filenames. It uses several helper functions to send an email to the researchers with both generated files as attachments identifying the userId in the subject.

- `GET /responses/get/reports`

  Uses a helper function to process all the JSON files to generate in-memory three different reports in CSV format with a timestamp appended to the filename:

  a) `general (timestamp).csv`
  b) `kebab_vs_camel(timestamp).csv`
  c) `mono_vs_color(timestamp).csv`

  When it finishes, it compresses the files into a ZIP file that is then downloaded automatically to the requester's device with the file:

  `EExp2 - Reports (timestamp).zip`

2.5.2 Front-end

The user interface (UI) was designed as a single-page application in a minimalistic style to reduce distracting elements that could interfere with the participants' concentration during the experiments.

It was structured in the following stages:

1. Introduction and informed consent explaining:

   The purpose of the project in one sentence

   What data is collected, and how it is used

   Anonymization of data

   Explanation of the rights of the participant

   Consent request and collection of age, gender, and academic background of the participant

   Identification of the university, the course, and the researchers

2. Quick tutorial instructions:

   The tutorial instructions consist of four steps. These describe what will appear on the screen and its position, the specifications on what the participant should do, and how to interact with the UI.

3. Tutorial:

   This section aims to familiarize the participant with the visual, conceptual, and mechanical logic of the experiment.

There are three tasks: one with a camel case, one with a kebab case, and one with a color camel case.

The application gives feedback on the result of the participant's selection

4. Experiment preamble:

   A quick pause to allow the participants to get mentally ready to perform the actual experiment

5. Experiment:

   Twenty randomized tasks functionally identical to the ones in the tutorial

6. Done: Thank you and share

   Used to thank the participant for joining the study

   Buttons to share the experiment's URL by copying the URI to the clipboard, WhatsApp, and email.

React version 18.1.0 was used to allow dynamic generation of all HTML content. A combination of state control and local storage data persistence enabled the application to anonymously "remember" the user, the stage in which the participant is, the questions, the responses, and other flow control variables.

The React components and functionality, in general, were designed considering further implementation of additional style cases. However, simple modifications solely on the front-end are needed.

Front-end: Inner workings

The UI checks if an ID is already stored in the browser's local storage upon loading the website. If it does not find one, it will request the back-end to generate a random unique ID.

If there was a previous ID after the new ID is received, the UI checks if a stringified JSON object contains the tasks (questions) assigned to the user. If it does not find it, proceeds to ask the server to generate one. In the same way, it looks for the tasks for the tutorial.

On every click of the continue button that appears asking the user to proceed, the stage is updated to the next one, and at the same time, and persists it in the local storage.

Everything described above allows the application to show the content at the same stage the user was if they reload the browser page or leave and come back another time.

During the tutorial, no state is tracked apart from the iteration over the tasks. Upon completion, the user can repeat the tutorial or proceed with the experiment.

On the contrary, the tasks and selected options are tracked in a responses JSON file generated within the same UI during the experiment. The data is kept in memory and the local storage until the user finishes the last task.

The responses are transmitted to the server without any user intervention, even though the last button asks the user to click it to send the data. This was designed to anticipate any user forgetting to click on it.

After the user proceeds, we thank the user for their participation. We also provided three methods to share the experiment and invited them to copy the URL to memory or send a predefined invitation message via WhatsApp or email.

2.6 Data analysis

The data was collected as JSON files and the reports were processed as CSV files (*See* 2.5 Procedure). Data were analyzed offline by using IBM® SPSS® Statistics version 27, 64-bit edition and/or JASP versions 0.16.2 (Intel and Silicon). The one-sample t-test was used to compare data within the same experimental group (i.e. accuracy_kebab, only), whereas paired-sample t-test was used for comparison between different groups (i.e. accuracy_kebab *vs.* accuracy_camel). The results were presented as mean±SEM (standard error of the mean) with the level of significance set as $p < 0.05$.

2.7 Replicability

The collected raw and analyzed data is packaged is provided to corroborate the findings (*See* Appendix).

We also generated artifacts that can be used to replicate the experiment. The source code is available on GitHub for the Front-end[4] and for the

[4] https://github.com/erickgarro/EE-Cases-Styles-Experiments-Frontend

Back-end[5]. The applications can be run locally or on the cloud.

2.7.1 Running the applications locally

To run the applications on your computer, open a terminal on your computer and navigate to the folder where you want to download the code. Then type the following commands[6]:

```
git clone https://github.com/
    erickgarro/EE-Cases-Styles-
    Experiments-Backend.git

git clone https://github.com/
    erickgarro/EE-Cases-Styles-
    Experiments-Frontend.git
```

The following folders will be created

```
EE-Cases-Styles-Experiments-Backend

EE-Cases-Styles-Experiments-Frontend
```

2.7.2 Environmental variables

After that, these environmental variables need to be set[7]:

Front-end:

```
REACT_APP_SERVER=<backend-server-URL>
  [<:PORT NUMBER>]
```

For local execution use `http://localhost:3001` For cloud execution refer to Running the applications on the cloud.

Back-end:[8]

```
MAILJET_API_KEY=<32-chars key>
MAILJET_SECRET_KEY=<32-chars key>
MAILJET_FROM=<sender's email>
MAILJET_TO_1=<recepient 1 email>
MAILJET_TO_2=<recepient 2 email>
```

[5] https://github.com/erickgarro/EE-Cases-Styles-Experiments-Backend

[6] In case you do not have Git installed, follow this guide.

[7] The enclosing "<>" denote a required value and "[ ]" refer to an optional value. These characters must be excluded.

[8] The application assumes two researches will receive every participant's results individually. The code can be modified to add or remove recipients, or to stop emails from being sent.

The following are guides to set the environmental variables on your system:

- Linux
- macOS
- Windows

To execute the back-end server, follow theses instructions:

1. Navigate to the back-end folder by typing:
   ```
   cd EE-Cases-Styles-Experiments-
      Backend
   ```
2. To download and install[9] all the dependencies type:
   `yarn install`
3. To run the server type:
   `yarn start`

   The server is now accessible at http://localhost:3000

To run the front-end server:

1. Open a new terminal
2. Navigate to the front-end folder named:
   ```
   EE-Cases-Styles-Experiments-
      Frontend
   ```
3. To download and install the dependencies type:
   `yarn install`
4. To run the server type:
   `yarn start`
5. When notified that port 3000 is in use, type "Y" to initialize another port. A browser window will open at http://localhost:3001
6. You can run now the experiment[10].

[9] If Yarn is not installed on your computer, follow this guide

[10] If you are running the experiment with multiple users on the same computer, make sure you copy the URL and paste it in a new private/incognito browse. When a user finishes, close the window and this process. This is necessary because data is stored in the browsers local storage to prevent the same person to participate more than once.

2.7.3 Running the applications on the cloud

The applications were optimized to run on remote servers, specifically Netlify for the front-end and Digital Ocean for the back-end. You are be able to run them using other providers but you might need to modify the code accordingly.

To start, you need to fork both repositories. If you have a GitHub account, follow this guide if needed, otherwise, you need to clone the repositories and push them as new ones where you do your version control.

Mailer system:
As a backup, authorized researches will always receive an email with a copy of each participant's results, in CSV and JSON formats. If required, this can be removed by modifying the source code.

We used Mailjet to handle our mailing. If you do not have an account, consider the following steps:

1. Open an account by visiting this link.
2. Choose your account type. We recommend the developer account as you get a good enough free-tier.
3. Create and get your API and Secret Keys by visiting this link.

Back-end:
If do not have a Digital Ocean account, sign up to get one. Then follow these steps for creating a new app in their App Platform:

1. Go to Create a new app
2. Select your source code provider and follow the instructions to authorize Digital Ocean to access your repository and choose your source branch. Leave "Autodeploy" selected and click *next*.
3. Edit your plan. We recommend a basic 512 MB RAM | 1 vCPU instance, or the cheapest one at the time you follow this steps. When done, click *back* and then *next*.
4. Add your environmental variables (gotten from the Mailer system steps) as indicated in section Environmental variables, and click *next*.
5. Choose the global region to use for your deployment.
6. Review your selection and click on *create resources*.
7. Add your credit/debit card information and complete your order.
8. Digital Ocean will build and deploy the site for you. In the account dashboard, the deployment status will be displayed indicating if the site is up and running.

Front-end:
If do not have a Netlify account, proceed and open a new one. Then follow the step-by-step guide: Deploying on Netlify. To setup your environmental variable for the front-end refer to section Environmental variables. You can set it during the setup of your site, or afterward by accessing your site settings[11].

Netlify will build and deploy the site for you. In the account dashboard, the deployment status will be displayed indicating if the site is up and running.

You can now share your site's URL with your participants.

Persistence of the responses JSON files on the back-end server:
After a participant finishes the experiment, their responses will be stored on the back-end server in JSON as CSV files. These files will be lost every time you push your commits to your source branch. Since each push triggers the build process, the virtual instance that runs your server will be destroyed and then recreated.

To consider those files the next time you request the reports via the REST endpoint, you need to add a copy of the JSON and CSV files received by email into your source code's `/data/responses` folder. Following this, you have to commit and push the additions.

[11] https://app.netlify.com/sites/<sitename>/settings/deploys#environment

2.7.4 Downloading the results files

To download the reports of your experiment, access the following URL:

```
<backend-server-URL>[<:port number>]/
   responses/get/reports
```

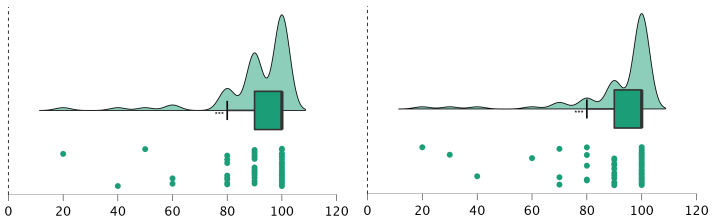
Refer to section 2.5.1 Back-end for more details.

3 Results

3.1 Letter case correctness

3.1.1 General responses

To compare measures of accuracy indexes between kebab case and camel case in reference to the total population, we observed no significant difference between kebab case (90.72%±1.79, n=69 participants) and camel case (91.74%±1.95, n=69 participants), *see* table 6 and table 7 for further details. Despite the fact that participant's accuracy to camel case was slightly higher than kebab case, both data samples did not represented a difference each other (*See* Appendix Statistics: Accuracy & time responses for Kebab case vs Camel case, general approach). When analyzing the data within each data set (either kebab- or camel-case), the vast majority of participants had high accuracy responses, about over 80%. However, as shown in figure 2, the data within each group is highly disperse ($p<0.001$ by one sample t-test. *See* table 8 for further details).



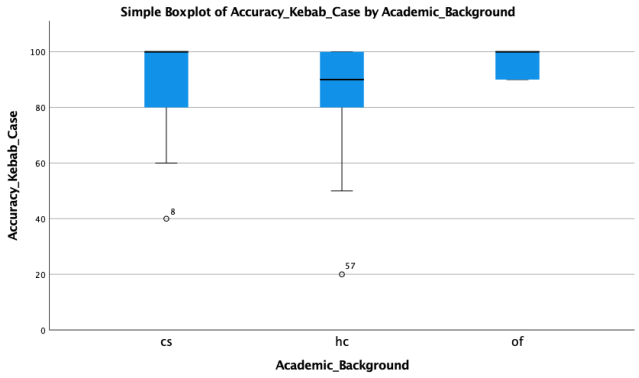
(a) Accuracy kebab case **(b)** Accuracy camel case

Figure 2: Accuracy responses for kebab vs camel case. Raincloud plots showing the general dispersion of median responses on participants when answering (a) kebab case or (b) camel case. The accuracy index is calculated as the ratio of correct answers over total answers times 100% per individual. Each dot correlates to the accuracy index per participant (n=69 participants). Notice that in either group data is strongly segregated >80%. Mean 90.725% in kebab case and 91.739% in camel case, ***, $p<0.001$ by One sample Student t-test. No significant correlations was observed between the two groups, kebab case or camel case, ($p>0.05$) by Paired sample Student t-test.

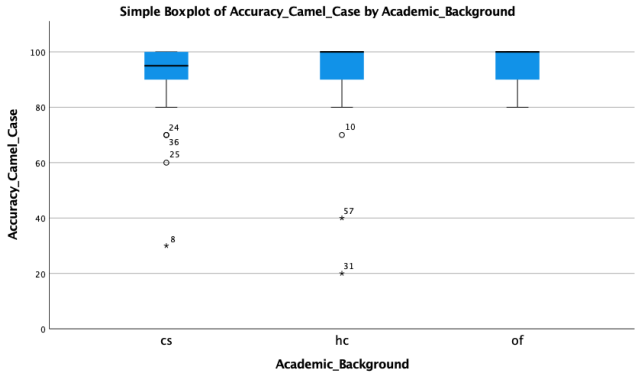
3.1.2 Correctness by academic background

The participants were classified in three different populations based on their educational background: 1) Computer Sciences (CS), includes graduates and undergraduates participants whose background is related directly to the Computer Sciences field. 2) High School (HS), includes participants whose educational degree are below university and are active students in a High School or Middle School program. 3) Other Fields (OF), includes graduates and undergraduates participants whose have completed or are currently enrolled in formal university education degree other than Computer Science. Participants that replied N/A as background were excluded for this analysis (n=2 participants).

In figure 3, we appreciated a more detailed response of participants based on their academic background. First of all, in general, we noticed a wider range in the responses from CS and HS participants (oscillating between 50% to 100%) when exposed to kebab case questions than camel case questions (>80%). Surprisingly, we observed a more homogeneous accurate response in OF participants (n=25 participants) for either kebab case (95.60%±1.01) or camel case (96.00%±1.41), in comparison with the other groups. In camel case, we can appreciate a more concentrated population of participants who responded accurately (>80%) for CS (89.17%±3.45, n= 24 participants) and HS (88.33% ±5.44, n= 18 participants). Unfortunately, these two groups had the highest number of outliers with low percentage of correct answers. Despite this, we observed that CS participants perform better when exposed to camel case than kebab case, whereas the opposite is true for HS. Unfortunately, the data did not show strong evidence to support these claims ($p>0.05$ paired t-test. *See* Statistics: Accuracy responses per educational background for Kebab case vs Camel case for further details). Thus, we claimed that there is no significant difference in the responses based on the educational background when considering kebab case (CS: 88.75%±3.31, n=24 participants; HS: 86.67%±4.92, n=18 participants; OF: 95.60%±1.01, n=25 participants) vs. camel case. (CS: 89.17%±3.453, n=24 participants; HS: 88.33%±5.44, n=18 participants; OF: 96.00% ±1.41, n=25 participants).



(a) Accuracy kebab case



(b) Accuracy camel case

Figure 3: Accuracy responses for kebab vs camel case based on academic background. Boxplots showing the mean responses on participants (n=69 participants) when answering (a) kebab case (n=10 questions) or (b) camel case questions (n=10 question). Data was grouped in three clusters based on their academic background 1) Computer Science (cs, n=24 participants), 2) High School (hc, n=18 participants), 3) Other field (of, n=25 participants) than Computer Science. Participants who selected not answer (n/a), were excluded (not shown, n=2 participants). The accuracy index is calculated as the ratio of correct answers over total answers times 100% per individual. No significant correlations was observed between the two major cases or within cluster groups (p>0.05) by Paired sample Student t-test.

3.1.3 Color case responses

The paired t-test from the table (b) in figure 6 reveals a significant difference in responses with a small magnitude of experimental effect when the correctness of responses Camel monochromatic and the Camel chromatic cases are compared, based on Cohen's d distance. This implies that we can partially reject the null hypothesis as there is a definite, consequential relationship between its two variants for the Camel case. Hence, we can accept the alternative hypothesis confirming that at least, for this case style, a color distinction improves the participants response time.

Contrary to Camel case, the elapsed time for Kebab monochromatic and chromatic cases and the accuracy of all case styles, we had to accept the null hypothesis.

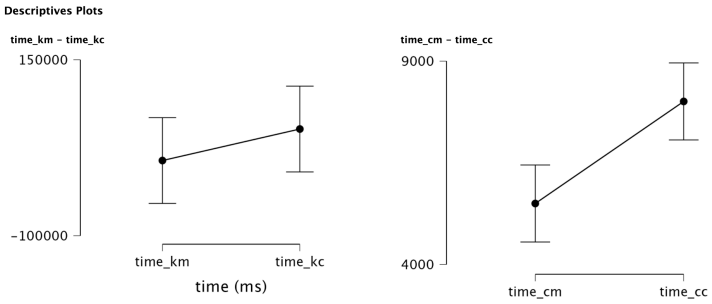


Figure 4: Descriptive plot of the elapsed time relation between Kebab case (on the left) and Camel case (on the right). Notice the significant difference ($p < 0.001$) between pairs for Camel case. Abbreviations: time, the response time taken by a participant took to answer a question; km, Kebab case monochromatic; kc, Kebab case chromatic; cm, Camel case monochromatic; cc, Camel case chromatic.

Figure 4 shows the effect in reading time given by the use of color to accentuate word separation. The plot on the right shows a strong significant effect (improvement) when the chromatic color style is applied to the Camel case.

Paired Samples T-Test

Measure 1		Measure 2	t	df	p	Mean Difference	SE Difference	Cohen's d
accu_km	-	accu_kc	0.866	334	0.387	1.791	2.069	0.047
accu_cm	-	accu_cc	1.443	334	0.150	2.687	1.861	0.079

Note. Student's t-test.

(a) Correctness monochromatic vs chromatic

Paired Samples T-Test

Measure 1		Measure 2	t	df	p	Mean Difference	SE Difference	Cohen's d
time_km	-	time_kc	−1.022	334	0.308	−44758.913	43802.611	−0.056
time_cm	-	time_cc	−3.686	334	< .001	−2508.955	680.759	−0.201

Note. Student's t-test.

(b) Elapsed time monochromatic vs chromatic

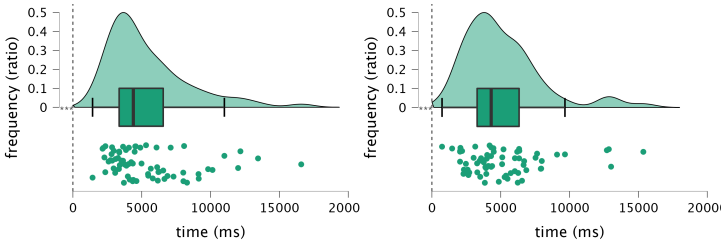
Figure 5: Color case paired sample Student T-Tests Table (a) shows no significant differences when comparing correctness among Kebab monochromatic and chromatic cases. Table (b) does not shows significant difference when elapsed time between Kebab monochromatic and chromatic cases are compared. It does reveal a significance value $p < 0.001$ and a small magnitude of experimental effect in elapsed time given by the value $|d| = 0.201$, when Camel monochromatic and chromatic are compared.

As noted in section 3.2.1, there was no major difference between camel case and kebab case responses' time. However, when we considered a color case distinction in each case population, treated as monochromatic or chromatic (when each word has a different contrasting color) cases, we discovered that only camel case has a strong disparity when color case was applied (camel case monochromatic: 4350.25 ±250.69 *vs.* camel case chromatic: 5861.23±410.54, p<0.001. *See* table 6 and 7)

3.2 Letter case efficiency

3.2.1 General time responses

The general response time of participants did not differ between kebab case (5390.77 ms ±358.42, n=69 participants) and camel case (5101.54 ms ±342.63, n=69 participants. *See* table 6 and 7). As shown in figure 6, the response time has a wide range between 0 to 18,000 ms, with a higher concentration between 1,000 to 8,000 ms (*See* Appendix Statistics: Accuracy & time responses for Kebab case vs Camel case, general approach for further details).

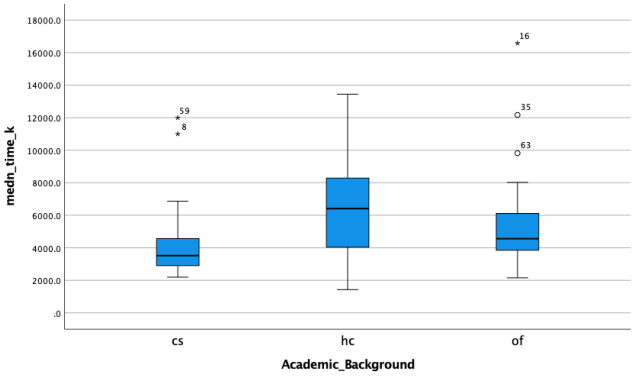


(a) Time kebab case **(b)** Time camel case

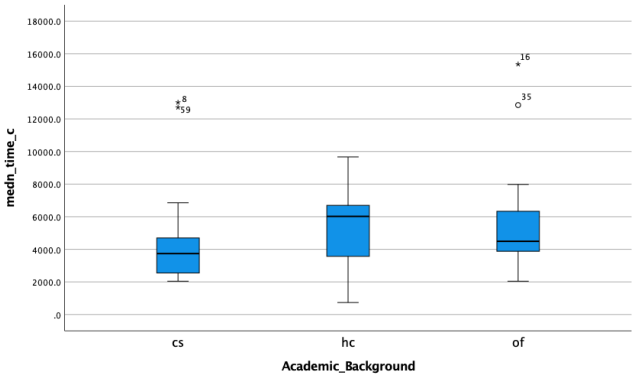
Figure 6: Time response for kebab vs camel case. Raincloud plots showing the general dispersion of median time responses, in milliseconds (ms), on participants when answering (a) kebab case or (b) camel case questions. The time is calculated as the total time for reading a question and selecting one of the four possible given options. Each dot correlates to the mean response time per participant (n=69 participants). Notice that in either group data is strongly segregated, with a higher concentration of time responses 5000 ms. The mean response time corresponds to 5390.77 ms ±358.42 in kebab case and 5101.54 ms ±342.63 in camel case, ***, p<0.001 by One sample Student t-test. No significant correlations was observed between the two groups, kebab case or camel case, (p>0.05) by Paired sample Student t-test.

3.2.2 Responses and academic background

Analogous to responses correctness data (*See* Correctness by academic background), time responses are highly disperse within each clustered-group (*See* table 13). In particular, as shown in figure 7, the response time of participants enrolled in a high school program (HS) is broadly disperse compared to university graduates and/or undergraduates, principally for kebab case questions (HS kebab case: 6529.278 ms ±700.774 *vs.* HS camel case: 5532.083 ms ±567.667, n=18 participants). In fact, the time responses for participants within Computer Science (CS, 4380.896 ms ±510.823, n=24 participants) field is substantially lower on kebab case than HS in the same case group. Unfortunately, participants belonging to other fields (OF) than Computer Sciences shown a largely spread data with several outliers in either kebab or camel cases (5101.54 ms ±645.986 in kebab case *vs.* 5404.360 ms ±619.720 in camel case, n=25 participants)



(a) Time response kebab case



(b) Time response camel case

Figure 7: Time response kebab vs camel case based on academic background. Boxplots showing the mean responses on participants (n=69 participants) when answering (a) kebab case (n=10 questions) or (b) camel case questions (n=10 question). Data was grouped in three main clusters based on their academic background 1) Computer Science (cs, n=24 participants), 2) High School (hc, n=18 participants), 3) Other field (of, n=25 participants) than Computer Science. Participants who selected not answer (n/a), were excluded (not shown, n=2 participants). The response time was calculated as the mean time for answering each group of questions (kebab or camel case). No significant correlations was observed between the two groups (p>0.05) by Paired sample Student t-test.

4 Discussion

Language is the most intelligible representation of thought, even though many of the processes behind the creation and development of languages are still unknown. As a method of communication and a cultural phenomenon, human language is quite rudimentary, though inherently complex. Humans acquire language through nature and nurture. To this date, humans do not have a complete understanding of language acquisition processes, and studies show conflicting results about

why and how humans acquire their languages as children and even as adults. So far, we know that read/written language (RW-L) is a human creation and is therefore as error-prone as it is grandiose. In our modern society, people are taught RW-L at a young age. In most cases, before we have fully developed our language skills (reading-writing-speaking). Also, we learn RW-L before our brains are fully developed. Therefore, many of the cognitive processes and brain rewiring that lead to our learning and understanding RW-L are not well understood. Nevertheless, some studies have shown that the type of learning that infants exhibit when learning a language may be characteristic of artificial natural language processing (NLP). NLP is a branch of computer science and artificial intelligence that gives computers the ability to perceive text and spoken language in ways similar to humans. NLP is one of the most comprehensive branches of computer science research, as it combines computer modeling, social science, neuroscience, psychology, literature, linguistics, and more.

The present study demonstrates participants' inherited abilities in reading language (R-L) recognition. We were able to compare scholars at different levels (high school, undergraduate, and graduate students & professionals) with an *a posteriori* knowledge of programming languages, programmers accustomed to a particular RW-L style using upper and lower case, such as camelCase, or with a filler separation between words, such as kebab-case; with those who have an *a priori* experience with these styles.

4.1 Web application vs Desktop application

The participants were instructed to use the web application EExp2, where they completed a tutorial and then a twenty-task experiment. A web application was chosen over a desktop application to make it easier for users to utilize the tool and for researchers to collect results.

4.2 Letter case response

Initially, we considered that participants with a computer science background would be able to read and comprehend better words in camelCase better than kebab-case. Surprisingly, we noticed that this is not the case. We have no strong evidence that supports this claim. Furthermore, when we compared the kebab-case vs camelCase, we noticed that not only the correctness in responses was not significantly different between groups, but also the time of response. Participants tend to have a better response to camelCase than kebab-case, but this difference did not fulfil our criteria (>95% difference) to be considered significantly different. This result was particularly interesting, since contradicts previous literature (*See* epelhoim, 1997) that advocates for the usage of spaces and fillers in a text and the fact that by using Latin-letter fillers to remove gaps from text reduced reading by only 10-20% [5]. A possible explanation to our results involves the usage of reduced number of words in identifiers (identifiers with just two to three words). This is because, when using a small number of words it is much easier to read and identify the begin and end of a word. Particularly, as we live in 'the social media era' where people are used to usage of short plain identifiers (hashtags) to express feelings or link content (i.e. #ilovepizza, #SaveUkraine).

Interestingly, a similar study by Binkley *et al.* [2] comparing camelCase with under_score case showed that camelCase resulted in higher accuracy for all participants regardless of their training in programming skills, but those taught camelCase were able to recognize camelCase-style identifiers faster than under_score-style identifiers. This supports our initial hypothesis that participants with a computer science background will perform better on camelCase. Unfortunately, our results did not support this claim. To do this, we must consider the length of the identifiers, as Binkley and colleagues used long text in blocks. Therefore, we recommend a different approach in the future that considers a larger number of words within each notation.

Despite these findings, another interesting aspect of this study is the correctness compared to the use of uncommon English words (i.e., telotaxi and asphyxy) and their demi-homograph pairs acting as distractors (i.e., tropotaxi and apoplexy, respectively. *See* Questionnaire). Notwithstanding the omission of lexicographic analysis, we found that complex words were not a barrier to correctness. Accuracy was relatively high even when un-

usual words were used. However, further analysis is needed to confirm this statement.

4.3 Color case responses

The fact that we have considered a color difference causes our brain to recognize words based on color rather than meaning or shape [8]. Previous studies have shown that when parts of the words (lowercase and uppercase) are removed it may have an effect in worsening the processing time [**?**]. It is proven to take some time to complete a word, and it is also shown to be a right hemisphere activity. It is wellknon that there is a language region in the right hemisphere (known as the Brocca's region) associated with language processing. Our results may indicate that people who distinguished Camel case color versus monochrome were posibly identifying the words by color (which belongs to the occipital portion of the brain). However, given the current study, we are unable to claim this statement. It may be possible to deep into this topic by exposing candidates to fMRI when perfoming these series of experiments (*See* section Materials and Methods).

4.4 Limitations and threats to validity

During this study, there are different aspects that may be considered when interpreting and understanding the data. Some external issues may directly or indirectly compromized the results shown above. Several efforts were made to minimize error and keep a high reproducibility when conducting this study. The research team followed closely the implementation of this study to assess its relevance and effectiveness at delivery.

Some of the aspects that may compromise data are described below:

- Distractors during experiments.

  Issue: People who get distracted during the experiment can lead to lengthening the response time, but not necessarily the accuracy responses. The opposite, could be also true. People who get distracted during experiments, could response faster but inaccurately.

  Solution: Either aware volunteers to reserve 10 to 15 minutes to fully focus in the assignment

- hardware or software malfunction.

  Issue: Since the experiment is run over the network. Any issue related to either internet access or device malfunction may lead to lengthening the response time.

  Solution: We could recommend the participants to check the internet service provider connection and close close extra software during the experiment. We partly check this issue by providing a small tutorial with three different questions, that would potentially highlight any issue before the experiment.

- Device selection.

  Issue: The usage of devices with small screen size can lead to wrong time response. A possible flaw involves the device where the participant answer the questions. If a participant used a mobile phone, the screen resolution is not good enough to show question and all answers in the same screen. This will add time of answering, since the participant must scroll down to click continue (this stop the counter for the specific question, and start the counter for the next one).

  Solution: We suggested the participants to use a laptop or desktop computer when solving the questionnaire.

- Data collection modifications.

  Issue: Changes in the standarized questionnaire during data collection (Age group) can lead to error. The age was considered as a possible factor for comparison (independent variable), as some individuals young individuals are highly trained in fast computer responses or other skills that gives them advantages. For example, kids and adults with gaming experience, middle age people like psychologists with experience in language processing and fast reading skills, among others. Whereas elders may not have strong computational skills or have slower responses.

Solution: Unfortunately, age group cases could not be considered as underage individuals were considered and added in the study later after initial collection of data.

- Sensitive Information. Limitations on data collection

  Issue: Due to the lack of permissions and/or licences related to the collection of user data with sensitive information (i.e. medical records, criminal records, socio-economical status, etc.) and not following the standard procedures for handling confidential information licence under Swiss law for conducting ethical user research, we were unable to include in the study information regarding language/speech impairment, such as dyslexia, aphasia, dysphasia, and so on. The inclusion of participants with any of these cognitive or speech disabilities may lead to errors and outliers that we were unable to exclude.

  Solution: Regarding this aspect, is not possible to have a solution. But it can be acknowledge the fact that we are not excluding individuals with these characteristics.

- Multiple responses per individual.

  Issue: A participant could possibly participate more than once. This is an important issue since he/she may have similar responses or have a rapid learning curve that favors him/her in later tries. In a way to avoid this issue, we created a cookie with device-specific information and a userID token (for privacy issues and reduce bias). However, the participant would potentially be able to repeat the experiment in a different device or in incognito mode. Since, we do not know the identity of any user, this may be a possible flaw in the current system.

  Solution: We can only ask participants to answer a single questioinnaire blank to avoid biases.

- Data processing and normalization.

  Issue:. The data collected, at least for time responses, is not normalized. We are aware that multiple word identifiers with a high number of words/characters will take physically longer to be read and process for an individual. The response time data is not normalized (i.e. divided per number of words or characters), as individuals could have find the answer to a question within the first option ($O(1)$), a normalization in this case may lead to wrong response time output. This normalization may be possible if the user need to search all over the options and select the correct one ($O(N)$).

  Solution: It is not possible to normalize data, but we could acknowledge it.

5 Conclusions

The study described in this report focuses primarily on the effects of the two main case styles, kebab case and camelCase, on the correctness and speed of responses by individuals with and without programming skills.

A general overview shows that participants respond somewhat faster to camelCase than to kebab-case. However, there is no clear evidence that participants respond better (correctness or speed) to either case style.

We developed a further analysis based on scholarly levels. We considered three major groups (high-school, computer science or other field). A notably distinction is observed when comparing individual responses based on their academic backgrounds. Particularly, the response time of untrained participants, high-school students, performed worst than trained computer science graduates and undergraduates. Despite we made the assumption that high-school students have no prior programming language training, we acknowledge that these participants may have some training. Thus, we cannot fully affirm that their performance is based on lack of knowledge. A further assessment on high-school students' programming skills may be done to discard any bias.

When considering a facilitated readability by color case, we observed no difference between plain or color responses. However, computer science individuals had a better performance for camelCase than kebab-case when using monochromatic color. This may be the result of a possible pre-conditioning

on plain color training, as computer science are prone to the usage of plain color layout (i.e. terminal or command window or standard programming IDE), a color case may be considered as a distractor per se that leads to a worst performance. Although, it may be necessary to expand the experimentation approach to consider this option (i.e. create a distinction on dark-mode vs light-mode programmers or terminal/command window users vs IDE users).

References

[1] Alexander C. (2018) "The importance of naming in programing, in: The mand in the Arena," https://carlalexander.ca/importance-naming-programming/, Accessed: Apr 30, 2022.

[2] D. Binkley, M. Davis, D. Lawrie & C. Morrell "To camelcase or under_score," 2009 IEEE 17th International Conference on Program Comprehension, 2009, pp. 158-167, doi: 10.1109/ICPC.2009.5090039.

[3] Chad J. Marsolek (1999) Task and Stimulus Demands Influence Letter-case-Specific Priming in the Right Cerebral Hemisphere, Laterality, 4:2, 127-147, DOI: 10.1080/713754331

[4] Caprile and Tonella (2000) "Restructuring program identifier names," Proceedings 2000 International Conference on Software Maintenance, pp. 97-107, doi: 10.1109/ICSM.2000.883022.

[5] Epelboim J, Booth JR, Ashkenazy R, Taleghani A, & Steinman RM. (1997) "Fillers and spaces in text: the importance of word recognition during reading." Vision Res. Oct;37(20):2899-914. doi: 10.1016/s0042-6989(97)00095-3.

[6] Flippo, H. (2020) "Capitalization in German. Comparing English and German Rules", https://www.thoughtco.com/capitalization-in-german-4069437, Accessed: May 1, 2022.

[7] Glaister, G.A. (1996) "Glaister's Encyclopedia of the Book.", 2nd edn., New Castle, DE, and London: Oak Knoll Press & The British Library, p. 494. ISBN 1884718140

[8] Microsoft (2020) "The science of word recognition". https://docs.microsoft.com/en-us/typography/develop/word-recognition. Accessed: May 14, 2022.

[9] Muzdakis, M. (2020) "Uppercase and Lowercase: An Etymological Tale of Two Scripts.", https://mymodernmet.com/uppercase-lowercase-letters/, Accessed: May 1, 2022.

[10] Nesbitt, A. (1957) "The history and technique of lettering", Dove Publications, New York.

[11] Pinker S. and Bloom P. (1990) "Natural language and natural selection," Behavioral and Brain Sciences, vol. 13, no. 4, pp. 707–727, doi: 10.1017/S0140525X00081061

[12] Soga R., Koreki G., Kanuka H., Ioku A. & Maeoka J. (2021) "Generating program identifier dictionary for maintaining legacy systems," Frontiers in Artificial Intelligence and Applications, Vol. 337 in: New Trends in Intelligent Software Methodologies, Tools and Techniques, pp. 3-12, doi:10.3233/FAIA210003

[13] Tol P. (2021) "Colour schemes and templates," Paul Tol's Notes. https://personal.sron.nl/~pault, Accessed May 04, 2022.

[14] Wikipedia (20.Apr.2022) "Sumerian language.", https://en.wikipedia.org/wiki/Sumerian_language, Accessed: May 1, 2022.

[15] Wikipedia (1.May.2022) "History of writing" https://en.wikipedia.org/wiki/History_of_writing#Cuneiform_script, Accessed: May 1, 2022.

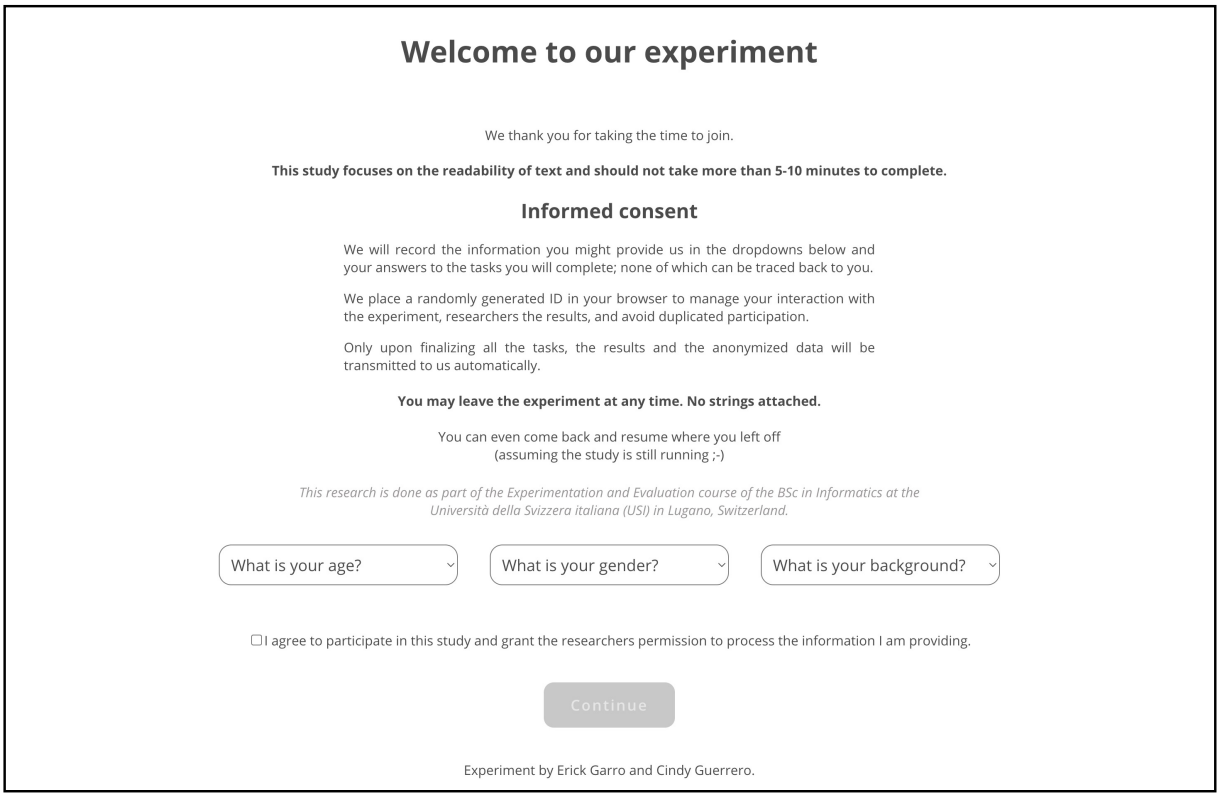
[16] Wikipedia (29.Apr.2022) "Letter case." https://en.wikipedia.org/wiki/Letter_case, Accessed: May 1, 2022.

[17] Wikipedia (30.Apr.2022) "Capitalization in English." https://en.wikipedia.org/wiki/Capitalization_in_English, Accessed: May 1, 2022.

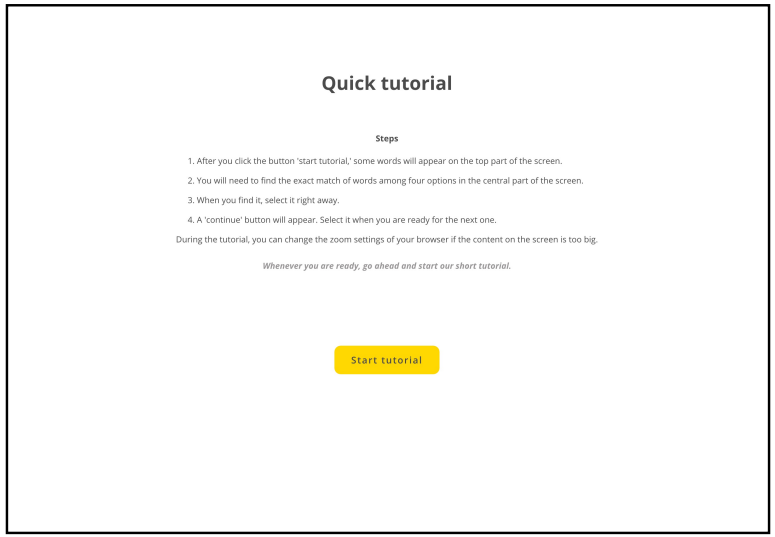
[18] Wikipedia (29.Apr.2022) "Camel case" https://en.wikipedia.org/wiki/Camel_case#History_of_modern_technical_use, Accessed: May 1, 2022.

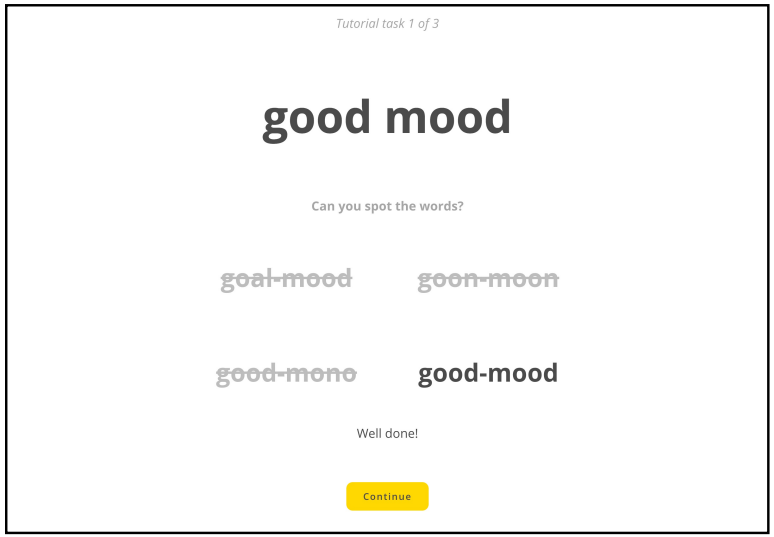
6 Appendix

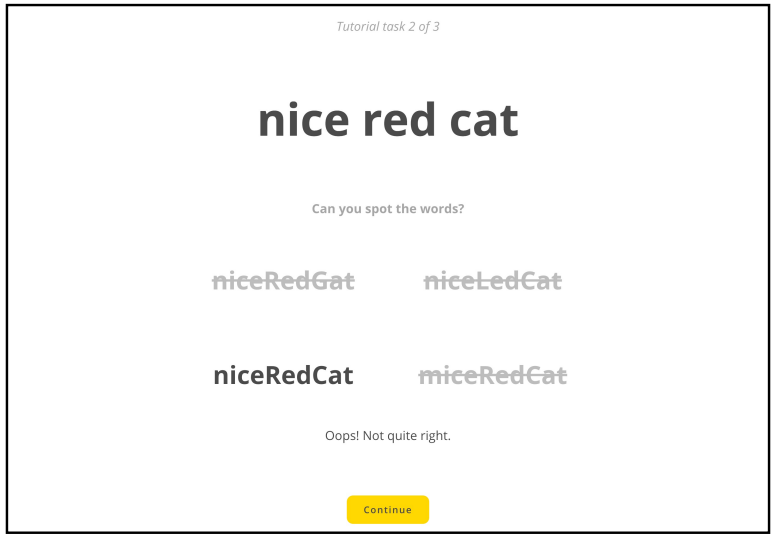
6.1 Web application: Consent & Tutorial

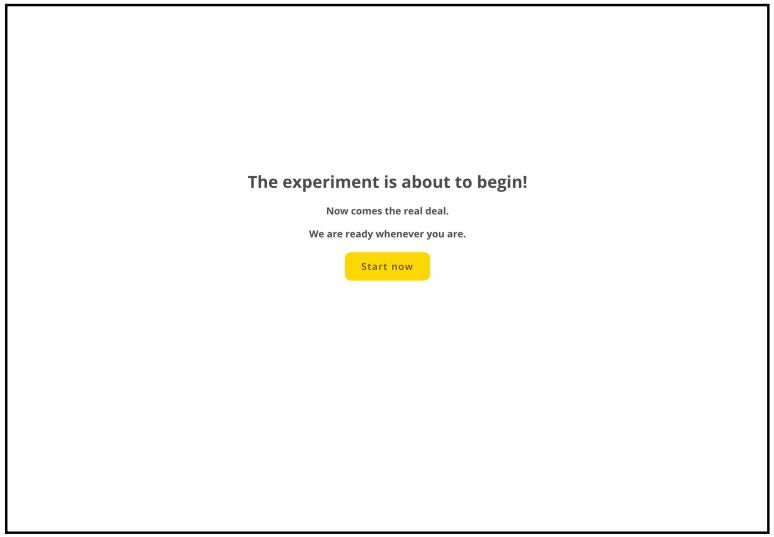


Web application: Consent & Tutorial

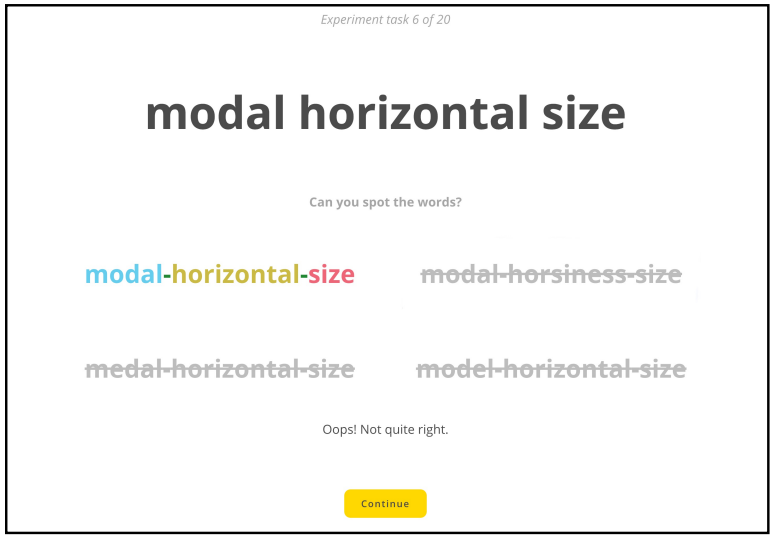


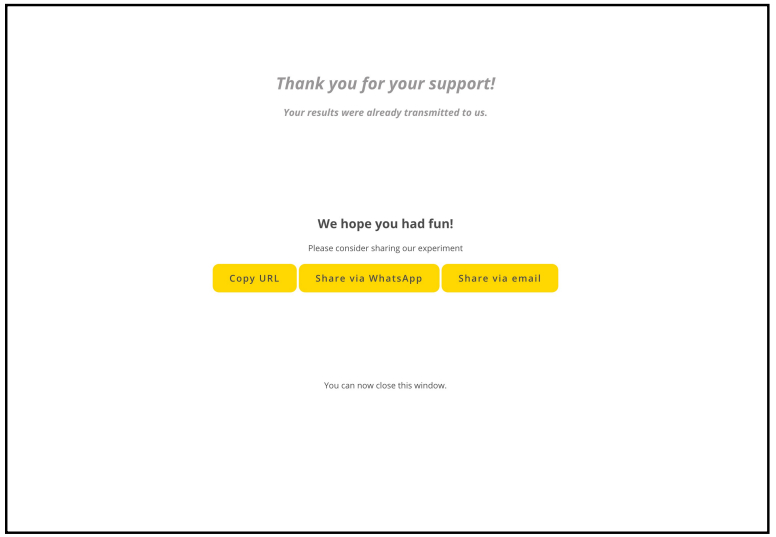






Web application: Consent & Tutorial





6.2 Questionnaire

№	Question	Correct	Incorrect	Incorrect	Incorrect
1	telotaxi is sour	telotaxi-is-sour	tropotaxi-is-sour	telotaxi-if-soup	telophase-is-sour
2	vinyl paint	vinyl-paint	vinyl-plant	vinyl-taint	vinyl-saint
3	handy dandy	handy-dandy	handy-daddy	handy-sandy	candy-cindy
4	file not found	file-not-found	fine-not-found	file-hot-found	file-not-sound
5	brew coffee now	brew-coffee-now	brew-coffee-meow	draw-coffee-now	brew-toffee-now
6	asphyxy cloud	asphyxy-cloud	cataplexy-cloud	apoplexy-loud	apoplexy-cloud
7	narrow polyphone	narrow-polyphone	narrow-polyphase	marrons-polyphagia	marrows-polyphone
8	dark side guy	dark-side-guy	bark-side-gui	dark-sine-buy	dart-side-guy
9	biotron mnemonic	biotron-mnemonic	biotron-anemone	biotin-mnemonic	biotrin-anemone
10	modal horizontal size	modal-horizontal-size	model-horizontal-size	modal-horsiness-size	medal-horizontal-size
11	sad user	sadUser	sodUser	sadSuer	setUser
12	come closer	comeCloser	coneCloser	comeClover	cameCloser
13	rainbow kid	rainbowKid	rainingKid	rainbowKit	raymondKif
14	has golden pass	hasGoldenPass	hatesGoldPass	hasMoltenPass	hasGoldenPath
15	megatron movies	megatronMovies	negatronMovies	megatonsMovers	megatonMavies
16	eudemons combo	eudemonsCombo	eudaemonsCompo	eudemonsComae	eudaemonsCombo
17	electron shower night	electronShowerNight	electricShowerNighs	electrodeShowerNight	electronShowelNight
18	computerphobe user	computerphobeUser	computernikUser	computerphobeUses	computerphobedUsed
19	quantum daemon	quantumDaemon	quantumDamian	quantongDenim	quantingDaemon
20	has not paid	hasNotPaid	hasNeverPaid	hasNowPaid	hasNoPath

Table 5: Questionnaire The questionnaire made to participants included twenty different questions randomly displayed. The participant must click one in four options to access the following question. Each question is composed by two to three words with black font (monochromatic) or colored font (chromatic). Question 1-5, include kebab case monochromatic. Question 6-10, include kebab case chromatic. Question 11-15, include camel case monochromatic. Question 16-20, include camel case chromatic. Chromatic colors contain color-blind friendly colors.

6.3 Statistics: Accuracy & time responses for Kebab case vs Camel case, general approach.

Measure 1		Measure 2	t	df	p	Mean Difference	SE Difference	Cohen's d
accu_k	-	accu_c	−0.776	68	0.441	−1.014	1.308	−0.093
accu_km	-	accu_kc	0.815	68	0.418	1.739	2.135	0.098
accu_cm	-	accu_cc	1.241	68	0.219	2.609	2.102	0.149
time_k	-	time_c	1.725	68	0.089	289.232	167.711	0.208
time_km	-	time_kc	−1.604	68	0.113	−830.957	517.982	−0.193
time_cm	-	time_cc	−5.641	68	*** <.001	−1510.986	267.849	−0.679

Table 6: Paired Samples T-Test for kebab case and camel case responses. Responses are measured as 1) response accuracy (%) and 2) median response time (ms). Each participant answered aleatory twenty questions total (n=69 participants. Median response was considered per group. Two major categories kebab ('k', n=10 responses) and camel ('c', n=10 responses). Notice that only the response time of camel case when compared monochromatic vs chromatic was significatively different. ***, p<0.001 by paired sample Student t-test. Abbreviations: accu: Accuracy; c: camel case; cm: camel case monochromatic; cc: camel case chromatic k: kebab case; kc: kebab case chromatic; km: kebab case monochromatic; time: median time

	N	Mean	SD	SE
accu_k	69	90.725	14.884	1.792
accu_c	69	91.739	16.175	1.947
accu_km	69	91.594	16.945	2.040
accu_kc	69	89.855	17.698	2.131
accu_cm	69	93.043	17.090	2.057
accu_cc	69	90.435	19.587	2.358
time_k	69	5390.768	2977.232	358.417
time_c	69	5101.536	2846.135	342.634
time_km	69	5180.899	3959.304	476.644
time_kc	69	6011.855	4027.418	484.844
time_cm	69	4350.246	2082.372	250.688
time_cc	69	5861.232	3410.172	410.536

Table 7: General responses for kebab case and camel case. Responses are measured as 1) the mean response accuracy (%, n=10 responses) and 2) median response time (ms, n=10 responses). Abbreviations: accu: Accuracy; c: camel case; cm: camel case monochromatic; cc: camel case chromatic k: kebab case; kc: kebab case chromatic; km: kebab case monochromatic; SD: Standard deviation; SE: Standard deviation error; time: median time

	t	df	p	Mean Difference	Cohen's d
accu_k	50.633	68	$< .001$	90.725	6.096
accu_c	47.112	68	$< .001$	91.739	5.672
accu_km	44.901	68	$< .001$	91.594	5.405
accu_kc	42.174	68	$< .001$	89.855	5.077
accu_cm	45.224	68	$< .001$	93.043	5.444
accu_cc	38.353	68	$< .001$	90.435	4.617
time_k	15.041	68	$< .001$	5390.768	1.811
time_c	14.889	68	$< .001$	5101.536	1.792
time_km	10.870	68	$< .001$	5180.899	1.309
time_kc	12.400	68	$< .001$	6011.855	1.493
time_cm	17.353	68	$< .001$	4350.246	2.089
time_cc	14.277	68	$< .001$	5861.232	1.719

Table 8: One Sample T-Test for kebab case and camel case accuracy and time responses. Notice that each group has significantly dispersed responses p<0.001 by one sample Student t-test. Abbreviations: accu: response accuracy; c: camel case; cm: camel case monochromatic; cc: camel case chromatic k: kebab case; kc: kebab case chromatic; km: kebab case monochromatic; time: response time
Note$_1$:For the Student t-test, the alternative hypothesis specifies that the mean is different from null-hypothesis (H_0).
Note$_2$: For the Student t-test, effect size is given by Cohen's *d*.
Note$_3$: For the Student t-test, location difference estimate is given by the sample mean difference *d*.

6.4 Statistics: Accuracy responses per educational background for Kebab case vs Camel case

Measure 1		Measure 2	t	df	p	Mean Difference	SE Difference	Cohen's d
accu_k_cs	-	accu_k_hc	0.189	17	0.852	1.111	5.879	0.045
accu_k_cs	-	accu_k_of	−1.881	23	0.073	−6.667	3.544	−0.384
accu_k_hc	-	accu_kc_of	−1.589	17	0.131	−8.889	5.595	−0.374
accu_c_cs	-	accu_c_hc	−0.210	17	0.836	−1.111	5.295	−0.049
accu_c_cs	-	accu_c_of	−1.964	23	0.062	−7.500	3.819	−0.401
accu_c_hc	-	accu_c_of	−1.498	17	0.153	−8.889	5.935	−0.353
accu_k_cs	-	accu_c_cs	−0.182	23	0.857	−0.417	2.290	−0.037
accu_k_hc	-	accu_c_hc	−0.546	17	0.592	−1.667	3.052	−0.129
accu_k_of	-	accu_c_of	−0.214	24	0.832	−0.400	1.869	−0.043
accu_km_cs	-	accu_km_hc	0.156	17	0.878	1.111	7.135	0.037
accu_km_cs	-	accu_km_of	−1.000	23	0.328	−4.167	4.167	−0.204
accu_km_hc	-	accu_km_of	−1.102	17	0.286	−6.667	6.050	−0.260
accu_cm_cs	-	accu_cm_hc	0.000	17	1.000	0.000	6.262	0.000
accu_cm_cs	-	accu_cm_of	−1.366	23	0.185	−5.000	3.661	−0.279
accu_cm_hc	-	accu_cm_of	−0.972	17	0.345	−6.667	6.860	−0.229
accu_km_cs	-	accu_cm_cs	−0.272	23	0.788	−0.833	3.064	−0.056
accu_km_hc	-	accu_cm_hc	−0.697	17	0.495	−2.222	3.189	−0.164
accu_km_of	-	accu_cm_of	−0.296	24	0.770	−0.800	2.703	−0.059
accu_kc_cs	-	accu_kc_hc	0.160	17	0.875	1.111	6.949	0.038
accu_kc_cs	-	accu_kc_of	−1.905	23	0.069	−9.167	4.812	−0.389
accu_kc_hc	-	accu_kc_of	−1.699	17	0.108	−10.000	5.886	−0.400
accu_cc_cs	-	accu_cc_hc	−0.416	17	0.682	−2.222	5.336	−0.098
accu_cc_cs	-	accu_cc_of	−2.015	23	0.056	−10.000	4.964	−0.411
accu_cc_hc	-	accu_cc_of	−1.656	17	0.116	−11.111	6.710	−0.390
accu_kc_cs	-	accu_cc_cs	0.000	23	1.000	0.000	3.807	0.000
accu_kc_hc	-	accu_cc_hc	−0.251	17	0.805	−1.111	4.420	−0.059
accu_kc_of	-	accu_cc_of	0.000	24	1.000	0.000	2.582	0.000

Table 9: Paired Samples T-Test for kebab case and camel case responses based on educational background. Accuracy responses corresponds to the percentage-ratio correct:total. Participants were divided in groups based on educational background: Computer Science (cs, n=24 participants), High School (hc, n=18 participants), Other field than Computer Science (of, n=25 participants). There is no significant difference between groups $p>0.05$ by paired sample Student t-test. Abbreviations: accu: Accuracy; c: camel case; cm: camel case monochromatic; cc: camel case chromatic k: kebab case; kc: kebab case chromatic; km: kebab case monochromatic; time: median time

	N	Mean	SD	SE
accu_k_cs	24	88.750	16.235	3.314
accu_k_hc	18	86.667	20.864	4.918
accu_k_of	25	95.600	5.066	1.013
accu_kc_of	25	96.000	8.165	1.633
accu_c_cs	24	89.167	16.918	3.453
accu_c_hc	18	88.333	23.073	5.438
accu_c_of	25	96.000	7.071	1.414
accu_km_cs	24	90.833	17.673	3.607
accu_km_hc	18	87.778	23.901	5.633
accu_km_of	25	95.200	8.718	1.744
accu_cm_cs	24	91.667	15.511	3.166
accu_cm_hc	18	90.000	26.789	6.314
accu_cm_of	25	96.000	8.165	1.633
accu_kc_cs	24	86.667	20.144	4.112
accu_kc_hc	18	85.556	22.550	5.315
accu_cc_cs	24	86.667	20.990	4.285
accu_cc_hc	18	86.667	25.668	6.050
accu_cc_of	25	96.000	11.547	2.309

Table 10: Responses for kebab case and camel case by educational background. Responses are measured as the mean response accuracy (%, n=10 responses). Participants were divided in groups based on educational background: Computer Science (cs, n=24 participants), High School (hc, n=18 participants), Other field than Computer Science (of, n=25 participants). Abbreviations: accu: Accuracy; c: camel case; cm: camel case monochromatic; cc: camel case chromatic k: kebab case; kc: kebab case chromatic; km: kebab case monochromatic; SD: Standard deviation; SE: Standard deviation error

6.5 Statistics: Time responses per educational background for Kebab case vs Camel case

Measure 1		Measure 2	t	df	p	Mean Difference	SE Difference	Cohen's d
time_k_cs	-	time_k_hc	−2.950	17	**0.009	−2430.722	823.973	−0.695
time_k_cs	-	time_k_of	−1.819	23	0.082	−1301.563	715.389	−0.371
time_k_hc	-	time_k_of	0.968	17	0.347	1012.250	1045.814	0.228
time_c_cs	-	time_c_hc	−2.030	17	0.058	−1396.722	687.917	−0.479
time_c_cs	-	time_c_of	−1.653	23	0.112	−1149.271	695.338	−0.337
time_c_hc	-	time_c_of	0.017	17	0.987	15.167	899.160	0.004
time_k_cs	-	time_c_cs	−0.063	23	0.950	−13.000	205.784	−0.013
time_k_hc	-	time_c_hc	2.028	17	0.059	997.194	491.702	0.478
time_k_of	-	time_c_of	0.931	24	0.361	169.580	182.077	0.186
time_km_cs	-	time_km_hc	−2.090	17	0.052	−3130.111	1497.794	−0.493
time_km_cs	-	time_km_of	−1.009	23	0.323	−688.333	682.118	−0.206
time_km_hc	-	time_km_of	1.320	17	0.204	2059.111	1560.129	0.311
time_cm_cs	-	time_cm_hc	−3.100	17	**0.007	−1684.556	543.483	−0.731
time_cm_cs	-	time_cm_of	−1.923	23	0.067	−756.667	393.474	−0.393
time_cm_hc	-	time_cm_of	1.282	17	0.217	794.944	620.137	0.302
time_km_cs	-	time_cm_cs	2.351	23	*0.028	515.917	219.469	0.480
time_km_hc	-	time_cm_hc	1.557	17	0.138	1898.167	1219.092	0.367
time_km_of	-	time_cm_of	1.327	24	0.197	447.200	336.899	0.265
time_kc_cs	-	time_kc_hc	−2.412	17	0.027	−2549.556	1057.046	−0.569
time_kc_cs	-	time_kc_of	−1.928	23	0.066	−2194.583	1138.531	−0.393
time_kc_hc	-	time_kc_of	0.770	17	0.452	982.278	1274.976	0.182
time_cc_cs	-	time_cc_hc	−1.064	17	0.302	−968.000	910.038	−0.251
time_cc_cs	-	time_cc_of	−1.493	23	0.149	−1250.917	837.984	−0.305
time_cc_hc	-	time_cc_of	−0.712	17	0.486	−717.556	1008.096	−0.168
time_kc_cs	-	time_cc_cs	−1.968	23	0.061	−626.458	318.244	−0.402
time_kc_hc	-	time_cc_hc	1.914	17	0.073	1162.167	607.330	0.451
time_kc_of	-	time_cc_of	0.502	24	0.620	340.080	677.684	0.100

Table 11: Paired Samples T-Test for kebab case and camel case time for responses based on educational background. Time responses corresponds to the total time (in miliseconds, ms) a participant takes to read a question and select one in four options. Participants were divided in groups based on educational background: Computer Science (cs, n=24 participants), High School (hc, n=18 participants), Other field than Computer Science (of, n=25 participants). Notice that response time of participants in k_cs vs k_hc, cm_cs vs cm_hc, and km_cs vs cm_cs represent a significant difference between each other, respectively. *, $p<0.05$ **, $p<0.01$ by paired sample Student t-test. Abbreviations: accu: Accuracy; c: camel case; cm: camel case monochromatic; cc: camel case chromatic k: kebab case; kc: kebab case chromatic; km: kebab case monochromatic; time: median time teste

	N	Mean	SD	SE
time_k_cs	24	4380.896	2502.510	510.823
time_k_hc	18	6529.278	2973.133	700.774
time_k_of	25	5573.940	3229.928	645.986
time_c_cs	24	4393.896	2909.681	593.936
time_c_hc	18	5532.083	2408.405	567.667
time_c_of	25	5404.360	3098.602	619.720
time_km_cs	24	4247.500	2684.977	548.069
time_km_hc	18	7040.111	6076.065	1432.142
time_km_of	25	4831.000	2725.943	545.189
time_cm_cs	24	3731.583	2094.356	427.509
time_cm_hc	18	5141.944	2411.820	568.471
time_cm_of	25	4383.800	1749.100	349.820
time_kc_cs	24	4617.000	2571.314	524.867
time_kc_hc	18	7027.778	3762.895	886.923
time_kc_of	25	6665.160	5114.396	1022.879
time_cc_cs	24	5243.458	3589.707	732.746
time_cc_hc	18	5865.611	2698.664	636.081
time_cc_of	25	6325.080	3731.097	746.219

Table 12: Time responses for kebab case and camel case by educational background. Time responses corresponds to the total time (in miliseconds, ms) a participant takes to read a question and select one in four options. Participants were divided in groups based on educational background: Computer Science (cs, n=24 participants), High School (hc, n=18 participants), Other field than Computer Science (of, n=25 participants). Abbreviations: c: camel case; cm: camel case monochromatic; cc: camel case chromatic k: kebab case; kc: kebab case chromatic; km: kebab case monochromatic; SD: Standard deviation; SE: Standard deviation error

6.6 Statistics: Accuracy & time responses dispersion for Kebab case vs Camel case based on academic background

	t	df	p	Mean Difference	Cohen's d
accu_k_cs	26.780	23	< .001	88.750	5.466
accu_c_cs	25.820	23	< .001	89.167	5.270
accu_km_cs	25.180	23	< .001	90.833	5.140
accu_kc_cs	21.077	23	< .001	86.667	4.302
accu_cm_cs	28.953	23	< .001	91.667	5.910
accu_cc_cs	20.228	23	< .001	86.667	4.129
accu_k_hc	17.624	17	< .001	86.667	4.154
accu_c_hc	16.243	17	< .001	88.333	3.828
accu_km_hc	15.582	17	< .001	87.778	3.673
accu_kc_hc	16.097	17	< .001	85.556	3.794
accu_cm_hc	14.254	17	< .001	90.000	3.360
accu_cc_hc	14.325	17	< .001	86.667	3.377
accu_k_of	94.350	24	< .001	95.600	18.870
accu_c_of	67.882	24	< .001	96.000	13.576
accu_km_of	54.601	24	< .001	95.200	10.920
accu_kc_of	58.788	24	< .001	96.000	11.758
accu_cm_of	58.788	24	< .001	96.000	11.758
accu_cc_of	41.569	24	< .001	96.000	8.314
time_k_cs	8.576	23	< .001	4380.896	1.751
time_c_cs	7.398	23	< .001	4393.896	1.510
time_km_cs	7.750	23	< .001	4247.500	1.582
time_kc_cs	8.797	23	< .001	4617.000	1.796
time_cm_cs	8.729	23	< .001	3731.583	1.782
time_cc_cs	7.156	23	< .001	5243.458	1.461
time_k_hc	9.317	17	< .001	6529.278	2.196
time_c_hc	9.745	17	< .001	5532.083	2.297
time_km_hc	4.916	17	< .001	7040.111	1.159
time_kc_hc	7.924	17	< .001	7027.778	1.868
time_cm_hc	9.045	17	< .001	5141.944	2.132
time_cc_hc	9.221	17	< .001	5865.611	2.174
time_k_of	8.629	24	< .001	5573.940	1.726
time_c_of	8.721	24	< .001	5404.360	1.744
time_km_of	8.861	24	< .001	4831.000	1.772
time_kc_of	6.516	24	< .001	6665.160	1.303
time_cm_of	12.532	24	< .001	4383.800	2.506
time_cc_of	8.476	24	< .001	6325.080	1.695

Table 13: One Sample T-Test for kebab case and camel case accuracy and time responses based on educational background. Participants were divided in groups based on educational background: Computer Science (cs, n=24 participants), High School (hc, n=18 participants), Other field than Computer Science (of, n=25 participants). Notice that each group has significantly dispersed responses $p<0.001$ by one sample Student t-test. Abbreviations: accu: response accuracy; c: camel case; cm: camel case monochromatic; cc: camel case chromatic k: kebab case; kc: kebab case chromatic; km: kebab case monochromatic; time: response time

Note$_1$:For the Student t-test, the alternative hypothesis specifies that the mean is different from null-hypothesis (H_0).

Note$_2$: For the Student t-test, effect size is given by Cohen's *d*.

Note$_3$: For the Student t-test, location difference estimate is given by the sample mean difference *d*.