# The importance of nothingness - a qualitative interview study with programmers

Hanna Oberg
Graduation Thesis, September 2020
Media Technology MSc program, Leiden University
Supervisors: Rob Saunders and Felienne Hermans
hanna.em.oberg@gmail.com

## Abstract

Not everyone is a designer but many people work closely with designers or contribute to the process of producing things that are professionally designed. Programmers are such a group of people. Another thing that designers and programmers have in common is the importance they place on layout for the sake of legibility. This study aims to investigate the opinions and ideas that programmers have concerning the topic of white space; one of the key building blocks of a layout. Interviews were conducted with six European male programmers about their thoughts on white space, their habits dealing with it, and the similarities and differences between code and visual layout.

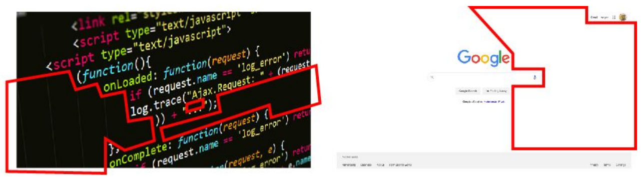
Two main attitudes were identified during the interviews. First, the difference of thought of consistency of white space in code and in visual layouts. In code white space is supposed to be used in a uniform way consistently throughout a document while in a visual layout white space has its strength in creating focus by having more or less of it present. The second difference is between how they were described to give value. In code the focus was explained to be more utility based and in a visual layout more cognitive benefits were detailed. For being a concept rarely thought of by programmers it is explained to be one of the pillars of coding practices and with all the benefits it gives makes a heightened focus on it valuable.

## Keywords

White space, negative space, indentation, programming code, programmers

## 1 INTRODUCTION

White space or as some might call it, negative space, is an area that is left empty (see Figure 1). White space has a special importance in visual design but exists in many other areas with different uses and values. Programmers utilize white space regularly in programming and some are even exposed to it while collaborating with designers, but what are their thoughts and ideas of this concept? In this paper we aim to look closer at white space from the viewpoint of a programmer.



```
      <link rel=
       <script type="text/javascript
    <script type="text/javascript">
      (function(){
          onLoaded: function(request) {
            if (request.name == 'log_error') retu
              log.trace("Ajax.Request: " + (request
                )) + "...");
      },  onComplete: function(request) {
            if (request.name == 'log_error') retu
```

Figure 1.Areas of white space found in code and on a website

### 1.1 What is white space?

White space is used in design to guide a viewer/user through content. It can create emphasis, hierarchy, and relieve visual stress otherwise caused by a filled design space. Imagine moving through a loud and noisy area, white space works as silence in such a space. It creates relief and a way to breathe. White space is used for this purpose in design across areas; interface design, advertising, art, and more.

There are three key historical moments for the white space concept to gain notoriety. First comes the minimalist movement when a new appreciation for selectiveness and minimalism found its way into advertising in the mid-1960s [1] and has carried on to this day in every field of design. The second important movement was the elevation of corporate design in the

1950s. Suddenly good design was seen as a business strength and the corporate profile shown to the outside world was meticulously crafted. White space was carefully considered when shaping the art style of a company to convey the right message. Minimalism in logotypes and interfaces were made popular with Paul Rand being one of the front runners with his minimalist IBM logo (See Figure 2).

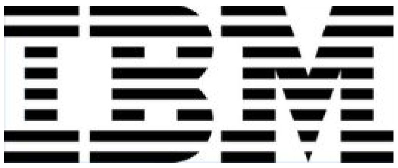


Figure 2. 8-bar IBM logo by Paul Rand [2]

The third historical moment for white space and minimalism is the use of it in architecture and upscale living. The concept of *Less is More* was found in furniture, architecture, and interior design among richer consumers. The white space taking place in the open spaces that were preferred in the minimalist living situation. In all these cases white space has shown to add an elevated level of sophistication and value [1], [3]. However, aside from advertising and architecture, white space is being used in new media, especially in interface design where it provides additional benefits [7][8][9].

## 1.2 Visual/Design

Aesthetics was for a long time not considered as a factor for good HCI. However, since the 1980s a small but growing focus of the importance of aesthetics for HCI has been initiated by Gait [4]. More recent studies have shown that aesthetics can raise the perceived usability of a platform [5]. Aesthetics is one of a few new areas of research where HCI is trying to find more value among others such as; effect, emotions, and trust [6]. A study on the accessibility of websites and the correlation with aesthetics concluded that the biggest factor of accessibility when it comes to visual aesthetics is the level of layout cleanliness that is experienced [5]. With the opposite of cleanliness being regarded as a cluttered interface, we come back to the topic of white space. White space is used on the web to let a layout "breathe" and create a focus for the user, two things that are especially important when competing for attention [A]. On a micro-level white space is being carefully considered in type to provide maximum legibility when applied between letters and words, and in the text as a whole to not create visible rivers through a text [7]. White space takes at least six shapes: margins, the area around headings and their distance to paragraphs, space at end of lines, leading, tracking, and around images and other elements [10].

Programmers have access to a lot of guidance when it comes to creating UI layouts but in proper HCI fashion, it is heavily focused on usability and user experience with little focus on aesthetics. With aesthetics being a crucial part of the perceived usability of an interface, this is crippling programmers to carry out tasks and projects on their own. The web provides plenty of tools for programmers in the shape of style frameworks and color tools, which allows for the simple creation of aesthetically pleasing artifacts of web design. However, one thing that is difficult to automate is the concept of white space and how it should be applied to a page. Design frameworks usually supply a grid-based system to structure a page but this alone does not guarantee an appropriate use of white space.

Several studies have taken very direct approaches to web layouts having users judge layouts with different amounts of white space [6][8][9]. It has proven that satisfaction levels are higher when white space is given ample room as opposed to layouts where content is placed tighter together.

The benefits of using white space in designs are many: appropriate use of white space facilitates contrast [11], simplicity, and balance in a document. White space can create tension between two design elements. Empty space provides resting points within a page that may facilitate deeper processing. As with all elements of a design white space should be treated with as much thought and care as any other part [10].

This study will look closer at how programmers perceive white space, in both their own environments as in visual layouts. Understanding their point of view is a first step in understanding how white space takes shape in other areas than just design. The study described in this paper has attempted to address the research question: *how do programmers' opinions and ideas of* *white space in code differ compared to white space in a visual layout?*

## 1.3 The Perception of White Space

Viewers of a piece that includes white space generally perceive the occupied area to be larger than it actually is. The take away from this is that to achieve focus or dominance in a layout or space it is not about size but achieving a balance between occupied and unoccupied space [12].

The Rubin Vase (see Figure 3) is a good example of how we perceive negative space through figure-ground perception. When viewing an image our mind will try to separate a dominant shape from the background. What a Rubin Vase does is that it makes a shape both from the foreground and the background and your mind has a possibility to favor either one or the other. The manipulation of white space in those instances can dictate which part of the image will be the object and what will be the background/ground [13]. The more white space present, the more emphasis is placed on the occupied space. (positive and negative space)

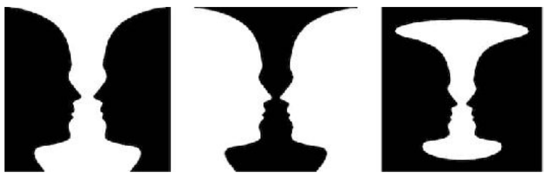


Figure 3. The Rubin Vase

Another cognitive aspect of white space can be found in Gestalt theory which is a school of psychology concerning how we interpret patterns in visual objects. According to Gestalt theories of perception, the appropriate use of white space leads to disparate entities to be interpreted as a whole by the brain. Consequently, white space is heavily applied in visual design to help the viewer make sense of what they are seeing by supporting the perception of groups. White space in this plays an important role not only for the layout of a visual image, but also to give space for eyes to rest, signal break/end of a communication, and variation of space (passive/dynamic).

## 2 METHODOLOGY

### 2.1 Aim of the study

The aim of this study is to investigate the thoughts and ideas that programmers have about white space as it is used in files of code and online in design. Little is written on the topic of spacing in code in comparison to the wealth of information there is about the impact of white space in design and art. As we expand this area of interest we get insights from the people using this sort of white space in their programming tasks on an everyday basis.

### 2.2. Research method

The study was based on interviews with six professional programmers. Questions involved opinions on white space in code, white space compared to other components, and white space in a sample layout. Interviews were not held face-to-face due to precautionary measures due to the spread of the novel coronavirus and instead were conducted over VoIP calls. All interviews were recorded and transcribed by hand. The interviews were semi-structured interviews and centered around an interview preparation document that was sent out a couple of days in advance to each participant. The document outlined some helpful vocabulary and their definitions as well as provided a few pieces of code in different languages and a visual layout in three different versions. The participants were interviewed about their opinions on the white space in the examples provided in the document as well as experiences from their own lives.

### 2.3 Participants

All participants had more than seven years of working experience. The criteria for participants were to have at least 3 years of work experience, actively writing code in their daily life, and not working in a role that included visual design tasks. Participants were sourced based on their educational background in computer engineering and by referral from previous interviewees. The number of participants was decided based on the saturation level of the data collected. Once no new unique data was provided in an interview it was considered saturated. Four interviews were held in English and two interviews were held in Swedish and translated to English before analysis.

## 2.4 Data analysis

An inductive approach was taken to go about this research as there are not many studies on the topic done previously. We aim to create new theories in an unexplored area and will hence conduct a Grounded Theory study based on Glaser and Strauss' [14] approach. The data collected in the interviews were transcribed and were separated into sentences or paragraphs of topics with some data left out due to not being relevant to the study. With the data formatted the coding of this data went through three steps: open coding, axial coding, and selective coding. The open coding step involved creating as many code labels or short sentences from each piece of the formatted data. The axial coding included finding common categories within the codes created and to sort them into a framework of high-level categories consisting of: The phenomenon studied, causal conditions, intervening conditions, context, strategies, and consequences. From the axial coding, the selective coding step included finding relations between the identified categories.

| Respondent | Title | Years of experience |
|---|---|---|
| R1 | Full Stack Developer | 9 |
| R2 | Senior Software Engineering Manager | 12 |
| R3 | Full Stack Developer | 23 |
| R4 | Full Stack Developer | 7 |
| R5 | Lead Developer PHP | 7 |
| R6 | Full Stack Developer | 8 |

# 3 RESULTS

In the discussion of white space in code all respondents had a very pragmatic view of the reasoning to why white space exists. And although the majority of participants claimed to never have thought of white space in code in the past they agree that white space is a crucial aspect of coding when interviewed about the topic. White space was first and foremost identified as utilitarian in coding. White space in layouts were described to have more cognitive benefits.
There were also recurring comments about the visual differences in white space between the two mediums. After processing the data and applying the three steps of coding two main categories emerged, one structural and one functional dimension:

- Consistent and variable structure
- Utility and cognitive benefits

## 3.1 Consistent and variable structure

The first emerging view among the respondents of white space in the contexts of code and visual layout was the aspect of consistency of use in code. On the opposite hand in a visual layout white space was explained to be used in an inconsistent manner to create focus and relations. R4 notes this about white space between operators: *The white space in between for example operators I think of a lot. You want it to be the same everywhere. space, equal sign, space, next. It's a little bit of an OCD with all of that to get it to look exactly the same everywhere.* Using white space in the same way consistently in every instance of coding was explained by the respondents to have the benefit of ease of use. As R4 explains: *the goal is that you shouldn't be able to tell who has written the code. You should be able to pick up a piece of code and it should look the same as if you wrote it. It turns into company specific standards.* This notion of that, you shouldn't be able to see who wrote the code, is important for the majority of the respondents. They explain that uniformity in a code file and amongst programmers working on the code collaboratively makes their work flow better and be more effective. All of them expressed that knowing how to properly and consistently write white space is a must for an experienced programmer. As R3 puts it: *How you use white space is a proof of knowledge.* Having a uniform distribution of white space can be seen as a requirement for several reasons. A project manager with a background in programming and deals with code on a daily basis explains the following about the readability that comes with consistency: *I can't imagine anyone that would shoot themselves in the foot* *saying "ok, every time I'm just gonna change this stuff around".*

In contrast to the importance placed on consistency in the use of white space in programming, the respondents were clear in articulating the necessity for variability in visual layout. Although the same words are used to describe the purpose of white space such as *providing focus* the white space distribution must be uneven to direct focus efficiently. R4 mentions that if there is too much white space *Everything gets the same importance*. While the aim of white space in code is to be as uniform as possible to remove style differences that could point to a certain programmer, white space in layouts is explained to be used as a way to differentiate yourself. How you work with white space and where you choose to utilize it is explained as a possibility to set a visual identity apart from others just as graphical elements would. R6 had this to say about an apple.com website that was used as an example: *I like this layout (with a certain amount of white space), maybe possibly because it feels more like their design I think. A bit more like Apple in general.*

In relation to the comment on proof of knowledge in how white space is used in code it has been described by the respondents that if less white space is used (to a point) the more advanced the programmer is. An abundance of white space in code is a sign of someone that lacks experience and needs the aid that white space provides. However, in a visual layout with a variable amount of white space there is no right or wrong in terms of amount to hint at the qualifications of the designer. Instead the respondents claimed to have a certain intuition about when there is too much white space used and as R5 says, would look for what might be missing: *Sometimes when you have too much white space, it could be that you have an ad blocker or some functionality was broken. I get this impression that something is broken or maybe they are doing some AB testing. It is not normal.*

## 3.2 Utility and cognitive benefits

Hearing programmers talk about white space, there was an additional difference that strongly made itself present in the wordings used by all participants. The words used to describe white space in code was strongly focused on utility such as; readability, comprehension, guides, and finding belonging. This is in contrast to how white space in layout was described with more cognitive benefits such as; area to relax, mental break, and less cognitive load. These differences were also explained by the respondents. R5 says; *I don't think so, because in coding it is mainly for readability, for our readability and here (in the visual layout) it's for the gain and manage focus of the client.* When white space in code and in a visual layout was prepared the respondents were comparing the utility of white space in code against the benefits white space gives in a visual layout. One of the most important uses of white space in code was explained to be indentation. *Most important type. Then I must say indentation. That must be the most important. Then the rest is just icing on the cake* Says R4. This importance comes from the general idea of the respondents is that you can get by without vertical white space, but indentation is crucial to be able to work with code. The benefits to utility that indentation gives was overwhelmingly unanimously explained to be readability. R2 says: *It really provides you so much readability if you do this (white space) right.* Among others he is backed up by R5: *It gives us readability, human readability. Because when I read code, if I have indentation, my mind can understand easily when we enter into sub functionality or a sub algorithm. So we know what was done before, and now we enter the sub process. And so we focus on that process before going back up.* The notion of *human readability* was an idea expressed by most programmers as white space in code does not do anything for the communication with the computer, it is only an aid for humans and will be removed once the process of coding is over.

White spaces in the visual layout were described in more broad terms to provide clear cognitive benefits. The focus aspect was mentioned by a majority of respondents. R3 explains: *I think it makes the customer focus on one thing rather than multiple tags and images and you don't know where to look. I think the main thing here is to make sure you look at one thing at a time and to really focus on every different element.* Aside from focus you can take mental breaks as R6 explains it: *You can take a mental break, stay on the picture, and then you can continue with the text. Then* *we can take another break and then the next picture and continue with the next text.*

The points of focus and emphasis is reflected in how the respondents consider where white space's importance in a visual layout lies. Rather than the horizontal white space, that is explained to be of little importance in visual layouts, it is the vertical white space that has the most impact. Its ability to separate content is explained to be its strongest asset. R2 explains: *So you have a fair bit of space in between each one of these things, almost giving people the area to relax their eyes and make sure that they are deep-diving into the component that they are looking into as opposed to super compact, full of information type of stuff.* As the respondents were describing the white space in the visual layouts they were able to take into account all existing white space in comparison to in code where only one respondent recognized all areas of white space.

## 4 CONCLUSION

From the analysis we can draw a few conclusions. The first two are the links between consistency and utility, and variable and cognition respectively. Consistency in how you utilize white space in code is ultimately utility as it makes it easier to navigate, read, and work with the code. Variability and cognition works together as more or less emphasis and focus can be put on objects with the help of carefully distributed white space. Programmers draw these distinctions between white space present in the two mediums both in the way they talk about white space and how they perceive its value. They, in many cases, fail to acknowledge white space that does not provide utility exists, especially in code. They have a more flexible mindset about white space in visual layouts where they freely comment on all white space present. White space is not something that programmers normally think of or consider as a crucial part of programming, however posed the question they can identify clear benefits and values of white space in code.

## 5 DISCUSSION

The outcome that programmers view white space in code as utility can have to do with them utilizing it on a day to day basis for their own benefit. While white space in a visual layout is only something they view subjectively and hence would not associate with utility in the same sense. Most respondents claim to not have thought of white space previously and would if they had it would be related to how it was poorly executed. They expressed dislike both for too much and too little white space in code. Some want less to be able to fit more content in a smaller space which would provide more utility, and others want more to be able to have clear focus. Spacing is something that is supposed to be done right, and when it is, will be forgotten.

The language used by the respondents, when it comes to how they differ the description of white space in code and in visual layouts, had a clear distinction between utility and cognitive benefits although the cognitive benefits can be found in the utility as well. The utility that white space in code provides; ease of orientation, simplifying understanding of context e.g. is all things that makes it cognitively easier for a programmer to understand the code in front of them. Especially in situations where you are confronted with someone else's code. Time spent understanding the code is effectiviced with appropriate use of white space in addition to other tools. So although the two instances were spoken about differently it boiled down to an appreciation of the cognitive benefits that white space gives both code and a visual layout.

The point that white space is only for humans and not for computers could be a reason as to why the respondents do not consider white space as much as potentially a designer would in their line of work. It is also worth noticing that how you space a document has value for some programming languages such as Python. This means that in some instances the computer does place more value on white space but for the respondents, this was not the case. They expressed that ultimately the white space will be removed or reduced to the bare minimum and all the benefits would be gone as well as the code would not be touched by a human programmer again. It turns into a utility in a very literal sense that you use it when you need it, and you remove it when you are done. So even if programmers do not think of or consider white space in their daily work environment it has clearly shown to still play a big part in both their work flow and in their mind. Subconsciously they have a lot of opinions about white space. It is something that, unless it is missing, it is not noticed. Something programmers cannot be without.

## 5.x Threats to validity

The possibility of not being able to have the interviews in person could lead to details and opportunities for further probing into a topic being left out. Qualitative details that come from an in-person interview are ultimately also lost.

The lack of female representation in this study means that valuable opinions were not captured. Differences in design decisions between males and females [15] could imply that there would be another aspect of white space for female programmers.

Ultimately the sample size was determined based on the saturation level of the data collected. However, a higher number of participants could have revealed different ideas and opinions. More respondents could have opened up for a different cultural influence as the majority of the respondents in this study was programmers working in a European work place. Additional insights from programmers dealing with different programming languages could potentially also yield different insights.

## REFERENCES

[1] Pracejus, J. W., Olsen, G. D., & O'Guinn, T. C. (2006). How nothing became something: White space, rhetoric, history, and meaning. Journal of Consumer Research, 33(1), 82-90.

[2] IBM. (2020, September 22). 8-bar. IBM Design Language. https://www.ibm.com/design/language/ibm-logos/8-bar

[3] Cheon, E., & Su, N. M. (2018, April). The Value of Empty Space for Design. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (pp. 1-13).

[4] Gait, J. (1985). An aspect of aesthetics in human-computer communications: Pretty windows. IEEE Transactions on Software Engineering, (8), 714-717.

[5] Mbipom, G., & Harper, S. (2011, October). The interplay between web aesthetics and accessibility. In The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility (pp. 147-154).

[6] Lavie, T., & Tractinsky, N. (2004). Assessing dimensions of perceived visual aesthetics of web sites. International journal of human-computer studies, 60(3), 269-298.

[7] Watzman, S. (2003). Visual design principles for usable interfaces. The human computer interaction handbook, 263-285.

[8] Bernard, M., Chaparro, B., & Thomasson, R. (2000). Finding information on the Web: Does the amount of whitespace really matter. Usability News, 2(1), 1.

[9] Coursaris, C. K., & Kripintris, K. (2012). Web aesthetics and usability: An empirical study of the effects of white space. International Journal of E-Business Research (IJEBR), 8(1), 35-53.

[10] Bradshaw, A. C., & Johari, A. (2002). Effects of white space in learning via the web. Journal of Educational Computing Research, 26(2), 191-201.

[11] Parker, R. C., & Berry, P. (1998). Looking good in print. Ventana.

[12] Tinker, M. A. (1966). Experimental studies on the legibility of print: an annotated bibliography. Reading Research Quarterly, 67-118.

[13] Peterson, M. A., Harvey, E. M., & Weidenbacher, H. J. (1991). Shape recognition contributions to figure-ground reversal: Which route counts?. Journal of Experimental Psychology: Human Perception and Performance, 17(4), 1075.

[14] Glaser, B. G., & Strauss, A. L. (2017). Discovery of grounded theory: Strategies for qualitative research. Routledge.

[15] Moss, G., Gunn, R., & Heller, J. (2006). Some men like it black, some women like it pink: consumer implications of differences in male and female website design. Journal of Consumer behaviour, 5(4), 328-341.

[16] [A] Simon, H. A. (1994). The Bottleneck of. Integrative views of motivation, cognition, and emotion, 41, 1.