Udacity Front End Nanodegree

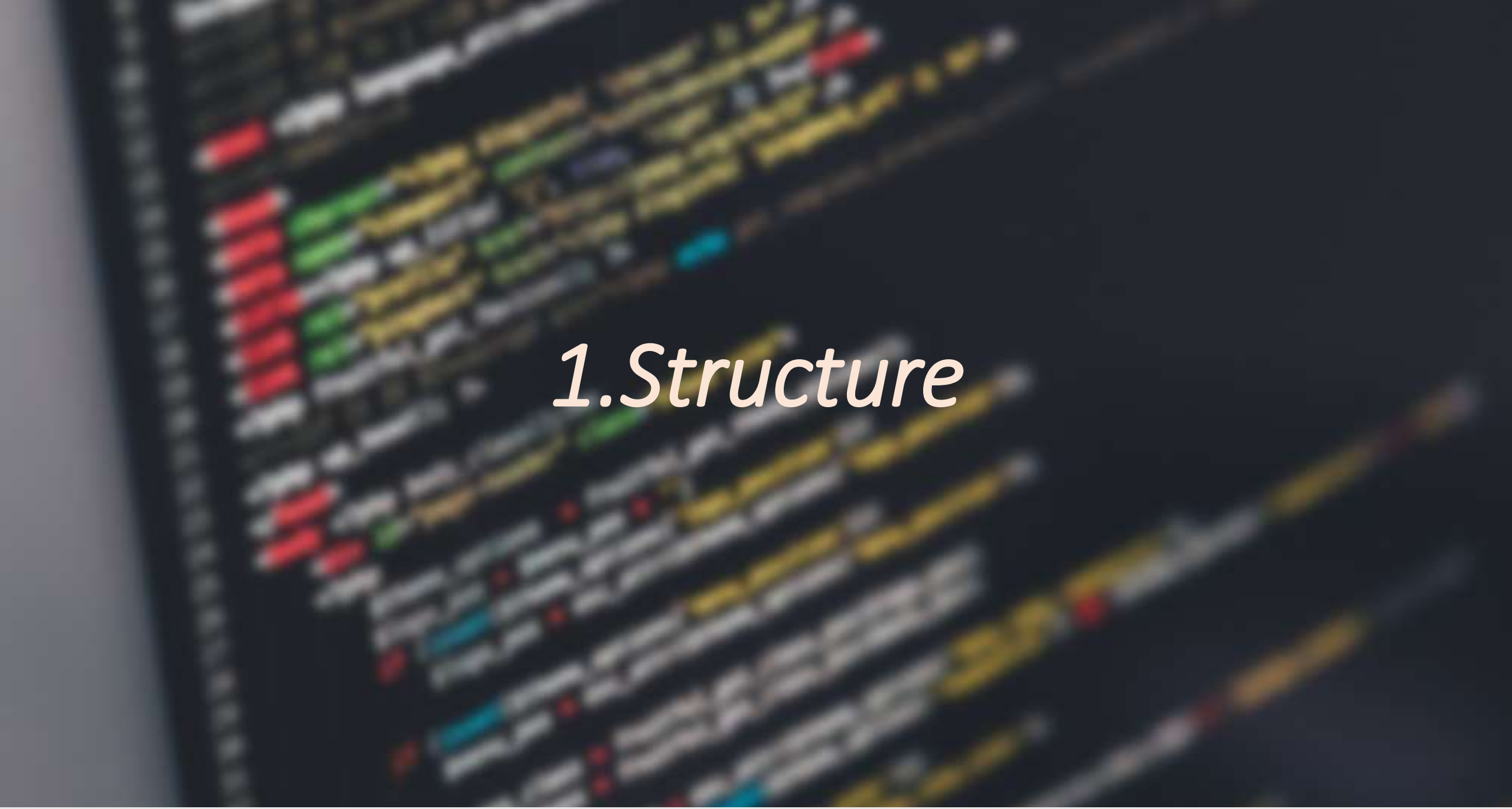
First project: Personal Blog Website

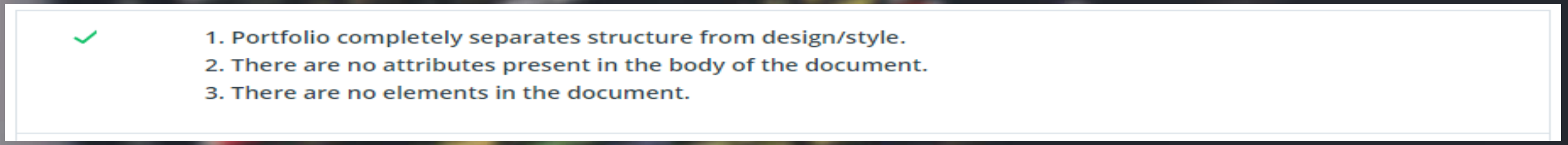
https://www.youtube.com/channel/UC51OIiCEJ1hAvJlpky5cuSA

Submission Evaluation

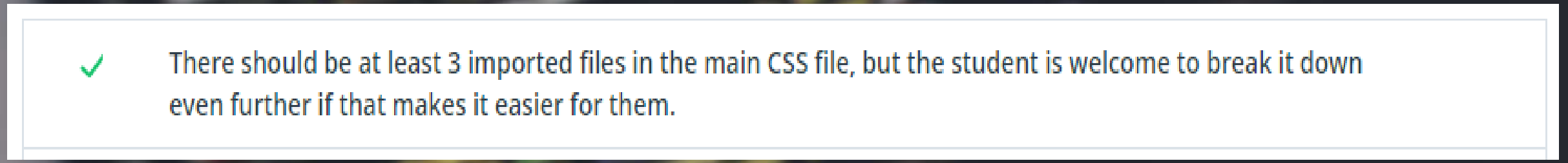
1. *Structure*
2. *Design*
3. *Layout*
4. *Responsiveness*
5. *Quality*

https://www.youtube.com/channel/UC51OIiCEJ1hAvJlpky5cuSA

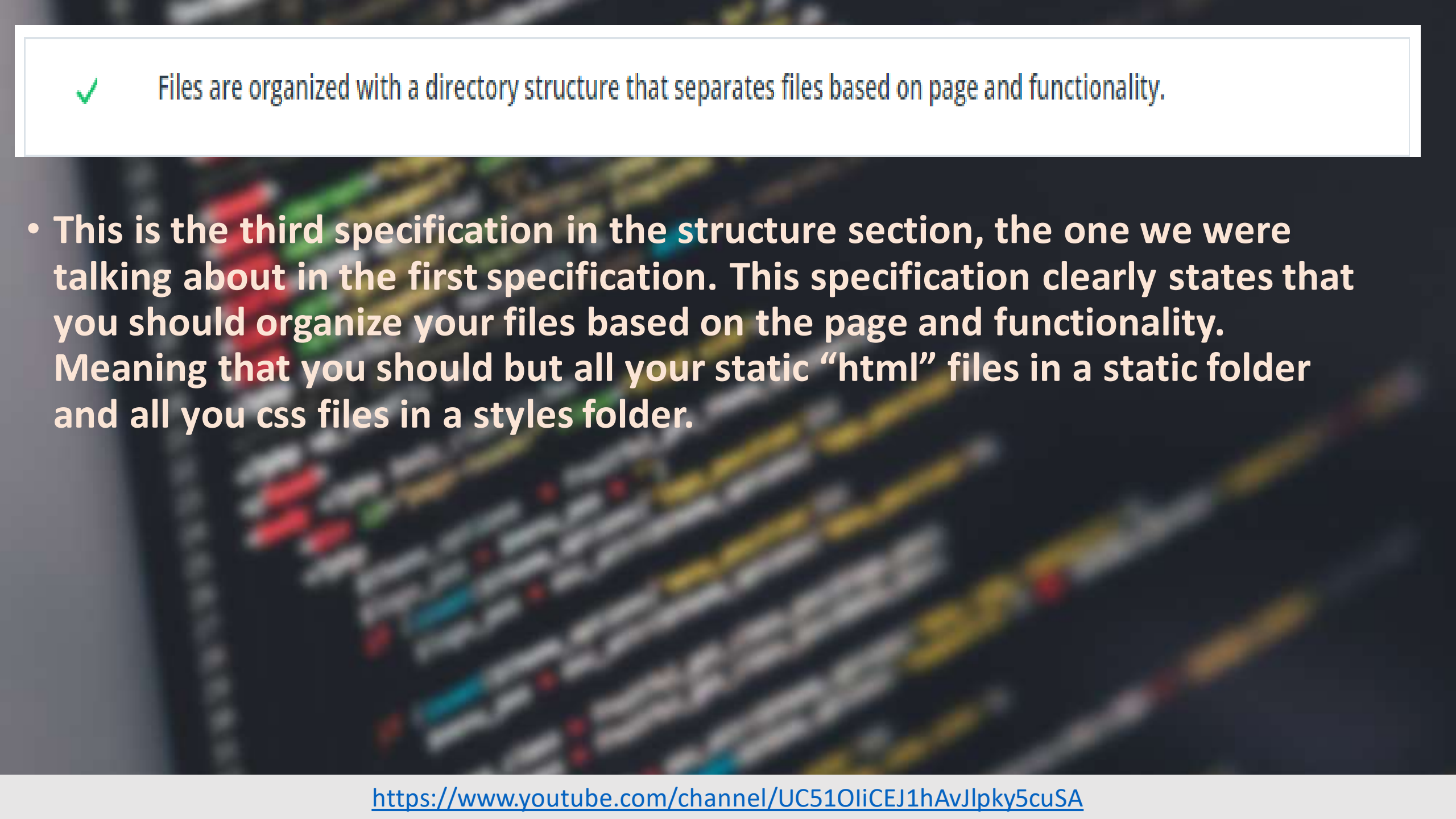

1.Structure



- This is the first specification in the structure section. This specification simply says that your project structure needs to be separate from the design/style meaning that your static “html” files should be separate from you design/style “css” files. Later on we will see another specification clearly states that your html files needs to be in a separate folder from your css files.
- The second points means that there shouldn’t be any attributes what so ever in the document body which is the body tag.
- And finally the last point means that there shouldn’t be any element present in the project’s document.



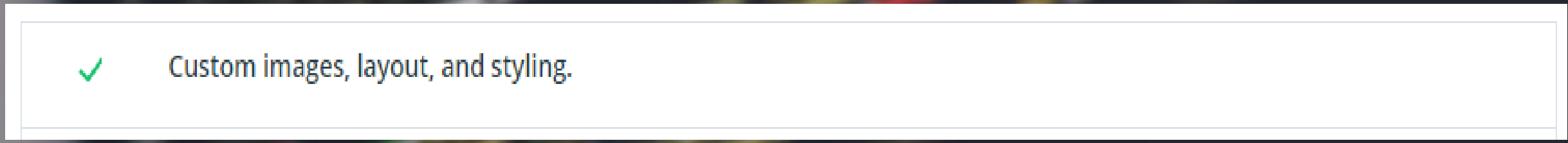
- This is the second specification in the structure section. This specification simply says that in your main css file that's connected to your main html file should be at least 3 imported files. Generally we import other css files with different functionalities, but of course you can separate the css files as you like. As long as there's 3+ imports in the main css you should pass this specification.



- This is the third specification in the structure section, the one we were talking about in the first specification. This specification clearly states that you should organize your files based on the page and functionality. Meaning that you should but all your static “html” files in a static folder and all you css files in a styles folder.

- This is the final specification in the structure section. This specification simply says that your project should have a user flow between the pages. Meaning that if you have to pages: index.html and posts.html that the index.html should have a link that when the user click on it will open the posts.html and the posts.html should have a homepage link that will open the index.html and basically all pages should have something like a door that transfer the user from and to other pages in your project.

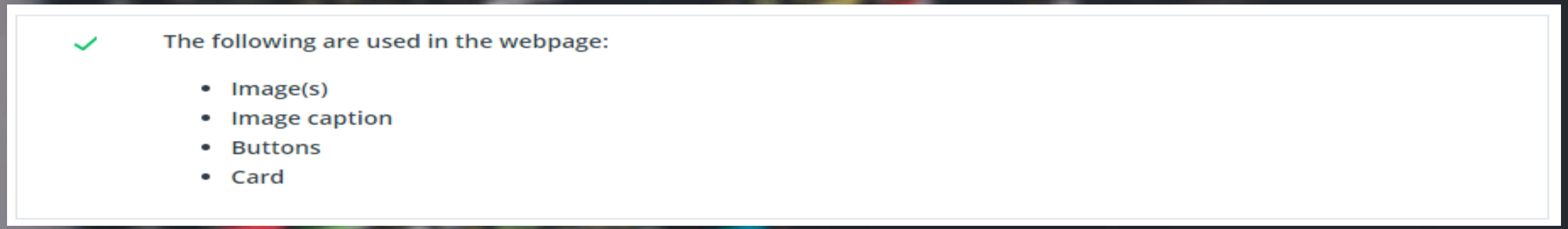

2.Design



- This is the first specification in the design section. This specification simply says that your project should have custom images, layouts and styling. That you should create them personally and don’t copy some images, layouts and styling from other projects or websites.

- Typography: Custom design for typography with at least 3 unique properties for each typography selector
    1. Headers (h1 to h3 at minimum) `<h1>`
    2. Paragraph Text `<p>`
        a. Bold
        b. Italic
        c. Underlined
    3. Links `<a>`
    4. Quotes
- Colors: At least 3 colors are used.

- **This is the second specification in the design section. This specification simply says that your project needs to have headers, links, quotes and bold, italic and underlined text. As addition to using at least 3 colors in the project.**
- **The project specification says that you need to have at least 3 unique properties plus the 3 colors but we will implement all of them in the project.**

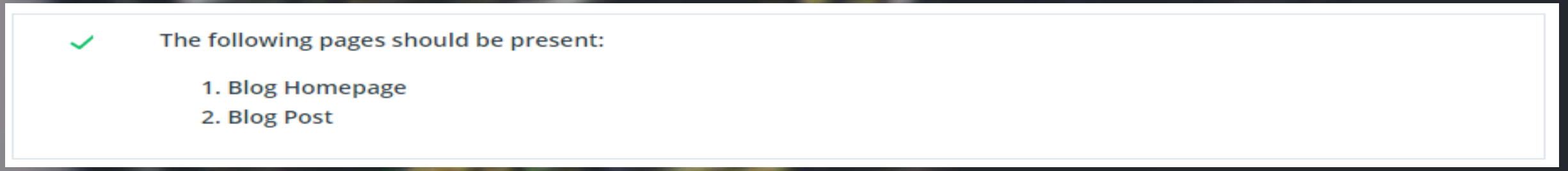


- This is the third specification in the design section. This specification simply says that your project needs to have images, image captions, buttons and cards in it.
- We will see in the next specification that the project's rubric has actually specified where should we put the images, image captions, buttons and the card.

The general pattern of the webpage should be as follows:

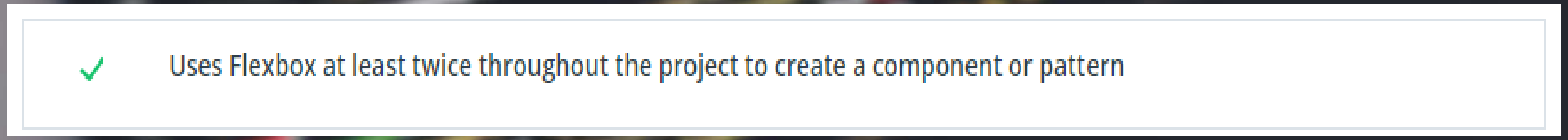
- Blog Homepage
  - Navbar
  - Blog Post Card
    - Card
    - Image
    - Buttons
  - Footer
- Blog Post
  - Header
  - Author Info
    - Name
    - Title
    - Current Company/School
    - Short Bio
    - Avatar
  - Social Sharing
    - Twitter
    - Facebook
    - LinkedIn

- This is the forth specification in the design section. This specification simply says that in your homepage you need to have: a nav tag, a blog post card that contains" card, image and buttons" and a footer.
- In your blog post page you need to have: a header tag, an author info section that contains "Name, Title, Current Company/School", Short Bio and an Avatar and a social sharing section that has "Facebook, twitter and LinkedIn " links.

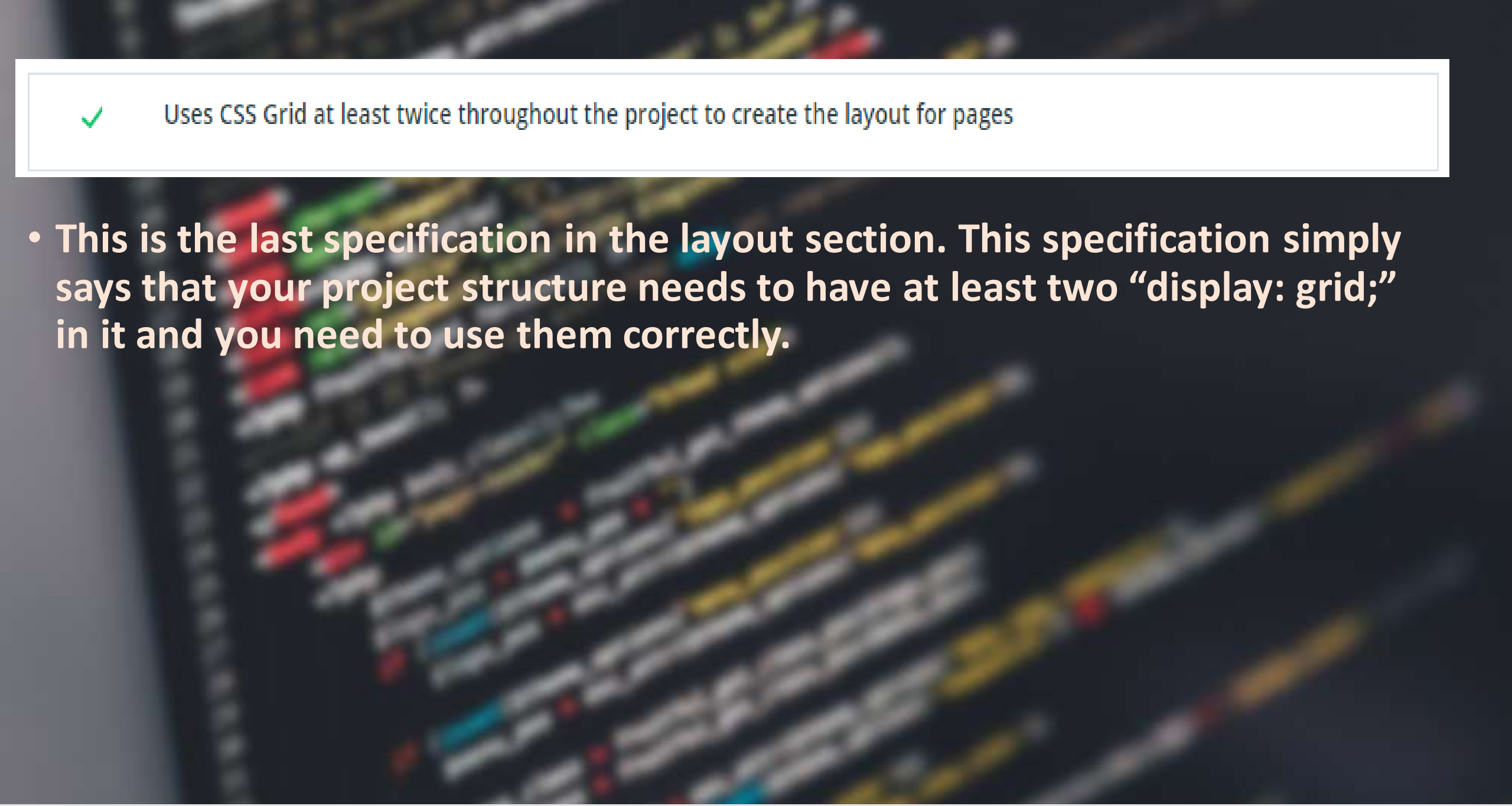


- This is the last specification in the design section. This specification simply says that your project needs to have a Blog Homepage and a Blog Post page in it's structure. You can have more pages but as long as you those two pages in your project you should pass this specification.


3.Layout

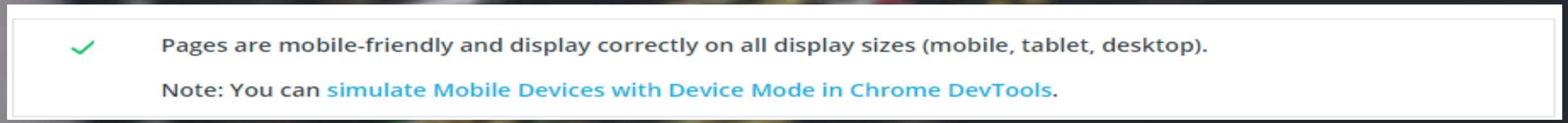


- This is the first specification in the layout section. This specification simply says that your project structure needs to have at least two “display: flex;” in it and you need to use them correctly.



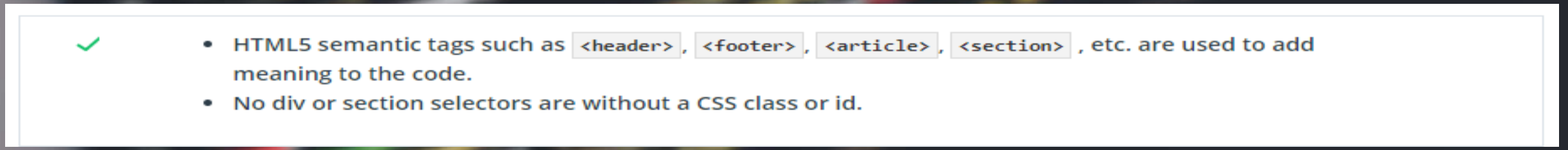
- This is the last specification in the layout section. This specification simply says that your project structure needs to have at least two “display: grid;” in it and you need to use them correctly.


4.Responsiveness

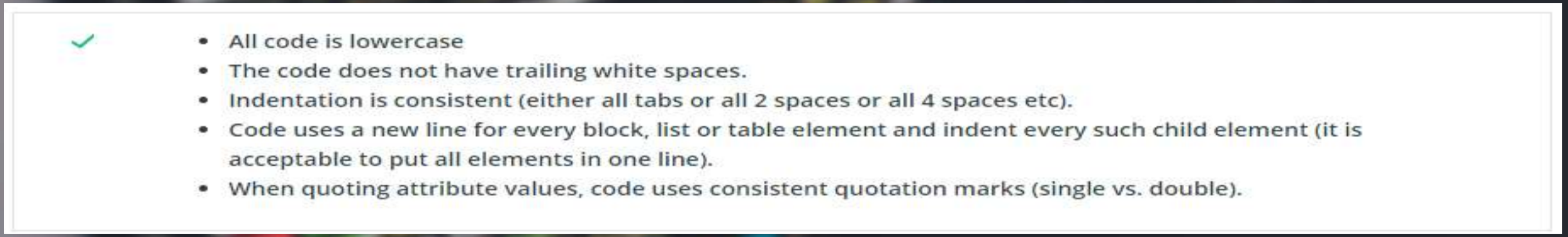


- **This specification simply means that your project needs to be user friendly and responsive in smaller screen sizes and view ports. Whether those view ports are mobile, tablet or even desktop.**

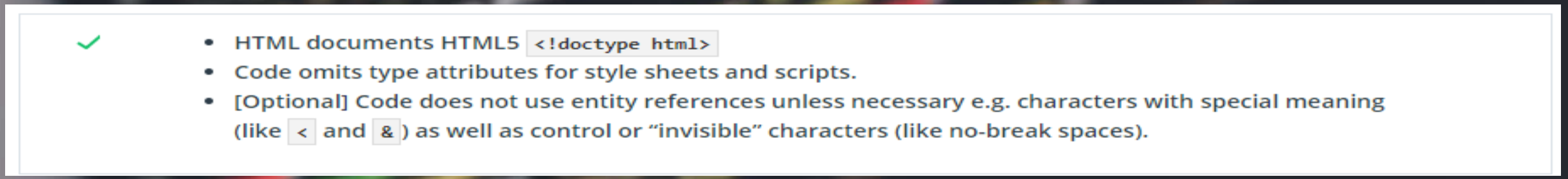

5.Quality



- This is the first specification in the quality section. This specification simply means that your project needs to use html5 semantic tags like nav, header and footer that were already mentioned in the previous specifications.
- The specification also says that there shouldn't be any divs or section selectors without any class or id.



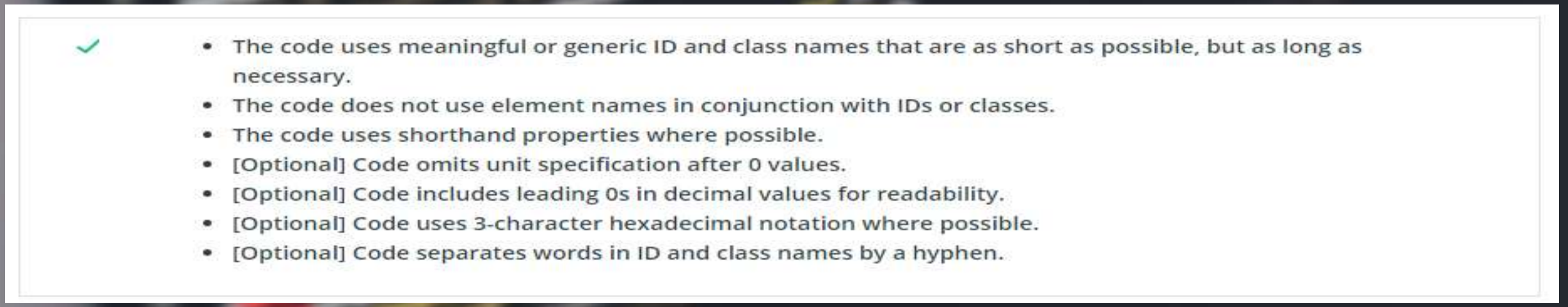
- This is the second specification in the quality section. This specification simply means that all your code needs to be in lowercase and no uppercase.
- You should avoid any trailing white spaces.
- The code indentation needs to be consistent.
- The code uses new line for every single block.
- And that you use consistent quotation marks when using quotes.



- This is the third specification in the quality section. This specification simply means that your project needs to use the <!doctype html> tag in the begging of each page.
- The code should exclude attributes for the style sheets and the scripts.
- And finally this is an optional choice that the code should avoid entity references like & and < and characters like invisible.

- The code does not have trailing white spaces.
- Indentation is consistent (either all tabs or all 2 spaces or all 4 spaces etc).
- Code indents all content, that is rules within rules as well as declarations to reflect hierarchy and improve understanding.
- The code uses a semicolon after every declaration for consistency and extensibility reasons.
- Code always uses a space after a property name's colon, but no space between property and colon, for consistency reasons.
- Code always uses a single space between the last selector and the opening brace that begins the declaration block. Code always start a new line for each selector and declaration.
- Code always put a blank line (two line breaks) between rules.
- Code uses consistent quotation marks for attribute selectors or property values (single vs. double).

- **This is the forth specification in the quality section. This specification simply says that your project should avoid trailing white spaces and consistent indentation which we already saw in the second specification, your code should indent all content "consistent indentation", use semicolons ";" after every declaration, space after properties name's colon, but no space between the property and the colon, you code use single space between the ending and the begging of selectors, blank line between rules and consistent quotations which we will put it under code consistency.**



- This is the last specification in the quality section. This specification simply means that your project needs to use meaningful ids and classes that are short.
- Your code should avoid element names that coexist with ids or classes.
- And that the code should use shorthand properties.