Website Content Final Project
Blog “Frontend for Beginners”

Student Name: Anastasiia Makarochkina
ID: A01064745
WORD COUNT 1739
URL: frontendbegins.ca

Presentation outline

Topic

Frontend for beginners - all the topics related to the frontend development that are potentially useful for novices. Tutorials and guides how to code particular elements of the UI and how to organize work processes.

Intended audience

- Primary - novices to frontend development;
- Secondary - developer looking for specific questions answers.

Purpose

Provide readers with comprehensive content:
- development tools;
- common code components;
- code and CSS organization approaches;
- other practical aspects for a scalable and maintainable code.

Meeting client’s needs by

Structuring all the articles into two broad categories:
- How to use modern tools, build common UI components in code, and utilize best practices for code organization;
- Listicles - overviews of tools and their comparison, and learning resources grouped by topics.

Structure each article to be clear and useful:
- provide with a brief overview of the subject;
- provide with code samples that are easy to copy and start working with;
- add subheads and summaries to each section of long articles.

Specific choices made

in content

- choose topics I encounter every day when working as a frontend developer;
- choose topics popular in frontend development community: in magazines like FreecodeCamp blog or Hackernoon, or by popular developers (open source mostly);
- choose topics that help to make decisions about how to organize code;

navigation

- set one global navigation with all available sections;
- set How To page separately to show better all articles in the subcategories;
- add tags and related articles at the end of the article;
- add sitemap link to major pages;
- add an indicator of a section - how-to or listicles - to the header of the article in the top;
- add links where possible (to other articles in the blog, to external resources).

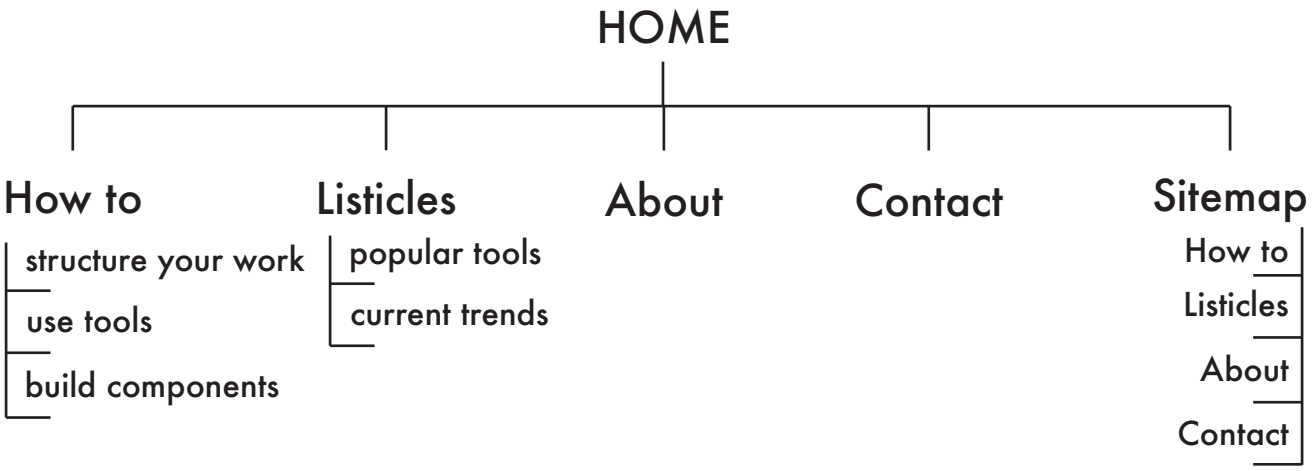
layout

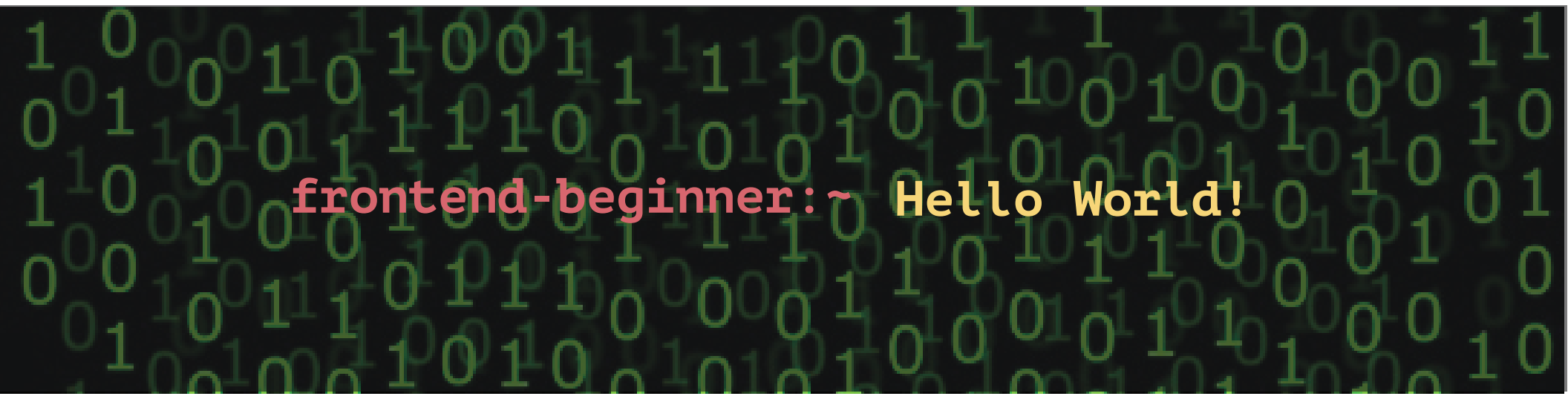
- show current featured article and examples from how-to and listicles sections;
- place subscription box in case people like the topics and want to stay in touch;
- limit article text width for a more readable format and leave a place for possible side-notes;
- add subheads in the article for better scannability.

images

- use recognizable symbols of popular frameworks and illustrations to highlight the article subject;
- use comics to illustrate one of the problems discussed in the article.

Sitemap





FEATURED

Ultimate guide to centering with CSS in 2018

Hello world! It's 2018 and still one of the most daunting CSS-related questions is the centering of the elements on a page. Recently a CSS centering code generator topped 1036 upvotes on HackerNews. Yet, our editorial staff doesn't think it's a solution for everything (see the review). So we compiled our own guide for you to get all the practices for CSS centering in one place.

Read more →

HOW TO See all tutorials →

webpack

How to setup Webpack

Setting up Webpack is notoriously difficult. It has already become a scary tale that frontend developers like to tell each other.

Read more →

LISTICLES See all lists →

Top Javascript Frameworks

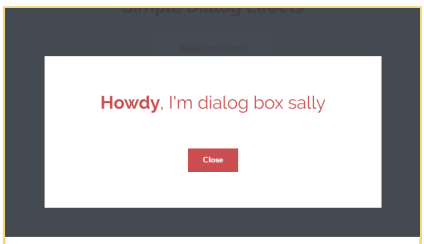
JavaScript community is known to be one of the most dynamic ones. Another day - another new framework. Has Vue.js surpassed React.js already? What to expect from the new version of Angular.js? Or have the new frameworks already emerged to take them all down?

Find out now →

Best JavaScript Learning Resources

This is our curated list of Interactive learning tools and best books in JavaScript. From freeCodeCamp to "You don't know JS" book series, be sure to check them all out and pick the best.

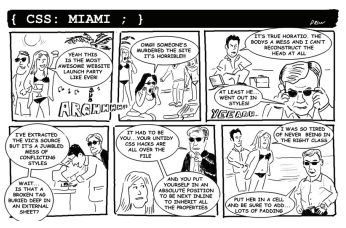
Start learning →



Best practices for creating modal windows

Crucial things to keep in mind when creating a modal window using React, Vue.js, or vanilla JavaScript.

Read more →



Review of modern approaches to CSS organization

Maintaining CSS can be hard, especially when the website grows. Several approaches and naming conventions make it easier.

Read more →

Subscribe to our monthly newsletter

We do send a newsletter with our newer articles,

also sometimes we highlight local meetups or important events.

You can check out an example of the newsletter here.

Email SUBSCRIBE

Haven't found what you were looking for? See sitemap for more links or contact us.

Frontend for Beginners, 527 Carnarvon Street, V3L 0J1 New Westminster, BC

frontend-beginner:~ My name is _

Mission

Mission of this blog is to spread best practices in frontend development. It serves both novices and seasoned professionals. Modern frontend development is a very dynamic and turbulent sphere. Lots of new frameworks, tools, approaches appear every day. And information about them is often chaotic, unstructured and sometimes incomplete.

This blog aims to provide structured reviews of tools and workflows, tutorials and guides. The focus is to help readers to improve their day-to-day working routines and to write clear, maintainable code, CSS and HTML.

Who I am

Anastasiia (Ana) Makarochkina

Founder and editor-in-chief. Ana started working on this blog as a student of the BCIT New Media Web Design and Development program. As a frontend developer in Blinkloader.com, Inc. she was facing loads of new professional information generated by different sources every day. Yet she felt that it still lacked structure and comprehensiveness, and did not address some questions. How to do things while working as a frontend developer? How to structure your stylesheets and code so that they remain maintainable? How to properly and on-time use all the tools? What are essential elements of daily created web components, like form validations, accessibility, cross-browser testing, etc?

All these topics can overwhelm even a Computer Science graduate, which many of the modern software developers are not. That is why, to make things easier for herself and novices like her, Ana started this blog.

Guest posts are welcome!

One author is good, but more authors is greater for the content delivered. So if **you**:

- are starting career as a frontend developer;
- or/and would like to start writing the blogs to become a better professional;
- or else you are simply want to give back to the frontend community -

you are welcome to publish on this website!

Just contact me per email ana@frontendbegins.ca
or via this website's contact form
and we work on it together.

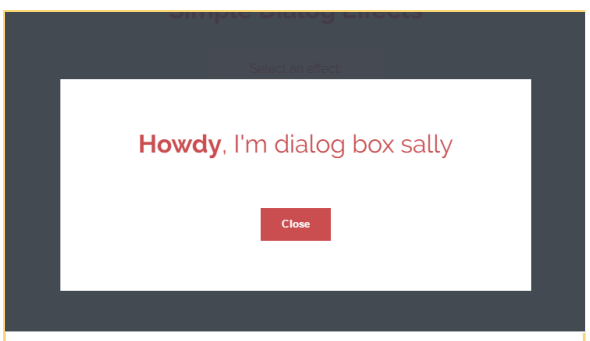
Frontend for Beginners, 527 Carnarvon Street, V3L 0J1 New Westminster, BC

frontend-beginner:~ how to_

This section contains:

- tutorials on how to build common web components, following best development practices;
- guides on how to use and setup modern tools and why they are necessary;
- guidelines about how to structure your code and organize your work in the most efficient and scalable way possible.

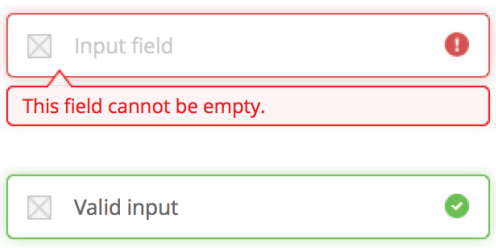
How to build components



Best practices for creating modal windows

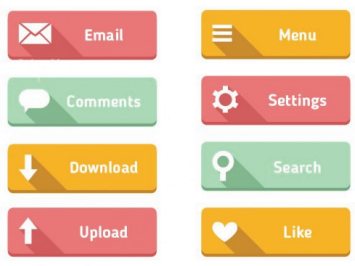
Crucial things to keep in mind when creating a modal window using React, Vue.js, or vanilla JavaScript.

Read more →



Form validation guidelines

Should you display an error message when user is still typing? Which HTML5 validation features can you use safely? Everything about form validation in this article. Read more →



Ultimate guide to buttons

What are common styles you should be adding to a button by default? What you should never do to a button? Why span is a bad button? All answers in our Ultimate guide to buttons. Read more →

How to use modern tools

How to setup Webpack

Setting up Webpack is such a tedious job, that frontend developers both suffer from it and joke about it. But, Webpack is just a module bundler, so you definitely can and should know how to work with it.

Read more →

How to configure Babel

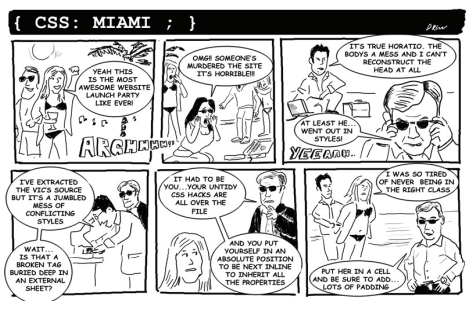
Babel configuration is another dreadful topic among most developers. Stop endless trips to StackOverflow and back, read this article and learn how to configure most popular Babel plugins and presets. Read more →

Tailor Tachyons to your needs

Tachyons is a CSS framework that allows you to use and maintain a system of modular CSS classes. Learn how to generate tachyons' classes tailored to your website design.

Read more →

How to structure your work



Review of modern approaches to CSS organization

Maintaining CSS can be hard, especially when the website grows. Several approaches and naming conventions make it easier.

Read more →

Building and maintaining an error system

Error messages are very important to a user in a stressful situation of things gone wrong. Here's our guide for web developers how to build valuable and actionable error messaging system.

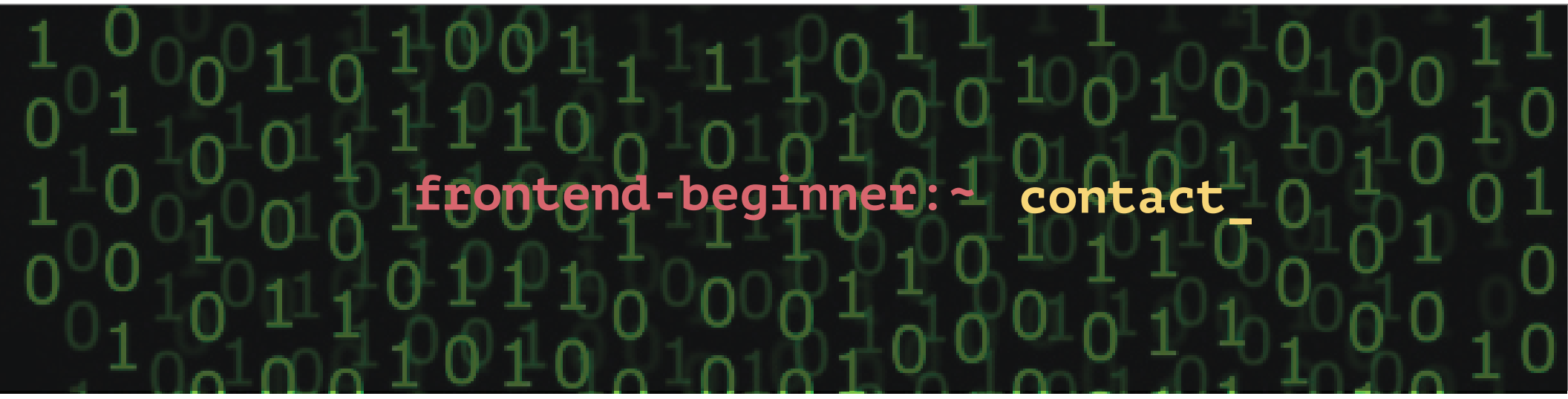
Read more →

Designing modular reusable components in React.js

React.js is a framework that promotes splitting code into small and reusable modules. Beginners tend to simply split existing markup into components, which are not reusable. Read our guide to write better React components. Read more →

Haven't found what you were looking for? See sitemap for more links or contact us.

Frontend for Beginners, 527 Carnarvon Street, V3L 0J1 New Westminster, BC



Feel free to contact me via this form or email ana@frontendbegins.ca!

Contact form

Your Email

Your Name

Your message

SEND

Frontend for Beginners, 527 Carnarvon Street, V3L 0J1 New Westminster, BC

how-to:-
Review of modern approaches to CSS organization

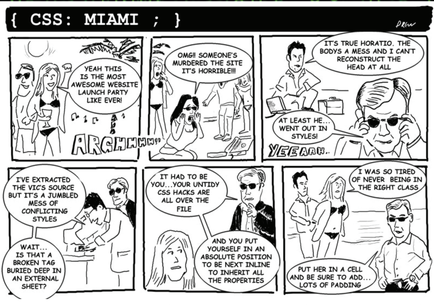


photo source: https://www.cartoonsidrew.com/2011/09/css-miami.html

Let's imagine: you start working on a website which already has a theme and main styles.

Case 1.
You need to adjust an old section styling. You click "inspect element" only to find that you cannot find the style property you are looking for. For example, it could exist only in an intersection of parent and own classes of the element. To find the right combination and then properly alter it becomes a genuinely tricky job. Or else your teammates become too scared to alter old CSS classes because it is used somewhere else, and might break when changed. Even if it is not used anywhere - checking may take quite some time.

Case 2.
You have finished untangling the old code and try to write some new one. You create a class "gray" only to know that a Bootstrap or another internal CSS stylesheet defines it with a different shade. A quest to find that class and fix the conflict begins again.

Semantic and functional CSS

These two cases illustrate two approaches to naming CSS classes.
In the first case, the scary legacy stylesheet uses traditional CSS or semantic approach to CSS. Such an approach usually requires constant thinking over which CSS classes already exist. Buttons/cards/other page elements often differ only in padding, margin, or something else. And CSS duplication begins.

In the second case, the website uses the atomic/utility/functional approach. In this approach, one CSS property uses one class. Key to solving the problem is generating your own system of CSS utility classes and avoid CSS frameworks overhead. Most of the time you don't need all the Bootstrap library styling anyway.

1199 words

Why not still use semantic approach?

By 2019 functional approach seems to win. Most of the semantic approach modern representatives do not stand their own critique. They are not very readable, tend to repeat code and are not intuitive and easy to decode and create new. Some examples are BEM and modular CSS.

In BEM you have to think in terms of an entity - for example, form. Then remember its inner elements. Then add some modifiers - different margins, font sizes, etc. The result is the name of the class. Not very simple.

```
medium2 ⇗ ack marginTop
lib/templates/collection/index.soy
50:      <div class="u-marginTop10 u-xs-top10 u-xs-bottom30 u-borderTopLighter">

lib/templates/collection/settings.soy
67:  <div class="container form form--borderless form--marginTop60 form--marginBottom
100">
74:    <ul class="list list--marginTop30 list--marginBottom0 js-navItemEditors u-size
320">
162:  <div class="container form form--borderless form--marginTop60">
446:    <div class="u-marginTop10">
459:  <div class="inputGroup inputGroup--fontSize18 inputGroup--marginTop40">
619:  <div class="container form form--borderless form--marginTop60">

lib/templates/payments/creditcards.soy
41:    {if $withMarginTop20} creditCardForm--marginTop20{/if}

lib/templates/payments/memberships.soy
286:    <div class="row row--borderTop row--marginTop20 row--paddingTop20 u-flexEnd"
301:    <div class="row row--marginTop15 u-flexEnd">
316:    <div class="row row--marginTop15 row--marginBottom20 u-flexEnd">

lib/templates/post/visibility.soy
55:          <span class="radioInput radioInput--marginTop4"></span>
71:        <span class="radioInput radioInput--marginTop4"></span>
72:          <span class="radioInput radioInput--marginTop4"></span>
```

The "marginTop" modifiers are doing the same thing. CSS duplication is one of the reasons Medium dev team ditched it in favor of utility classes. (https://medium.engineering/simple-style-sheets-c3b588867899)

Or say the modular CSS approach replaces this code

```
<div class="box profile pro-user">
  <img class="avatar image" />
  <p class="bio">...</p>
</div>
```

with this one

```
<div class="l-box m-profile m-profile--is-pro-user">
  <img class="m-avatar m-profile__image" />
  <p class="m-profile__bio">...</p>
</div>
```

source: https://spaceninja.com/2018/09/17/what-is-modular-css/#css-at-scale-is-difficult

The only difference is claimed to be the more clear relationships between the components. If it is relevant for some working environments - that's perfectly fine. When using React for example, you are already establishing that relationship with the Javascript component naming and organization.

CSS can be just CSS, responsible only for the styling.

Like this:
```
<div class="black pa10px ma10px bg-white"></div>
```

This is more readable once you remember simple naming conventions or invent your own. Utility classes use recall rather than painful logical structuring the whole app just to alter some color.

How to use utility classes

1) Use separate classes for separate properties.
Example: relative for position: relative, b--red for border-color: red, black for font-color: black.

2) Decide with the UX designer the consistent components and scales.
For example, heading 1 is 1.5 bigger than heading 2. Or generally, paddings of 1rem, 2rem, and so on are used. Or there are some consistent heights. Design consistency will help to maintain the same look and feel in the website design when creating new components. But that's the topic of yet another article about design systems.

3) Leverage the power of existing utility classes frameworks, like Tachyons, Atomic, or BassCss.
You can start with their predefined classes, or generate your own media breakpoints and update CSS classes with custom values of paddings, margins, color shades, etc.

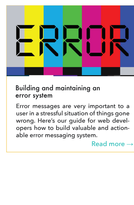
4) Use a semantic class if needed.
Sometimes you may feel you need an entity that is repeated and therefore could use one class, like a small button, or you have to alter several properties at once on :hover and similar events. In such cases using semantic classes is still perfectly fine.

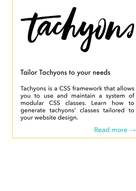
All in all, the simplicity of CSS depends on the level of the design you have at hand to implement. And inconsistent visual elements will force you to grow your CSS stylesheet as well. But at least if a somewhat consistent design system is in place, functional/utility/atomic CSS will allow you greater **flexibility**, **reusability**, and **maintainability** - all vital for growing websites.

More on functional vs semantic CSS:

Simple Style Sheets Or How to Improve Your Sleep by Dropping the Cascade by Medium developers;
The problem with atomic CSS by Maintainable CSS creator;
CSS Utility Classes and "Separation of Concerns" - great in-depth discussion of semantic vs utility classes;
Evolution of CSS: Class Naming Methodologies - more examples of semantic and functional methodologies.

More on components, approaches, and tools





Frontend for Beginners, 527 Carnarvon Street, V3L 0J1 New Westminster, BC