

How to structure, scale and maintain CSS

Methodologies that helps with styling applications

11. 03. 2022

cloudtalk

A little bit of history

The screenshot shows the ViolaWWW browser interface. At the top, there is a menu bar with 'File', 'Navigation', 'Fonts', 'Guides', and 'Help'. Below the menu bar is a toolbar with icons for HOME, BACK, PREV, NEXT, and RELOAD, along with text buttons for 'Viola Central', 'What's New?', and 'Viola GUI help'. The address bar contains the URL 'http://berkeley.ora.com/proj/viola/vw/help_vww_3.3.html'. The main content area displays the title 'User's Guide to ViolaWWW (V 3.3)' and the heading 'Viola Graphical Interface Help'. A left sidebar lists various topics: 'Opening a Doc', 'Navigation', 'Hotlist', 'Cloning Page', 'Show Author', 'Show Source', 'Status', 'Contacts', and 'References'. The main text area provides a quick guide to the GUI functions and details on 'Opening a Document', explaining how to use the 'Operations' menu and the 'Use Selected' button to open a document from a URL. At the bottom, there are 'Up' and 'Down' buttons and a note about editing the URL in the display field.

File Navigation Fonts Guides Help

Viola HOME BACK PREV NEXT RELOAD Viola Central What's New? Viola GUI help

http://berkeley.ora.com/proj/viola/vw/help_vww_3.3.html

User's Guide to ViolaWWW (V 3.3)

Topics:

- [Opening a Doc](#)
- [Navigation](#)
- [Hotlist](#)
- [Cloning Page](#)
- [Show Author](#)
- [Show Source](#)
- [Status](#)
- [Contacts](#)
- [References](#)

Viola Graphical Interface Help

This is a quick guide to the functions on the ViolaWWW's GUI. For details on

Opening a Document

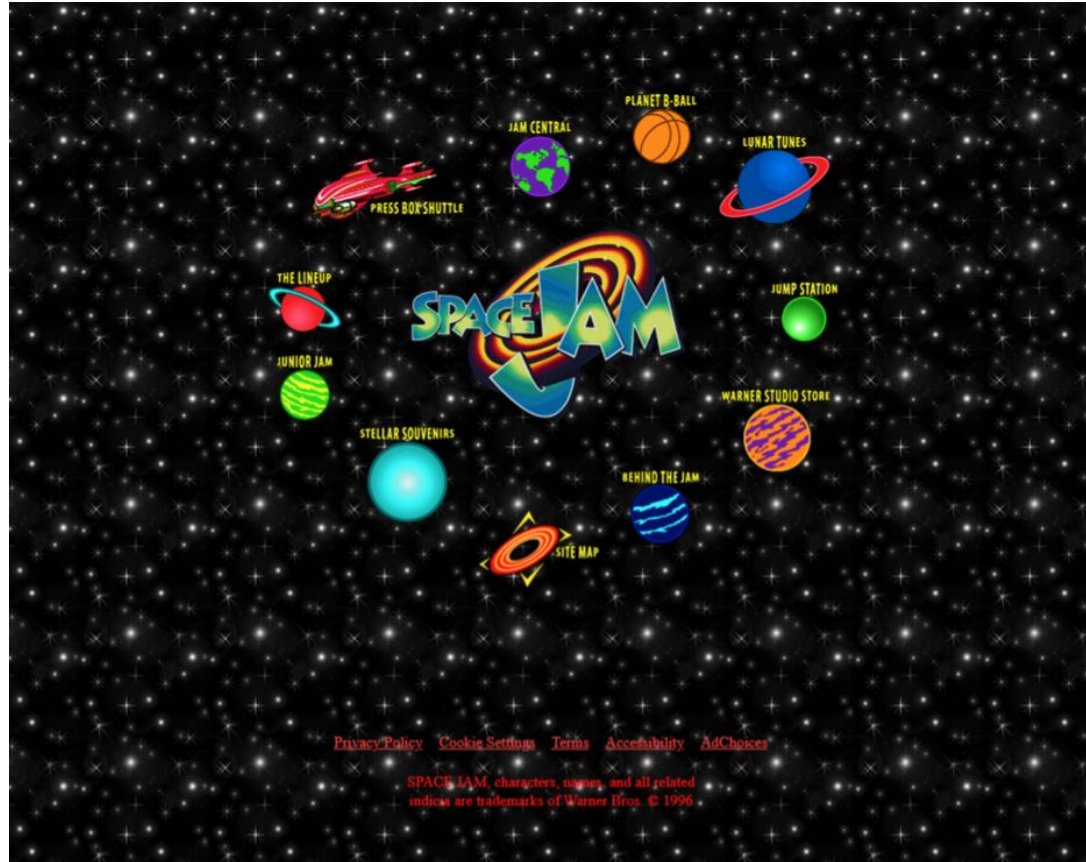
To open a document, pull down the **Operations** menu, choose **Open URL** item. That will cause a dialog box to appear, and into which you can type in a document URL (Uniform Resource Location).

If you already have the URL in a cut-buffer, then you can click on the **Use Selected** button to transfer the URL from the cut-buffer to the dialog box entry field.

If you don't have any interesting URLs on hand, and you're eager to explore the World Wide Web, follow the URL links in the "**Guides**" menu. It will lead you to some interesting and popular destinations on the Web.

Up Down Note that you can also edit the URL in the URL display field, and then

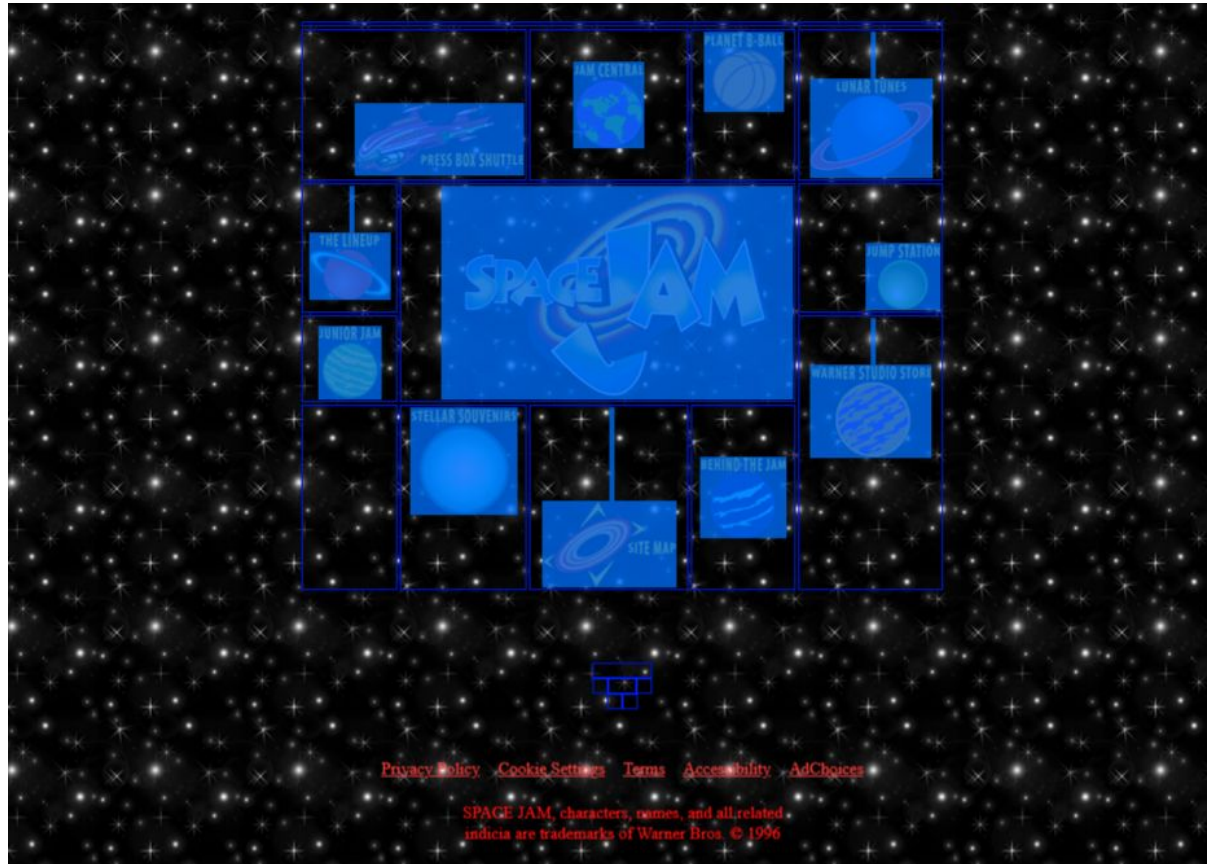
Space Jam



How we style web sites

```
<body bgcolor="#000000" text="#ff0000" link="#ff4c4c" vlink="#ff4c4c"  
alink="#ff4c4c">
```

Layout



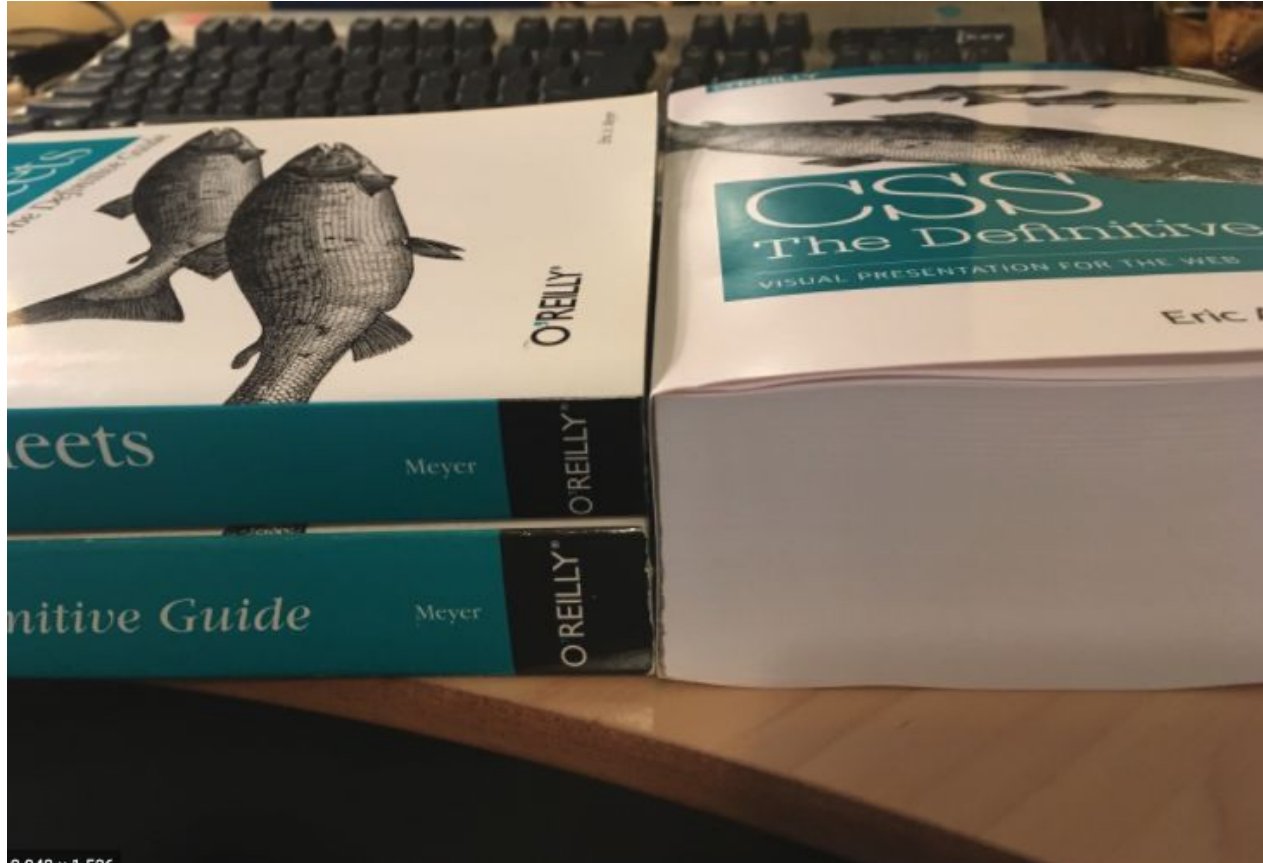
Tables for everything

```
<table width=500 border=0>
<TR>
<TD colspan=5 align=right valign=top>
</td></tr>
<tr>
<td colspan=2 align=right valign=middle>
<br>
<br>
<br>
<center>
<a href="cmp/pressbox/pressboxframes.html">
</a>
</center>
</td>
...
</tr>
...
</table>
```

Today, we have layout properties

- Float (before flexbox come into game)
- Flexbox
- Grid
- Subgrid
- etc.

So, CSS is improving every day



Writing good CSS is hard

“CSS: the only language that is both so easy it’s not worth learning but also so hard that it’s not worth learning.”

So why is CSS so hard to maintain?

Why?

- Deep nesting of selectors and high specificity
- Nesting Hell
- Cascade
- Inheritance
- **Misunderstanding of how CSS works**

Let's take a quick simplest example

```
.red {  
  color: ■red;  
}  
  
.blue {  
  color: ■blue;  
}
```

```
<div class="red blue">Hello</div>  
<div class="blue red">World</div>
```

<https://codepen.io/ondrejko/pen/xxpKRyo>

That means that order in the resulting file matters!

```
8
9 - .red {
10   color: red;
11 }
12 - .blue {
13   color: blue;
14 }
```

≠

```
8
9 - .blue {
10   color: blue;
11 }
12
13 - .red {
14   color: red;
15 }
16
```

Can you check for e.g. selector order in the resulting CSS file?



Because there is thousands of lines

Current Dashboard CSS file with 8200 line of code:

```
8185     display: flex !important;
8186   }
8187   .xs\:table-cell {
8188     display: table-cell !important;
8189   }
8190   .xs\:hidden {
8191     display: none !important;
8192   }
8193   .xs\:flex-row {
8194     flex-direction: row !important;
8195   }
8196   .xs\:pl-5 {
8197     padding-left: 1.25rem !important;
8198   }
8199 }
8200
```

Deep nesting of selectors and high specificity

HTML:

```
<ul class="seznam">
  <li><a href="prvni.html">0dkaz 1</a></li>
  <li><a href="druhy.html">0dkaz 2</a></li>
  <li><a href="treti.html">0dkaz 3</a></li>
</ul>
```

CSS:

```
.seznam li a {
  font-size: 20px;
  color: blue;
}
```

The link color has not changed

What is in CSS and what blocks me?

```
body #content .page ul li a {  
  font-size: 16px;  
  color: red;  
  font-weight: bold;  
}
```


What are the options?

```
body #content .page ul li a {  
  font-size: 16px;  
  color: red;  
  font-weight: bold;  
}
```

Selector overload (or refactor code or use !important)

```
body #content .page ul.seznam li a {  
  font-size: 16px;  
  color: red;  
  font-weight: bold;  
}
```

Such a selector can be really hard to modified

Nesting Hell

```
.Checkbox--toggle {
  padding: $checkbox-toggle-diameter / 10 0;
  .Checkbox {
    &-input {
      &:checked {
        & + .Checkbox-label {
          @extend .Checkbox-toggle - active;
        }
      }
      &:not(:checked) {
        &:focus {
          & + .Checkbox-label {
            &:before {
              background-color: $checkbox-toggle-active-handle-bg;
            }
          }
        }
      }
      & + .Checkbox-label {
        background-color: rgba($checkbox-toggle-bg, 0.46);
      }
      &[disabled], &[readonly] {
        & + .Checkbox-label {
          @extend .Checkbox-toggle - disabled;
        }
      }
    }
    &-label {
      @extend .Checkbox-toggle;
    }
  }
}
```

You would say, that is not so bad, but..

Output in style.css:

```
.Checkbox-toggle--active,  
.Checkbox--image .Checkbox-input:checked + .Checkbox-image .Checkbox-toggle,  
.Checkbox--image  
  .Checkbox-input:checked  
  + .Checkbox-image  
  .Checkbox--toggle  
  .Checkbox-Label,  
.Checkbox--toggle  
  .Checkbox--image  
  .Checkbox-input:checked  
  + .Checkbox-image  
  .Checkbox-Label,  
.Checkbox--toggle .Checkbox-input:checked + .Checkbox-Label {  
  background-color: #71c5e8;  
}  
  
.Checkbox-toggle--active::before,  
.Checkbox--image  
  .Checkbox-input:checked  
  + .Checkbox-image  
  .Checkbox-toggle::before,  
.Checkbox--image  
  .Checkbox-input:checked  
  + .Checkbox-image  
  .Checkbox--toggle  
  .Checkbox-Label::before,  
.Checkbox--toggle  
  .Checkbox--image  
  .Checkbox-input:checked  
  + .Checkbox-image  
  .Checkbox-Label::before,  
.Checkbox--toggle .Checkbox-input:checked + .Checkbox-Label::before {  
  left: calc(100% - 20px);  
  background-color: #0284ff;  
}  
  
.Checkbox-toggle--disabled,  
.Checkbox--image .Checkbox-input[disabled] + .Checkbox-image .Checkbox-toggle,  
.Checkbox--image  
  .Checkbox-input[disabled]  
  + .Checkbox-image  
  .Checkbox--toggle  
  .Checkbox-Label,  
.Checkbox--toggle  
  .Checkbox--image  
  .Checkbox-input[disabled]  
  + .Checkbox-image  
  .Checkbox-Label,  
.Checkbox--image .Checkbox-input[readonly] + .Checkbox-image .Checkbox-toggle,  
.Checkbox--image  
  .Checkbox-input[readonly]  
  + .Checkbox-image  
  .Checkbox--toggle  
  .Checkbox-Label,
```

“Nesting selectors” is a good friend but a bad lord

It is always good approach to stay simple as possible

```
.button {  
  padding: 10px;  
  
  @include breakpoint(tablet) {  
    padding: 8px;  
  }  
  
  &:hover {  
    background: blue;  
  }  
  
  &.is-active {  
    color: red;  
  }  
  
  &-icon {  
    max-width: 16px;  
  }  
  
  &-text {  
    font-size: 0.875rem;  
  }  
}
```

Takeaway: Avoid nesting as much as possible

If you need to nest selectors in the third level, something is wrong with design/usage of the component and we call it “**Design smell**”.



The image shows a screenshot of a Wikipedia article page for "Design smell". The page layout includes a sidebar on the left with the Wikipedia logo and navigation links, a main content area with a title and a paragraph, and a table of contents box.

WIKIPEDIA
The Free Encyclopedia

[Main page](#)
[Contents](#)
[Current events](#)
[Random article](#)
[About Wikipedia](#)
[Contact us](#)
[Donate](#)

[Contribute](#)
[Help](#)
[Learn to edit](#)

Article [Talk](#)

Design smell

From Wikipedia, the free encyclopedia

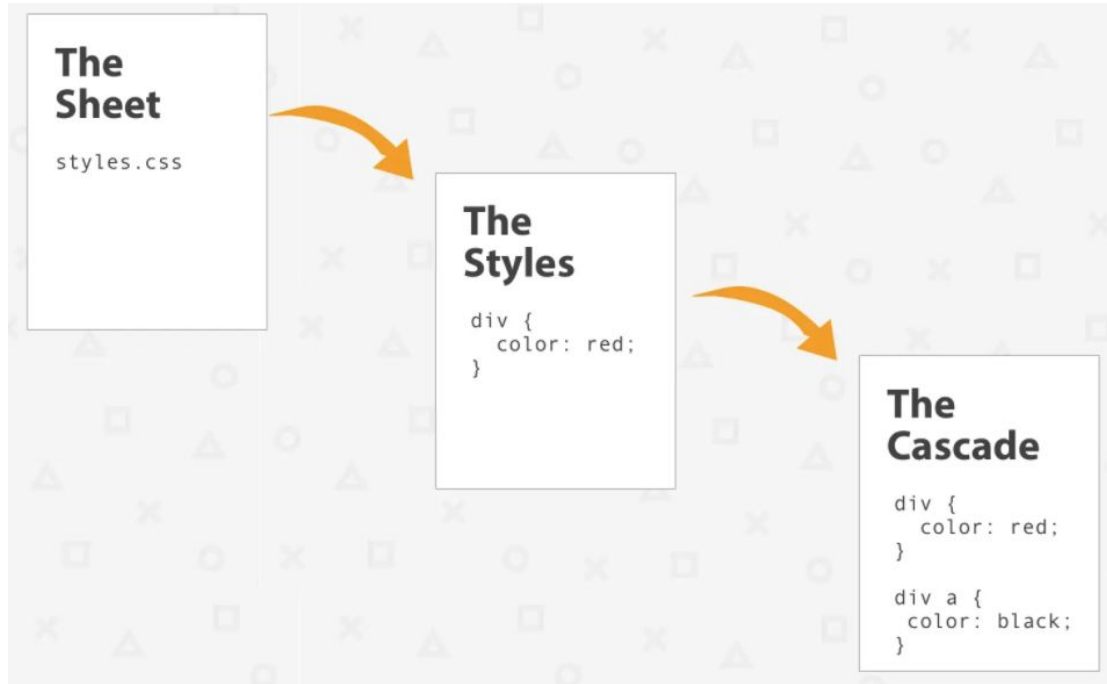
In **computer programming**, **design smells** are "structures in the design that indicate violation of fundamental design principles and negatively impact design quality".^[1] T

Contents [\[hide\]](#)

- [Details](#)
- [Common design smells](#)
- [See also](#)
- [References](#)

Details [\[edit \]](#)

Cascade



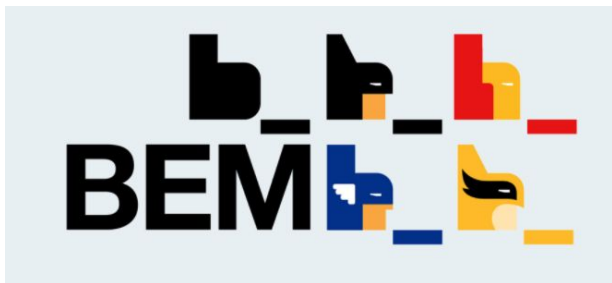
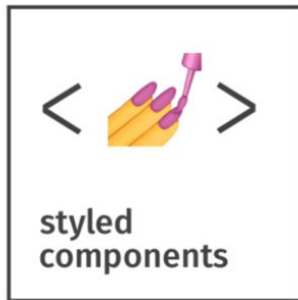
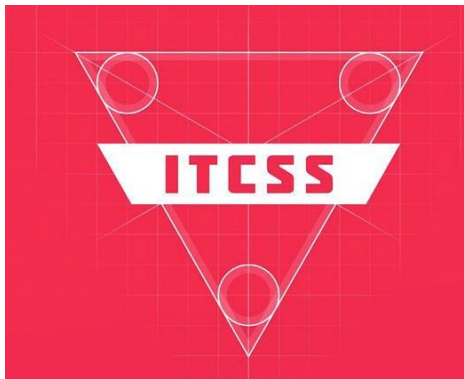
We often end up like this



Our current Dashboard

<pre>body { color: #3d3e40; background-size: 220px 220px; }</pre>	styles.0510_d7b7f.css:5
<pre>body, html { font-family: Roboto,sans-serif; -webkit-font-smoothing: antialiased; font-size: 12px; }</pre>	styles.0510_d7b7f.css:5
<pre>body { font-family: Helvetica-Neue,Helvetica,Arial,sans-serif; font-size: 14px; line-height: 1.428571429; color: #333; background-color: #fff; }</pre>	styles.0510_d7b7f.css:5
<pre>body { font-family: Roboto,sans-serif; }</pre>	styles.0510_d7b7f.css:1
<pre>body { color: #3d3e40; background-size: 220px 220px; }</pre>	<style>
<pre>body, html { font-family: Roboto,sans-serif; -webkit-font-smoothing: antialiased; font-size: 12px; }</pre>	<style>
<pre>body { font-family: Helvetica-Neue,Helvetica,Arial,sans-serif; font-size: 14px; line-height: 1.428571429; color: #333; background-color: #fff; }</pre>	<style>

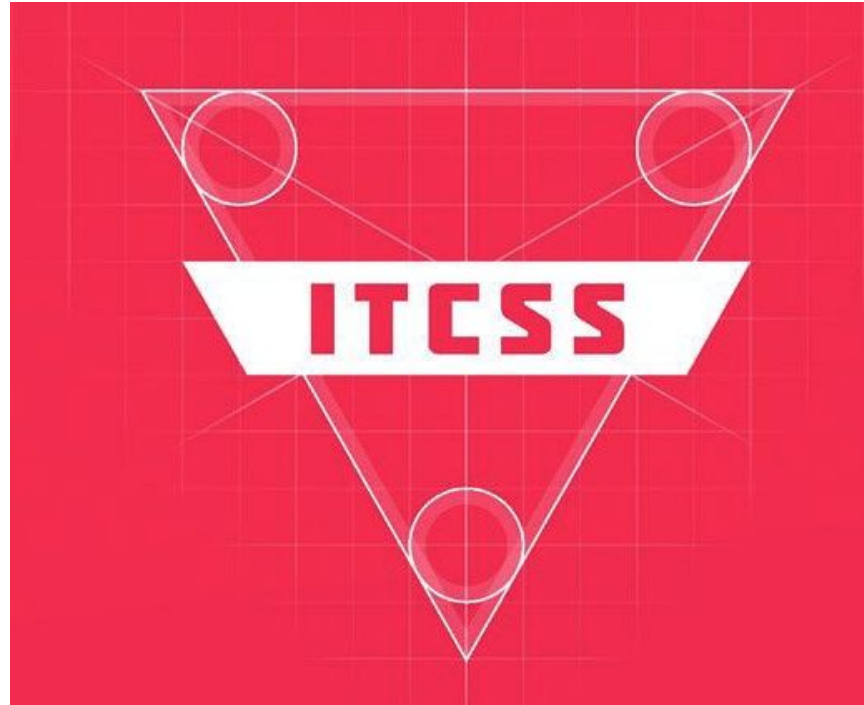
What we can do? Use some good approaches



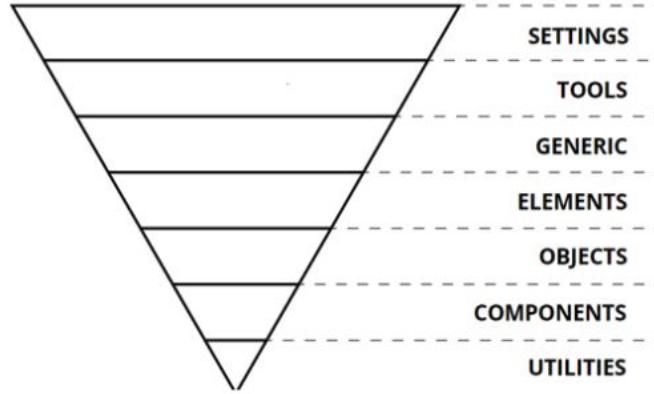
SUIT
CSS



ITCSS



ITCSS



ITCSS

Settings — Space for preprocessors with variables such as colors, design tokens, typography, grid.

Tools — Layer with mixins, functions, media queries.

Generic — Here we insert styles for third party libraries such as normalize, reset or any others

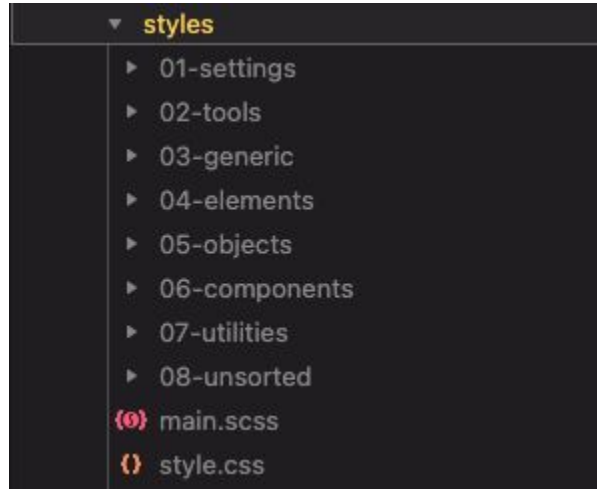
Elements — Selectors for bare HTML elements such as h1, p, article, a

Objects — Class definitions for layout, grid, indentation - reusable non-decorative styles.

Components — Specific components across the project - accordion, buttons, breadcrumbs, tooltip.

Utilities — Class utilities that are designed to affect one particular CSS property and are in most cases written with the utmost importance. Utilities and helper classes with ability to override anything which goes before in the triangle.

ITCSS structure is great for any project and it is easy to use



BEM



BLOCK



ELEMENT



MODIFIER

What is important to realize?

“The more experienced developer you are, the more you prefer **code readability** to efficiency”

Why BEM?

- Find and write CSS rules in a large project is easy.
- Organize rules for media queries and reusable libraries.
- **Reduce the complexity and nesting of your CSS selectors.**
- Have a consistent approach to positioning elements on the page.
- Have a consistent approach to changing the look of HTML.
- **Have a consistent approach to composing larger components from smaller components.**
- A unified approach that is easy to explain to newcomers
- It keeps the world of CSS safe from mess and clutter.

BEM is G. R. E. A. T

SOME ENGAGING TITLE

There are many variations of passages of Lorem Ipsum available.

The majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable.

READ MORE

```
<article class="card">
```

```
<h1 class="card_title">Some engaging title</h1>
```

```
<p class="card_text"> There are many variations....</p>
```

```
<p class="card_text card_text--secondary"> The majority....</p>
```

```
<a class="card_button button" href="#">Read more</a>
```

```
</article>
```

G. R. E. A. T

G for Global

BEM is one of the most recognized naming conventions out there. So if you are introducing a new team member to your BEM project, there's a good chance they already know the convention, which reduces initial friction and allows them to be productive since day 1.

G. R. E. A. T

R for Readable

Thanks to descriptive class names given to basically every element, the stylesheet is easy to read on its own. Not only selectors look better, they also work faster than deeply nested ones.

DO

```
.menu{
  ...
  &__list { ... }
  &__link { ... }
  &__icon { ... }
}
.menu--secondary {
  ...
}
```

DON'T

```
nav {
  ...
  ul {
    li {
      i { ... }
      ul { .. }
    }
  }
}
```

G. R. E. A. T

E for Expandable

As the specificity of CSS selectors is minimal, adding another variation is very simple. Single modifier class should be enough — no more ‘at least equal selector weight’ toil.

```
.menu{
  ...
  &__list { ... }
  &__link { ... }
  &__icon { ... }
}
.menu--secondary {
  .menu__icon{...}
}
```

...

```
.menu--secondary .menu__icon{ ... }
```

```
nav {
  ...
  ul {
    li {
      i { ... }
      ul {
        i { ... }
      }
    }
  }
}
```

...

```
nav ul li ul i { ... }
```

G. R. E. A. T

A for Adaptable

Sharing the philosophy of modularity, BEM naturally works fine with frameworks. Also, styling is independent of elements type and nesting, making it less prone to break when tackling with document structure.

HTML:

```
<article class="card">
  <a class="card__button">click me</a>
</article>
```

CSS:

```
.card__button { ... }
```

HTML changed:

```
<article class="card">
  <div>
    <button class="card__button">click me</button>
  </div>
</article>
```

STILL WORKS

HTML:

```
<article>
  <a>click me</a>
</article>
```

CSS:

```
article a { ... }
```

HTML changed:

```
<article>
  <div>
    <button>click me</button>
  </div>
</article>
```

NO LONGER WORKS

G. R. E. A. T

T for Tough

**There are only two hard things in Computer Science:
cache invalidation and naming things.**

When you start following BEM (fully and honestly), you'll probably find yourself struggling with it constantly. Paradoxically, it's a good thing:

- finding proper block names makes the code clean and legible to others (your future self included)
- reusing existing blocks
- avoiding multi-level nesting makes you rethink document structure

Tailwind, and why I would consider not using it



Tailwind is good for

- Quickly prototyping
- Safety - there is nothing in design that is not in config
- Small projects like personal sites, blog sites etc.

Tailwind is (IMHO) not good for large scaled projects

- Styling and HTML are Mixed
- It Takes Time to Learn (not necessarily a disadvantage)
- Lack of Important Components (not so much components)
- Components aren't provided by default
- Unreadable class names
- It's Inconsistent (items-*: align or justify? content-*: align or justify?)
- It's Difficult to Read
- Really hard to do code reviews
- You Can't Chain Selectors
- Tailwind Locks You Into the Utility CSS Paradigm
- Tailwind Is an Unnecessary Abstraction
- Tailwind, Dev Tools, and Developer Experience (Impossible create variants)

It's Inconsistent

items-*: align or justify?

content-*: align or justify?

justify-*: content or items?

align-*: content or items?

It's hard to read

```
<div class="w-16 h-16 md:w-32 md:h-32 lg:w-48 lg:h-48"></div>
```

```
<div class="w-16 h-16 rounded text-white bg-black py-1 px-2 m-1 text-sm md:w-32 md:h-32  
md:rounded-md md:text-base lg:w-48 lg:h-48 lg:rounded-lg lg:text-lg" > Yikes. </div>
```

It's hard to read

input.tw-w-[20px].tw-h-[20px].tw-p-0.tw-border.tw-mr-1.tw-mt-[2px].tw-mb-0.tw-ml-[2p...



Delete submissions

Mark as v

Console Sources Network Performance Memory Application Security Lighthouse Redux

```
<div class="media"> flex
  <div class="media-figure"> flex
    <div class=
      <label>
        <input data-testid="checkbox" class="tw-w-[20px] tw-h-[20px] tw-p-0 tw-border tw-mr-1 tw-mt-[2px] tw-mb-0 tw-ml-[2px] tw-box-border tw-absolute tw-top-auto before:tw-content-empty before:tw-absolute before:tw-origin-top-left focus:tw-shadow-checkbox tw-cursor-pointer hover:tw-border-teal tw-border-gray focus:tw-border-gray focus:hover:tw-border-teal focus:hover:checked:tw-border-teal-darkest checked:tw-bg-teal-darker checked:tw-border-teal-darker focus:checked:tw-border-teal-darker hover:checked:tw-bg-teal-darkest hover:checked:tw-border-teal-darkest dark:hover:checked:tw-bg-teal dark:hover:checked:tw-border-teal checked:focus:tw-shadow-checkbox tw-bg-teal-darker hover:tw-bg-teal-darkest hover:tw-border-teal-darkest checked:before:tw-bg-gray-lightest checked:after:tw-bg-gray-lightest dark:checked:tw-bg-teal-lighter dark:checked:tw-border-teal-lighter dark:checked:hover:tw-bg-teal dark:checked:hover:tw-border-teal before:tw-h-[11px] before:tw-inline-block before:tw-w-[3px] before:tw-rounded-sm before:tw-left-[7px] before:tw-top-[13px] before:tw-transform before:tw-rotate-[-135deg] after:tw-w-[3px] after:tw-h-[7px] after:tw-rounded-sm after:tw-content-empty after:tw-absolute after:tw-top-[7px] after:tw-transform after:tw-rotate-45 after:tw-left-[3px] dark:after:tw-bg-transparent checked:before:tw-bg-gray-lightest checked:after:tw-bg-gray-lightest dark:checked:after:tw-bg-gray-darkest dark:checked:before:tw-bg-gray-darkest dark:checked:tw-bg-teal-lighter dark:checked:tw-border-teal-lighter tw-bg-teal-darkest dark:tw-bg-gray-dark before:tw-bg-gray-lightest after:tw-bg-gray-lightest dark:before:tw-bg-gray-darkest checked:dark:before:tw-bg-gray-darkest dark:after:tw-bg-gray-darkest hover:checked:tw-bg-teal-darkest hover:checked:tw-border-teal-darkest" type="checkbox" value>...</input> == $0
```

```
.thing {  
  width: 16px;  
  height: 16px;  
  color: white;  
  background-color: black;  
  padding: 0.25rem 0.5rem;  
  margin: 0.25rem;  
  border-radius: 0.25rem;  
  font-size: 0.875rem;  
  line-height: 1.25rem;  
}  
  
@media screen and (min-width: 768px) {  
  .thing {  
    width: 32px;  
    height: 32px;  
    border-radius: 0.375rem;  
    font-size: 1rem;  
    line-height: 1.5rem;  
  }  
}  
  
@media screen and (min-width: 1024px) {  
  .thing {  
    width: 48px;  
    height: 48px;  
    border-radius: 0.5rem;  
    font-size: 1.125rem;  
    line-height: 1.75rem;  
  }  
}
```

More pleasure for eyes

You Can't Chain Selectors

```
.nav-link:focus,  
.nav-link:hover,  
.nav-link[aria-current="page"] {  
  /* CSS goes here */  
}
```

Or better yet, this:

```
.nav-link:is(:focus, :hover, [aria-current="page"]) {  
  /* CSS goes here */  
}
```

It's Harder to Tweak CSS in Dev Tools

- It is hard to simulate styling in DevTools
- It's Harder to Find Components in Dev Tools
- Recompiling HTML Is Slower Than Recompiling CSS

Tailwind Is Still Missing Some Key Features of CSS

What will be in CSS specification soon? What is in working drafts?

- Container Queries
- :has() selector
- @when/@else rules
- Cascade Layers
- Subgrid
- Nesting