CSS in JS

or an iron ring made of wood

Frontend Junior Program - 2022

Fashion-forward development

Buzzword-first Design

The Definitive Guide

O RLY?

@ThePracticalDev

What is it then? CSS? JavaScript?
Or both at the same time?

CSS in JS is a discomforting concept at first – sometimes even seasoned CSS gurus do not understand it.

The more experienced, the more confused they are.

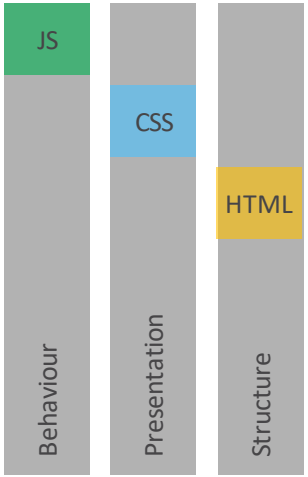
They've learned that concerns must be separated, so presentation and behavior should not be tangled together.

They are right.
Still, they are wrong – at the same time.

Let me explain.

is it a cat or a bus? obviously both – just to save the day

Separation of Concerns – presentation vs behaviour



separation on the technology level

Amazing results can be achieved by editing the CSS only, so you may think presentation = CSS, behaviour = JS

Web = interaction

Technology, however, is just a low-level implementation layer here, why would we stick with that?

CSS Zen Garden is a pure document. While a hyperlinked, web document, still a document. JavaScript – if there is any – can be separated from the presentation.

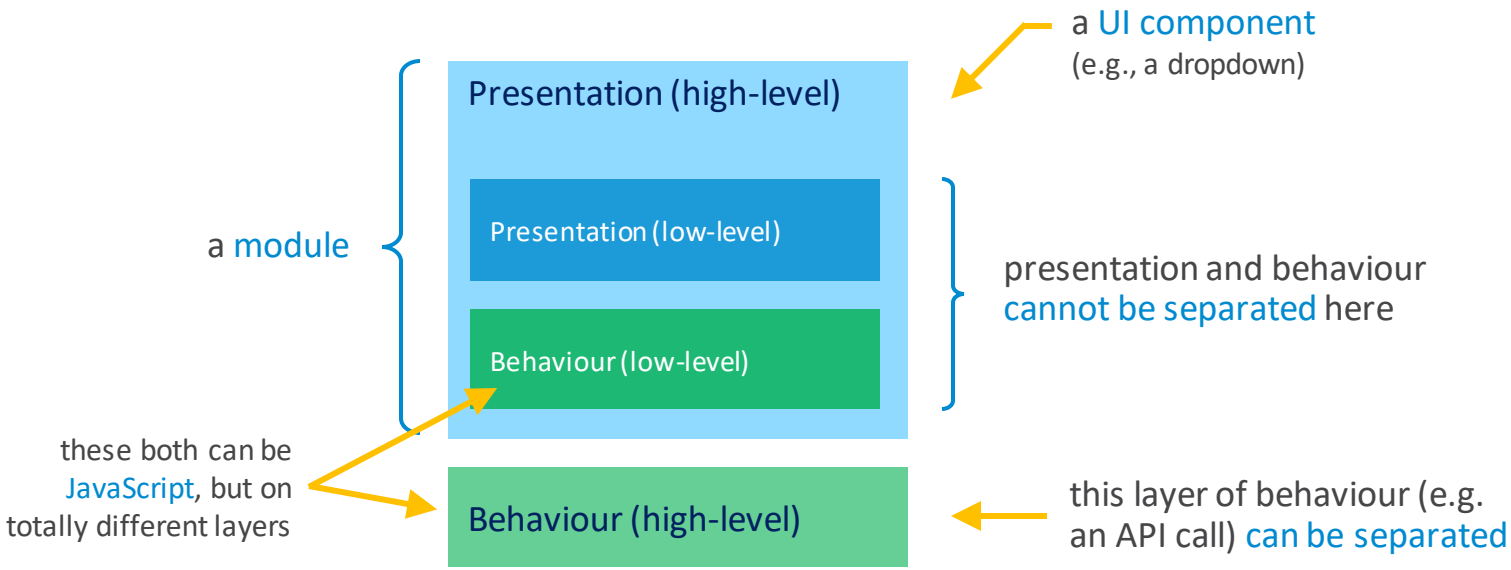
However, this is not true for most of the web. A complex website (even content focused) is full of interaction – and interaction is a presentation and behaviour at the same time. They cannot be separated – if you redesign the presentation, you must redesign the behavior as well.

Also, form and function – by nature – cannot be separated.

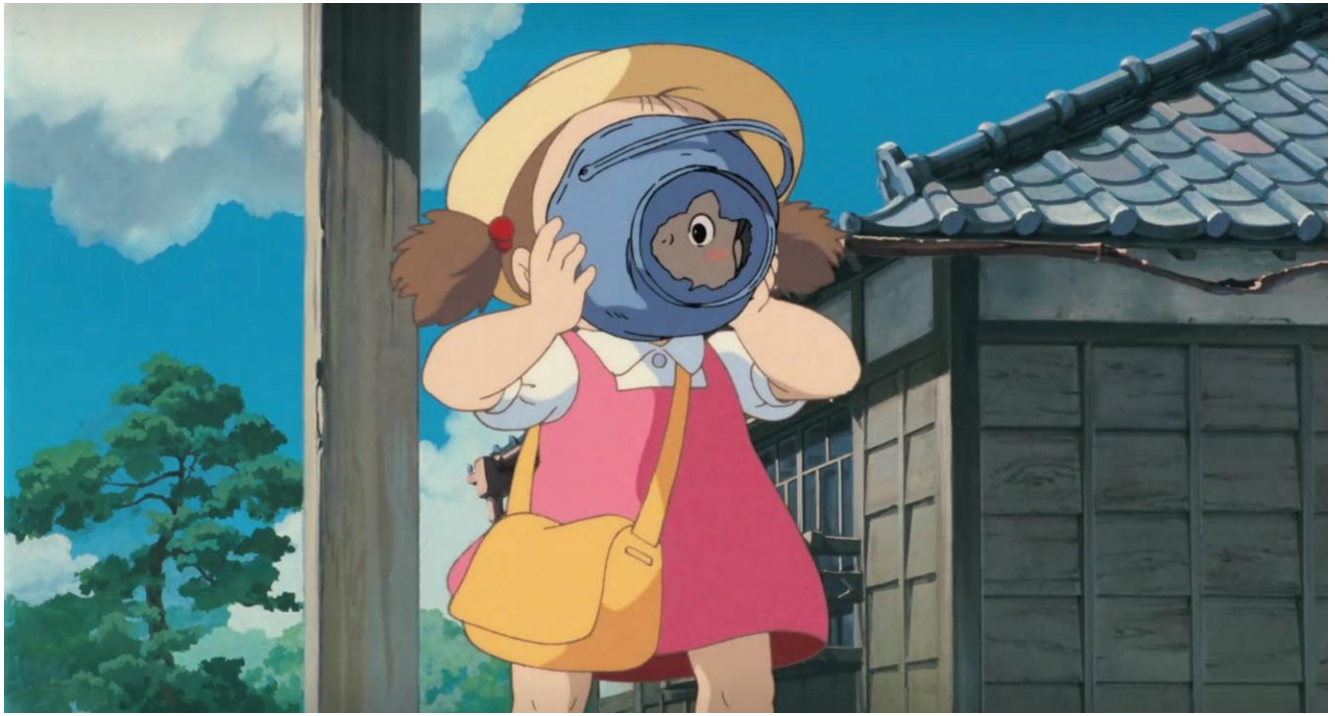
form and function come together

Our understanding on beauty is deeply rooted in functionality; sometimes the connection is not that trivial, still, it is there.

Presentation vs behavior – on a higher level



Think this way – multiple layers of implementation



CSS in JS does not contradict of the separation of concerns.

It just should be considered from a different viewpoint.

CSS in modules

Presentation and behavior are sometimes tightly connected

Therefore, it makes sense to encapsulate them into one module. There is a problem, however, and it is the global nature of CSS.

How can we modularize the CSS, then?

One way is using class name conventions, like BEM (Block Element Modifier) + SASS. It is fine, still, it is just a convention.

Another method is the CSS Modules.

```
<form class="form form--theme-xmas form--simple">
  <input class="form__input" type="text" />
  <input
    class="form__submit form__submit--disabled"
    type="submit" />
</form>
```

BEM is a naming convention for class names

CSS Modules

CSS MODULES

CSS Modules uses conventional CSS files

However, the class names are special.

unconventional class name declarations…

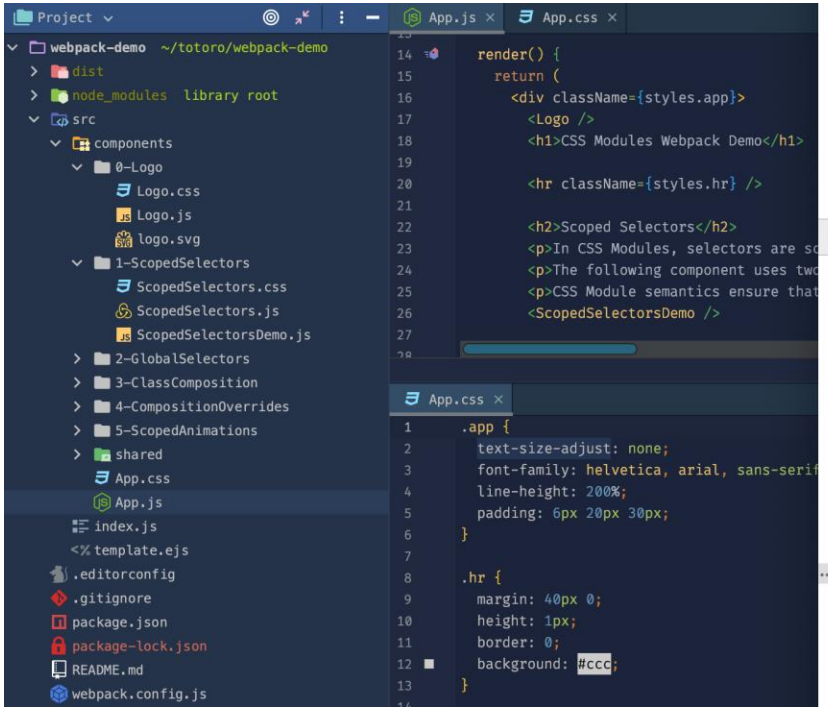
```
render() {
  return (
    <div className={styles.app}>
      <Logo />
      <h1>CSS Modules Webpack Demo</h1>

      <hr className={styles.hr} />
```

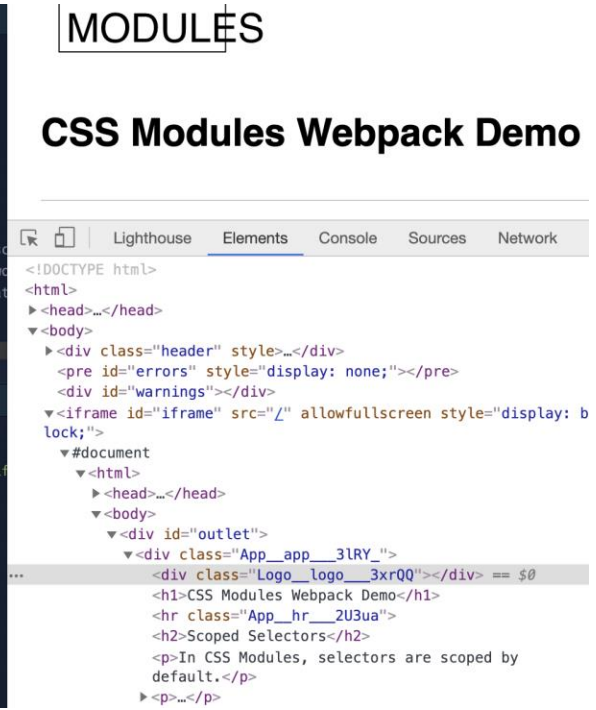
…and the results

```
▼<body>
  ▼<div id="outlet">
    ▼<div class="App__app___3lRY_"> == $0
      <div class="Logo__logo___3xrQQ"></div>
      <h1>CSS Modules Webpack Demo</h1>
      <hr class="App__hr___2U3ua">
      <h2>Scoped Selectors</h2>
      <p>In CSS Modules, selectors are scoped by
```

MODULES

independent components
encapsulating
CSS and JavaScript files





conventional CSS files,
these files will be concatenated together
with the modified class names

please download and test the demo!

It still requires SASS to be able to reuse common CSS parts; you will run into issues with theming, composing CSS into a hierarchy and the whole solution will be anachronistic - it will lack the power of the modern JavaScript ecosystem: using variables (with static type-checking), conditionals, simple composition, etc., staying within one realm: JavaScript!

CSS in JS – styled components

If you use React or Vue, then Styled components provides a radically new way of encapsulating the CSS and JS into a module: just write CSS in JS

Basically, it is a JS to CSS compiler, and in this regard, it is somewhat similar to SASS, it just uses JavaScript as a language – no need to learn another syntax, to run into different issues, to integrate a separate toolset.

```
// Create a Title component that'll render an
<h1> tag with some styles
const Title = styled.h1`
  font-size: 1.5em;
  text-align: center;
  color: palevioletred;
`;

// Create a Wrapper component that'll render a
<section> tag with some styles
const Wrapper = styled.section`
  padding: 4em;
  background: papayawhip;
`;

// Use Title and Wrapper like any other React
component — except they're styled!
render(
  <Wrapper>
    <Title>
      Hello World!
    </Title>
  </Wrapper>
);
```

Hello World!

Basically, that's it: CSS in tagged template literals…

and forget class names: you don't need it anymore!

A tagged template literal is just a fancy way of a function call with some extras.

this is a live editor in the styled components website, try it out!

Which to use?

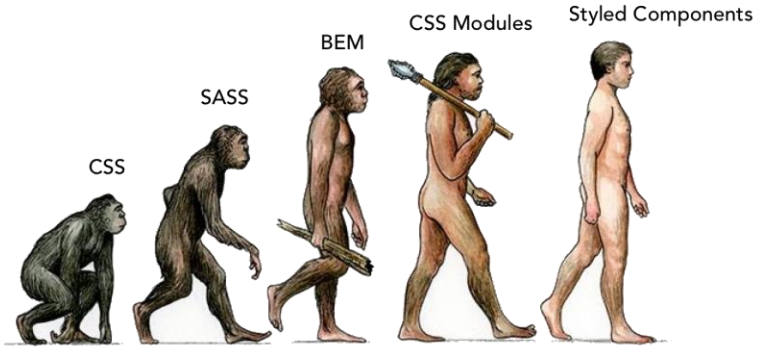
The bad news

The complexity always exists in the product level, not in the technology itself, so you will be a hard time using any of these – you cannot run away with complexity, you to need to live with it.

The good news

The complexity always exists in the product level; therefore, you will be able to use any of it, properly. Also, it depends on the JavaScript stack: *styled components* does not support Angular; if the project uses web-components, then probably it is easier to use simple CSS, because the CSS in the shadow DOM is already scoped.

Just to go back to the start of the history. ¯_(ツ)_/¯



edu_hu@epam.com