JAVASCRIPT*, HTML*, CSS*

CSS-модули

vitkarpov 3 ноября 2015 в 17:15 41,3k

Какими CSS обладает особенностями, которые приносят боль на больших проектах?

- глобальное пространство имен
- разрешение зависимостей
- поиск «мертвого» кода
- отсутствие констант
- неоднозначный результат (каскад)

Давайте разберемся с тем, как мы сейчас пишем CSS на больших проекта и как хотелось бы его писать в идеальном мире.

Возьмем простой пример: кнопка и ее состояния.

В реальности

Учитывая, что пространство имен глобальное, приходится вводить определенные соглашения по именованию классов, чтобы избежать случайных пересечений.

В БЭМ-нотации это бы выглядело так:

```
.button {...}
.button_state_disabled {...}
.button_state_error {...}
.button_state_progress {...}
```

Я думаю, что многие согласятся с тем, что при первом знакомстве с БЭМ когнитивный диссонанс вызывают огромные названия классов, которые получаются в итоге.

Естественно было бы написать:

```
.button {...}
.button.disabled {...}
```

Однако, через месяц, когда все забудут про это место, появится класс *.disabled* в другом файле(который будет означать совсем другое), а здесь неожиданно сломается — *пространство имен единое*.

Можно было бы написать так:

```
.button {...}
.button-disabled {...}
```

Но тогда получается слишком много дублирования кода, потому что кнопки отличаются всего одним стилем: *.button-disabled* должен содержать все то же, что и *.button*, но, например, другой цвет

фона.

Сейча эта задача решается с использованием миксинов на уровне препроцессоров, потому что в CSS нет такой возможности.

В идеальном мире

```
.button {
  display: inline-block;
  padding: 8px 2px;
  border-radius: 3px;
}

.button-disabled {
  composes: button;
  background-color: gray;
}
```

Все селекторы *локальные в рамках конкретного файла*.

Это означает, что в файле button.css, я пишу:

```
.text {...}
```

А мой коллега в недрах совсем другого компонента:

```
.text {…}
```

Нет пересечений для *.text* — нет необходимости в специальных классах для элементов блоков.

Локальное пространство имен справедливо так же для так же для анимаций, объявленных через *@keyframes*.

В шаблоне не хочется думать про композицию классов вида *.button.button_state_disabled* для получения определенного состояния.

Чтобы этого избежать, каждый класс должен содержать в себе все необходимое для отрисовки каждого состояния компонента:

```
.button-disabled {
  composes: base from "./base.css";
}
```

Ключевое слово *composes* дает мне функционал миксинов.

Причем я могу попросить стили из другого файла, что дает мне модульность на уровне CSS.

Реальность или вымысел

Выглядит неплохо. Что нужно для реализации такого интерфейса? Очевидно, необходимо установить связь между шаблонами и CSS.

Все зависит от того, какой шаблонизатор используется. В современном фронтенде практически все шаблонизаторы —

javascript-приложения, задача которых превратить шаблоны в html.

Представим, что у нас есть простой шаблонизатор, который умеет только интерполяцию строк:

```
<% var styles = require("./button.css") %>
<button class="<%=styles.button%>">Отправить заявку</button>
```

Весь CSS экспортируется как объект, ключами которого являются понятные, семантичные, имена классов для использования в шаблоне, а значениями — те имена классов, которые будут в итоговой разметке (например, уникальные хеши).

Сейчас это можно сделать с помощью плагина для webpack или плагина для browserify.

Более современный, реальный пример — в шаблоне reactjs-компонента:

```
import { Component } from 'react';
import styles from './button.css';

export default class button extends Component {
     render() {
        let className = styles.button
        let text = "Отправить заявку"

        if (this.state.loading) {
            className = styles.buttonDisabled
        }

        return <button className={className}>{text}</button>
    }
}
```

Что почитать

Кажется, видимая движуха началась с доклада «CSS in JS»

Статья CSS-модули: добро пожаловать в будущее. Прежде, чем читать, откройте исходный код, посмотрите на скомпилированные названия классов: красота! :)

Организация на гитхабе, где ребята штурмуют тему модульности в CSS. Здесь документация: примеры, концепции и конкретные инструменты: postcss, browserify и webpack плагины.

Доклад Павла Ловцевича на последнем WSD (слайды)

Похожие публикации

Стильный CSS переключатель без JavaScript

40

ИЗ ПЕСОЧНИЦЫ

nulldef • 6 апреля 2013 в 19:08

DocHub.io — удобный справочник HTML, CSS, Javascript

26

lsis • 8 декабря 2011 в 19:47

Прилипающий футер с не фиксированной высотой Css+Javascript

38

zorro1211 • 14 августа 2009 в 07:34

Популярное за сутки

Яндекс открывает Алису для всех разработчиков. Платформа Яндекс.Диалоги (бета)

69

BarakAdama • вчера в 10:52

Почему следует игнорировать истории основателей успешных стартапов

20

ПЕРЕВОД

m1rko • вчера в 10:44

Как получить телефон (почти) любой красотки в Москве, или интересная особенность MT_FREE

24

ИЗ ПЕСОЧНИЦЫ

cab404 • вчера в 20:27

Java и Project Reactor

10

zealot_and_frenzy • вчера в 10:56

Пользовательские агрегатные и оконные функции в PostgreSQL и Oracle

6

erogov • вчера в 12:46

Лучшее на Geektimes

Как фермеры Дикого Запада организовали телефонную сеть на колючей проволоке

31

NAGru • вчера в 10:10

Энтузиаст сделал новую материнскую плату для ThinkPad X200s

49

alizar • вчера в 15:32

Кто-то посылает секс-игрушки с Amazon незнакомцам. Amazon не знает, как их остановить

85

Pochtoycom • вчера в 13:06

Илон Маск продолжает убеждать в необходимости создания колонии людей на Марсе

139

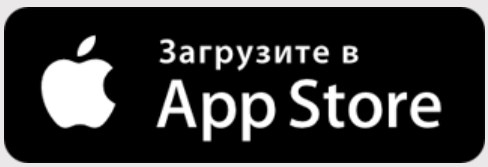
marks • вчера в 14:19

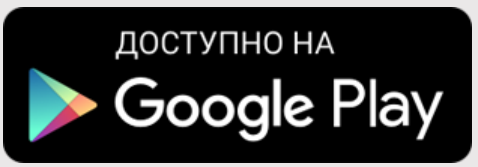
Дела шпионские (часть 1)

16

TashaFridrih • вчера в 13:16

Мобильное приложение





Полная версия