

CSS. Практика

Как пересесть с препроцессоров обратно на CSS.

MiniQ #14, 25.04.2019

Цель

Вернуться на CSS, захватив с собой "преимущества" препроцессоров

Почему CSS:

- Стандарт
- Активно развивается

Почему препроцессоры:

- Это удобно
- Элементы ЯП

Квант раздражения

1. Вычисляемые выражения

LESS

```
01. a {  
02.   font: 12pt/10pt sans-serif ;  
03. }  
04. .item-c {  
05.   grid-column: 3 / span 2 ;  
06.   grid-row: span 3 / 6 ;  
07. }
```

CSS

```
01. a {  
02.   font: 12pt/10pt sans-serif;  
03. }  
04. .item-c {  
05.   grid-column: 3 span 2;  
06.   grid-row: span 0.5;  
07. }
```

2. Экранирование функций

LESS

```
01. div {  
02.   width: calc(50% - 100px);  
03.   height: ~"calc(50% - 100px)";  
04. }
```

CSS

```
01. div {  
02.   width: calc(-50%);  
03.   height: calc(50% - 100px);  
04. }
```

3. Проблемы с Gulp

Пакет `gulp-less` не работает с Gulp 4, только 3.9.1



Невозможность обновиться на более новые версии



Раздражающие сообщения об уязвимостях и аудите

Основные понятия

Препроцессор

Свой язык, на выходе CSS



SASS



LESS



Stylus

- Расширяют язык стилей: примеси, вложенные правила
- Дополняют элементами языков программирования: условия, циклы, переменные

Постпроцессор

На входе CSS, на выходе CSS



PostCSS

- Вендорные префиксы
- Поддержка современных и будущих стандартов
- Генерация дополнительного кода

Инструментарий

В примерах использованы:

- `gulp 4` — Таск менеджер
- `gulp-postcss` — Постпроцессор
- `postcss-preset-env` — набор плагинов
- `postcss-import` — пример "нестандартного" плагина
- `postcss-rtl` — пример плагина-генератора кода

Подготовка

1. Подключаем

gulpfile.js

```
01. const { task, src, dest } = require( 'gulp' )  
02. const postcss = require( 'gulp-postcss' )  
03. const postcssPresetEnv = require( 'postcss-preset-env' )  
04. const atImport = require( 'postcss-import' )
```

2. Настраиваем

gulpfile.js

```
01. const postcssOptions = [  
02.   atImport(),  
03.   postcssPresetEnv({  
04.     stage: 0,  
05.     autoprefixer: {  
06.       grid: true,  
07.       browsers: [ 'last 2 versions', 'ie >= 11' ]  
08.     }  
09.   })  
10. ]
```

3. Запускаем

gulpfile.js

```
01. const css = () => {  
02.   return src( './src/assets/css/*.css' )  
03.     .pipe( postcss( postcssOptions ) )  
04.     .pipe( dest( './dest/assets/css' ) )  
05. }  
06. task( 'CSS', css )
```

1.

Хочу верстать
и не страдать

1. Grid Layout ❤️ IE10+

Исходный CSS

```
01. .layout {  
02.     display: grid;  
03.     grid-template-areas:  
04.         "item1 item1 item2"  
05.         "item3 item4 item4";  
06.     grid-template-columns: 1fr 1fr 1fr;  
07.     grid-template-rows: auto auto;  
08.     grid-gap: 20px;  
09. }
```


1. Grid Layout ❤️ IE10+

```
01. .layout {
02.   display: -ms-grid;
03.   display: grid;
04.   grid-template-areas:
05.     "item1 item1 item2"
06.     "item3 item4 item4";
07.   -ms-grid-columns: 1fr 20px 1fr 20px 1fr;
08.   grid-template-columns: 1fr 1fr 1fr;
09.   -ms-grid-rows: auto 20px auto;
10.   grid-template-rows: auto auto;
11.   grid-gap: 20px;
12. }
```

1. Grid Layout ❤️ IE10+

Что пока нельзя использовать:

- `grid-auto-columns`
- `grid-auto-rows`
- `grid-auto-flow`

Ограничения:

- Каждый grid-элемент сетки должен иметь уникальное имя
- Для использования `grid-gap` свойства `grid-template-areas` и `grid-template-columns` должны быть определены.

2.

Мама, я в Дубае

2. RTL: Генерация кода

Для гридов и флексов RTL не нужен!

Подключаем плагин:

```
01. const rtl = require( 'postcss-rtl' )  
02. src( 'style.css' )  
03.   .pipe( postcss( [ rtl( options ) ] ) )  
04.   .pipe( dest( './dest' ) )
```

2. RTL: Генерация кода

Исходный CSS

```
01. .foo {  
02.   float: right;  
03.   margin-left: 13px;  
04.   font-size: 13px;  
05. }
```

Обработанный CSS

```
01. .foo {  
02.   font-size: 13px;  
03. }  
04. [dir="ltr"] .foo {  
05.   float: right;  
06.   margin-left: 13px;  
07. }  
08. [dir="rtl"] .foo {  
09.   float: left;  
10.   margin-right: 13px;  
11. }
```

2. RTL: Псевдокласс `dir`

Исходный CSS

```
01. blockquote:dir rtl) {  
02.   margin-right: 10px;  
03. }  
  
04. blockquote:dir ltr) {  
05.   margin-left: 10px;  
06. }
```

Обработанный CSS

```
01. [dir="rtl"] blockquote {  
02.   margin-right: 10px;  
03. }  
  
04. [dir="ltr"] blockquote {  
05.   margin-left: 10px;  
06. }
```

3.

Мне бы в
HTTP/1.1

3. Модульность и объединение

Этот плагин должен быть первым в вашем списке плагинов!

Подключаем плагин:

```
const atImport = require( 'postcss-import' )
```


3. Модульность и объединение

```
01. @import url('vars.css');
```

```
02. @import url('common.css');
```

```
03. @import url('node_modules/swiper/dist/css/swiper.min.css');
```

```
04. @import url('https://fonts.googleapis.com/css?family=Roboto');
```

4.

Не переменные,
а **Custom Properties**

4. Custom properties

Исходный CSS

```
01. :root {  
02.   --fontSize: 1rem;  
03.   --mainColor: #12345678;  
04. }  
05. body {  
06.   color: var(--mainColor);  
07.   font-size: var(--fontSize);  
08. }
```

Обработанный CSS

```
01. :root {  
02.   --fontSize: 1rem;  
03.   --mainColor: rgba(18, 52, 86, 0.47);  
04. }  
05. body {  
06.   color: rgba(18, 52, 86, 0.47);  
07.   color: var(--mainColor);  
08.   font-size: 1rem;  
09.   font-size: var(--fontSize);  
10. }
```

4. Custom properties

Ограничения:

- Custom properties только в `:root` , иначе не будет обратной совместимости
- Custom properties в `@media` тоже без обратной совместимости

5.
We need
to go deeper

5. Вложенные селекторы

Исходный CSS

```
01. a {  
02.   color: white;  
03.   & span { color: green; }  
04.   @nest span & { color: blue; }  
05.   @media (min-width: 30em) {  
06.     color: yellow;  
07.   }  
08.   &:hover {  
09.     color: red;  
10.   }  
11. }
```

Обработанный CSS

```
01. a { color: white; }  
02. a span { color: green; }  
03. span a { color: blue; }  
04. @media (min-width: 30em) {  
05.   a { color: yellow; }  
06. }  
07. a:hover { color: red; }
```

5. Вложенные селекторы

Ограничения

Исходный CSS

```
01. .block {  
02.   color: white;  
03.   &__el { color: green; }  
04. }
```

Обработанный CSS

```
01. .block {  
02.   color: white;  
03.   &__el { color: green; }  
04. }
```

6. Media queries

6. Custom media queries

Исходный CSS

```
01. @custom-media --s (max-width: 480px);
02. @media (--s) {
03.   /* стили для телефона */
04.   width: 100%
05. }
```

Обработанный CSS

```
01. @media (max-width: 480px) {
02.   /* стили для телефона */
03.   width: 100%;
04. }
```

6. Media queries

Исходный CSS

```
01. @custom-media --only-tablet (width >= 768px) and (width < 1200px);  
02. @media (--only-tablet) {  
03.   /* планшет */  
04. }
```

Обработанный CSS

```
01. @media (min-width: 768px) and (max-width: 1199px) {  
02.   /* планшет */  
03. }
```

Песочницы

- <https://autoprefixer.github.io/> — Autoprefixer
- <https://preset-env.cssdb.org/playground> — PostCSS Preset Env

Ссылки

- <https://youtu.be/CaDnbOjXjRg> — Вадим Макеев, Мой ванильный CSS
- <https://youtu.be/g20pCKeSgUU> — Вероника Новикова, CSS ещё не торт

Конец,
или что дальше?

CSS. Практика

Бондаренко Юрий / bnd.by

Вопросы?