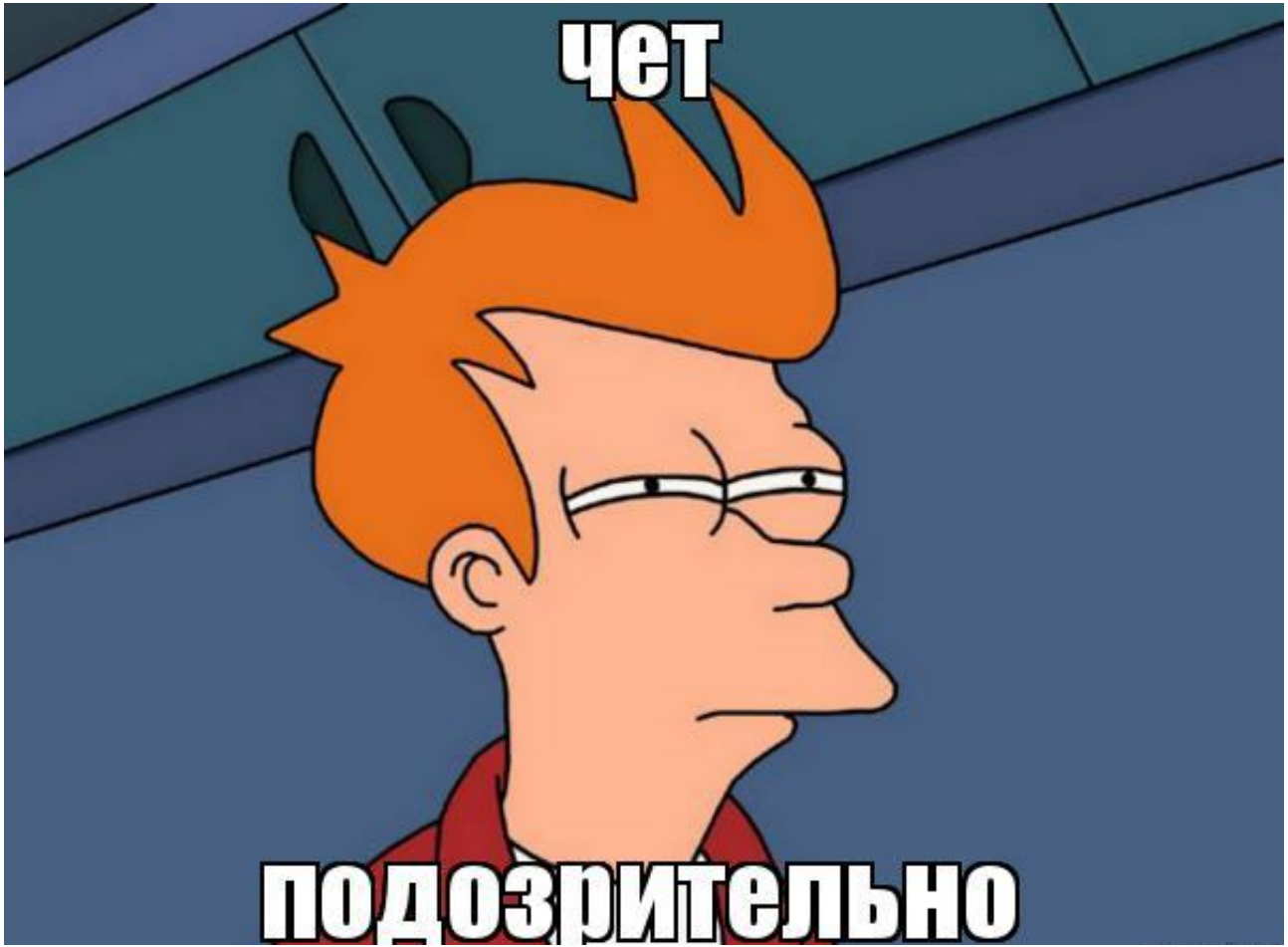
Как много девушек хороших: обзор препроцессоров и task runners для frontend разработки

Гилимханов Артур
НФ БашГУ



WAT???

Какие технологии крутятся на frontend?

Пакетные менеджеры

Установка

1) Установить https://nodejs.org/en/

2) Установить пакетный менеджер

https://yarnpkg.com/en/docs/install

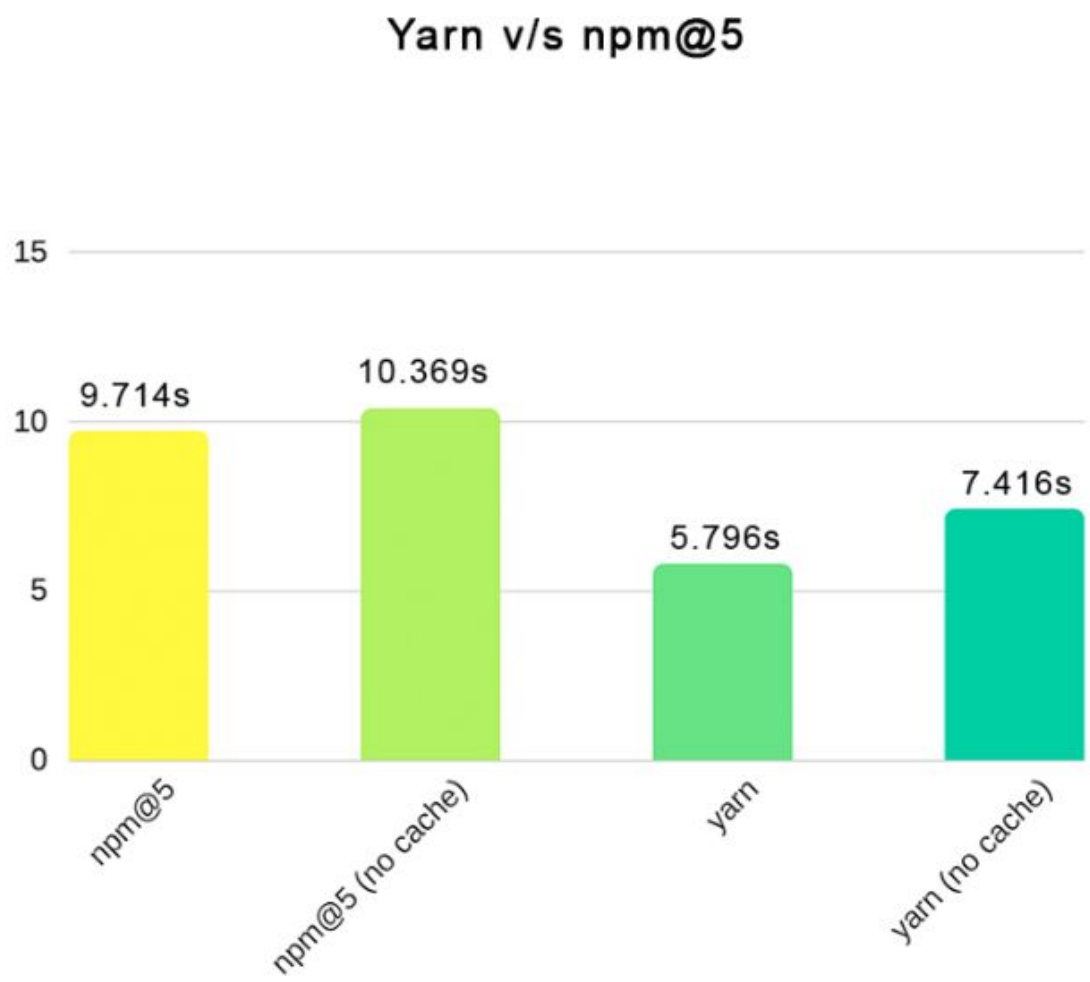
Ну тут больше делать ничего не надо, npm идет в комплекте с nodejs

Инициализация проекта	yarn init (-y)	npm init (-y)
Установка пакета	yarn add <package>	npm install <package> --save
Обновление пакета	yarn upgrade <package>	npm upgrade <package>
Удаление пакета	yarn remove <package>	npm uninstall <package>
Установка зависимостей	yarn / yarn install	npm i / npm install
Вывод списка зависимостей	yarn list (--depth=0)	npm ls (--depth=0)

Yarn может работать оффлайн

yarn add <package> --offline

yarn add <package>@version --offline


Yarn v/s npm@5
15
10
5
0
9.714s
10.369s
5.796s
7.416s
npm@5
npm@5 (no cache)
yarn
yarn (no cache)

HTML

HTML-препроцессоры

Haml

PugJS (Jade)

Плагины

Emmet

Haml

Установка:

Установить ruby, затем gem install haml

npm: npm i gulp-haml

yarn: yarn add gulp-haml

Haml

Syntax:

%h1 Hello World! - -> <h1>Hello World!</h1>
.content Hello World! - -> Hello World!
#main Yeah!!! --> Yeah!!!

%span{:class => “code”, :id => “main”} What’s up!
%span.code#main What’s up!

%span
 Multiline
 text

PugJS (Jade)

npm: npm i gulp-pug

yarn: yarn add gulp-pug

PugJS (Jade)

Syntax:

h1 Hello World! - -> <h1>Hello World!</h1>
.content Hello World! - -> Hello World!
#main Yeah!!! --> Yeah!!!

span(class=“code”, id=“main”) What’s up!
span.code#main What’s up!

PugJS (Jade)

Syntax:

```
span                        -->  <span>
 |some                      -->       some
 |text                      -->       text
                            -->  </span>
input(                      -->  <input type=”checkbox” name=”agreement>
  type=”checkbox”
  name=”agreement”
)
```

PugJS (Jade)

CASE

```
- var friends = 10
case friends
  when 0
    p you have no friends
  when 1
    p you have a friend
  default
    p you have #{friends} friends
```

```
<p>you have 10 friends</p>
```

CODE

```
for (var x = 0; x < 3; x++)
  li item
```

```
<li>item</li>
<li>item</li>
<li>item</li>
```

COMMENTS

```
//(-) some comment
```

```
<!--some comment-->
```

PugJS (Jade)

CONDITIONALS

```
- var authorised = false
#user
  if !authorised
    h2.green Not Authorised
  else if authorised
    h2.blue Authorised
```

```
<div id="user">
  <h2 class="green">Not Authorised</h2>
</div>
```

PugJS (Jade)

INCLUDES

```
//index.pug
include includes/head.pug
  body
    h1 some header

//head.pug
head
  title My Site
```

```
<head>
      <title>My Site</title>
</head>
<body>
      <h1>some header</h1>
</body>
```

PugJS (Jade)

ITERATION

```
ul
  each val, index in ['zero', 'one', 'two']
    li= index + ': ' + val
```

```
<ul>
  <li>0: zero</li>
  <li>1: one</li>
  <li>2: two</li>
</ul>
```

MIXINS

```
mixin pet(name)
  li.pet= name
ul
  +pet('cat')
  +pet('dog')
```

```
<ul>
  <li class="pet">cat</li>
  <li class="pet">dog</li>
</ul>
```

Emmet

Syntax:

> - вложенность
+ - следующий элемент на том же уровне
^ - расположить элемент на уровне выше
. - класс
- идентификатор
*(number) - дублировать элемент
() - группировка
{} - текст внутри тега
[] - аттрибуты
$(@-) - заменяется на цифры

CSS

CSS-препроцессоры

Плагины

Emmet

Установка:

npm: npm i gulp-sass

yarn: yarn add gulp-sass

Установка через app:

compass.app, koala, livereload, prepros, scout-app

Syntax:

Variables:

```
$main-color: red;
body {
  color: $main-color;
}
```

```
body {
  color: red;
}
```

Nesting:

```
body {
  font-size: 14px;
      .content {
        line-height: 1px;
      }
}
```

```
body {
  font-size: 14px;
}

body .content {
  line-height: 1px;
}
```

Syntax:

Partials:
_partials.scss
@import “partials”

Mixins:

```
@mixin border-radius($radius) {
 -webkit-border-radius: $radius;
    -moz-border-radius: $radius;
     -ms-border-radius: $radius;
         border-radius: $radius;
}
.box { @include border-radius(10px); }
```

Syntax:

Extends:

```
  %message-shared {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

  .success {
  @extend %message-shared;
  border-color: green;
}
```

Operators:

+ - / * %

{less}

Установка:

npm: npm i gulp-less

yarn: yarn add gulp-less

{less}

Syntax:

Variables:

```
@main-color: red;
body {
  color: @main-color;
}

body {
  color: red;
}
```

Nesting:

```
.main {
  font-size: 14px;
      &_content {
        line-height: 1px;
      }
}

.main {
  font-size: 14px;
}

.main_.content {
  line-height: 1px;
}
```

{less}

Syntax:

Merge:

```
.mixin() {
  box-shadow+: inset 0 0 10px #555;
}
.myclass {
  .mixin();
  box-shadow+: 0 0 20px black;
}

.myclass {
  box-shadow: inset 0 0 10px #555, 0 0 20px black;
}
```

{less}

Syntax:

Mixins:

```
.my-hover-mixin() {
  &:hover {
    border: 1px solid red;
  }
}
button {
  .my-hover-mixin();
}
```

Imports:

```
@import (less) "foo.css";
```

Установка:

npm: npm i gulp-stylus

yarn: yarn add gulp-stylus

Syntax:

Variables:

```
main-color = #f1f1f1
body {
  color: main-color
}
```

```
body {
  color: #f1f1f1;
}
```

.
[]
! ~ + -
is defined
** * / %
+ -
... ..
<= >= < >
in
== is != is not isnt
is a
&& and || or
?:
= := ?= += -= *= /= %=
not
if unless

Syntax:

Mixins:

```
 border-radius(n)
  -webkit-border-radius n
  -moz-border-radius n
  border-radius n

form input[type=button]
  border-radius(5px)
```

```
 form input[type=button] {
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;
}
```

Syntax:

Functions:

```
 add(a, b = a)
   a + b

 add(10, 5)
 // => 15

 add(10)
 // => 20
```

Syntax:

@import and @require:

```
@require ‘header’

@import ‘footer.css’
```

@extend:

```
.message {
  padding: 10px;
  border: 1px solid #eee;
}

.warning {
  @extend .message;
  color: #E2E21E;
}
```

Emmet

Syntax:

+ - добавить свойство

```
* {
   m0+p0
}
```

```
* {
   margin: 0;
   padding: 0;
}
```

PostCSS

Increase code readability

Add vendor prefixes to CSS rules using values from Can I Use. Autoprefixer will use the data based on current browser popularity and property support to apply prefixes for you.

CSS input

```
:fullscreen {
}
```

CSS output

```
:-webkit-:full-screen {
}
:-moz-:full-screen {
}
:full-screen {
}
```

CSS MODULES

The end of global CSS

CSS Modules means you never need to worry about your names being too generic, just use whatever makes the most sense.

CSS input

```
.name {
  color: gray;
}
```

CSS output

```
.Logo__name__SVK0g {
  color: gray;
}
```

CSS input

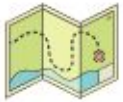
```
a {
  color: #d3;
}
```

Console output

```
app.css
2:10 Invalid hex color
```

Avoid errors in your CSS

Enforce consistent conventions and avoid errors in your stylesheets with stylelint, a modern CSS linter. It supports the latest CSS syntax, as well as CSS-like syntaxes, such as SCSS.



Powerful grid system

LostGrid makes use of calc() to create stunning grids based on fractions you define without having to pass a lot of options.

CSS input

```
div {
  lost-column: 1/3
}
```

CSS output

```
div {
  width: calc(99.9% * 1/3 -
  (30px - 30px * 1/3));
}
div:nth-child(1n) {
  float: left;
  margin-right: 30px;
  clear: none;
}
```

CSS input

```
:root {
  --red: #d33;
}
a {
  &:hover {
    color: color(var(--red) a(54%));
  }
}
```

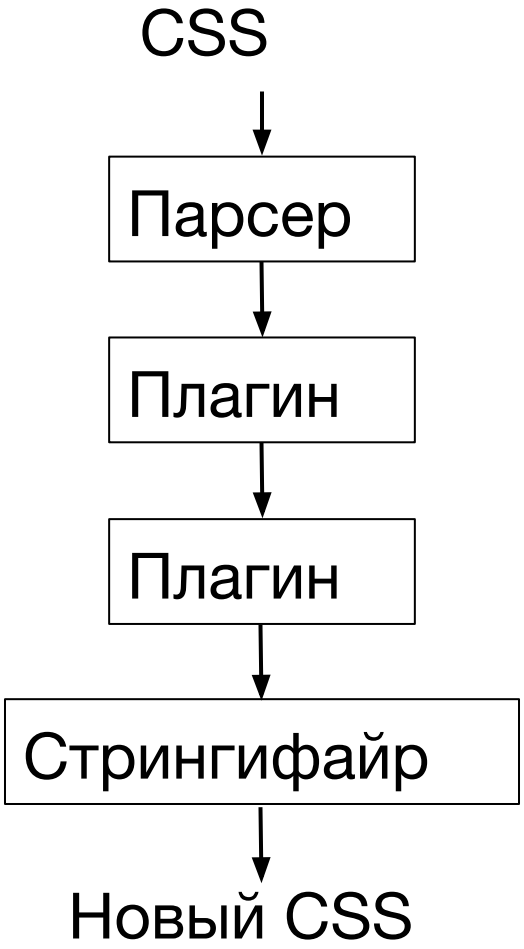
CSS output

```
a:hover {
  color: #dd3333;
  color: rgba(221, 51, 51, 0.54);
}
```

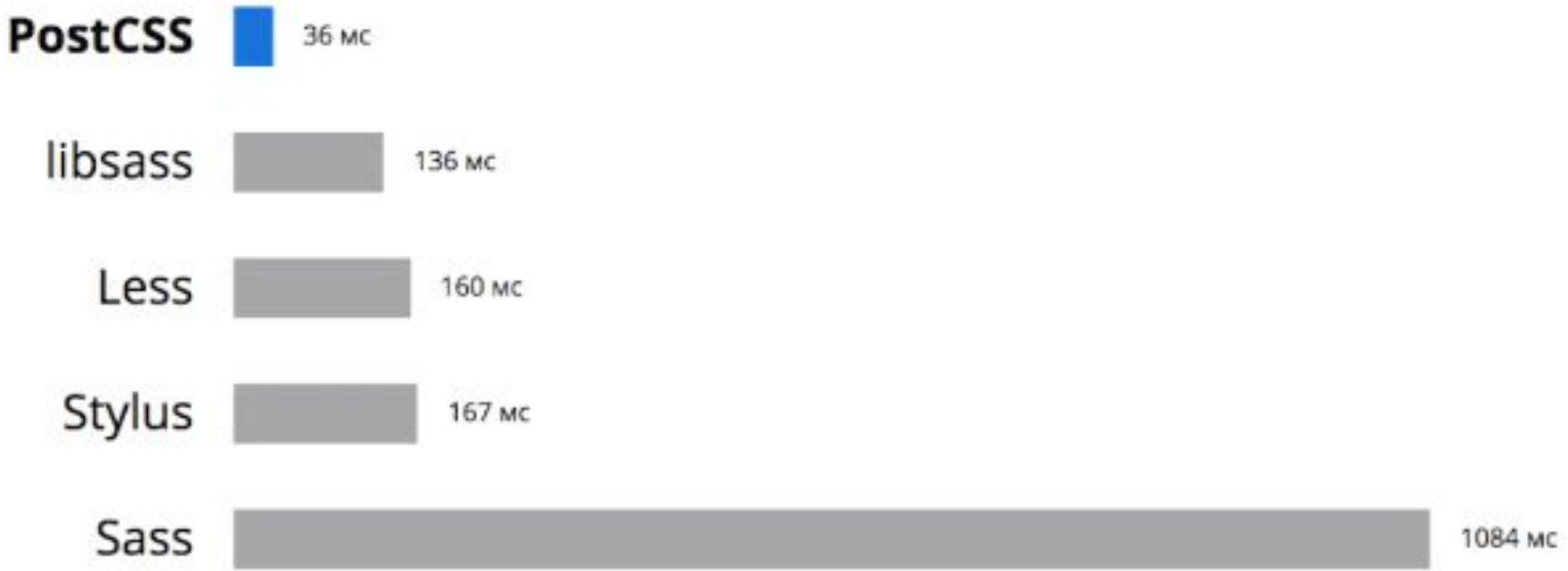
Use tomorrow's CSS, today!

Write future-proof CSS and forget old preprocessor specific syntax. Use the latest CSS syntax today with cssnext. It transforms CSS specs into more compatible CSS so you don't need to wait for browser support.

- automatic vendor prefixes
- custom properties set & `@apply`
- custom media queries
- custom selectors
- `image-set()` function
- `hwb()` function
- `#rrggbbaa` colors
- `rebeccapurple` color
- `filter` property (svg fallback)
- `rem` unit (px fallback)
- `:matches` pseudo-class
- `::`pseudo syntax (`:` fallback)
- attribute case insensitive
- `hsl()` function (functional-notation)
- custom properties & `var()`
- reduced `calc()`
- media queries ranges
- nesting
- `color()` function
- `gray()` function
- `rgba` function (rgb fallback)
- `font-variant` property
- `initial` value
- `:any-link` pseudo-class
- `:not` pseudo-class (to l.3)
- `overflow-wrap` property (`word-wrap` fallback)
- `rgb()` function (functional-notation)
- `system-ui` font-family (font-family fallback)


CSS
Парсер
Плагин
Плагин
Стрингифайр
Новый CSS

Скорость


PostCSS
36 мс
libsass
136 мс
Less
160 мс
Stylus
167 мс
Sass
1084 мс

Преимущества:

1. Скорость
2. Модульность
3. Функции, невозможные на Sass

Новый проект

```
.pipe( postcss([
    require('postcss-nested'),
    require('postcss-mixins'),
    require('postcss-simple-vars'),
    require('autoprefixer'),
    require('postcss-easings'),
    require('cssnext')
]) )
```

Разница

Препроцессор

— Монолитный

— Логика прямо в шаблоне

PostCSS

— Все функции как плагины

— JS трансформирует CSS

Task runners

Grunt

Grunt

Установка:

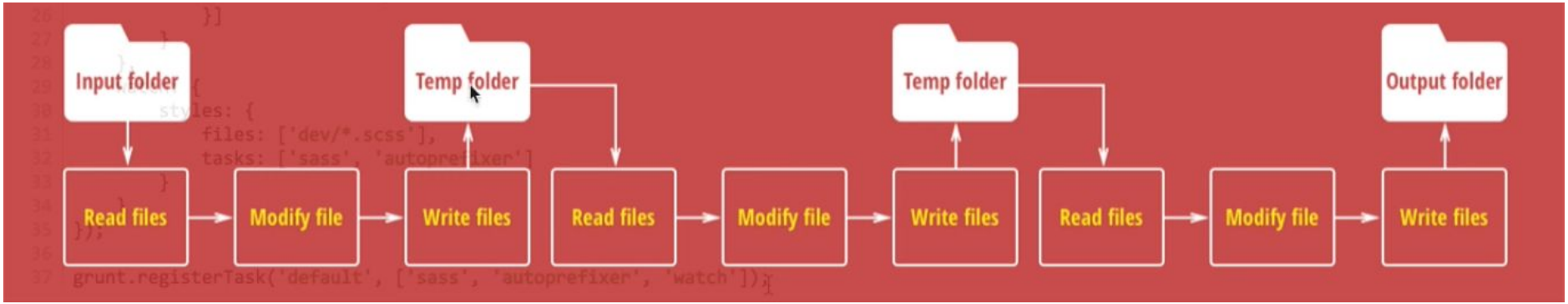
npm: npm i grunt-cli -g
npm i grunt -D
touch Gruntfile.js

yarn: yarn add grunt
touch Gruntfile.js

Grunt

```
'use strict';

const grunt = require('grunt');

grunt.loadNpmTasks('grunt-sass');
grunt.loadNpmTasks('grunt-autoprefixer');

grunt.initConfig({
    sass: {
        dist: {
            files: [{
                src: 'dev/*.scss',
                expand: true,
                dest: '.tmp/styles',
                ext: '.css'
            }]
        }
    },
    autoprefixer: {
        dist: {
            files: [{
                src: '{,*/}*.css',
                expand: true,
                cwd: '.tmp/styles',
                dest: 'css/styles'
            }]
        }
    },
    watch: {
        styles: {
            files: ['dev/*.scss'],
            tasks: ['sass', 'autoprefixer']
        }
    }
});

grunt.registerTask('default', ['sass', 'autoprefixer', 'watch']);
```

Grunt


Input folder
Temp folder
Temp folder
Output folder
Read files
Modify file
Write files
Read files
Modify file
Write files
Read files
Modify file
Write files

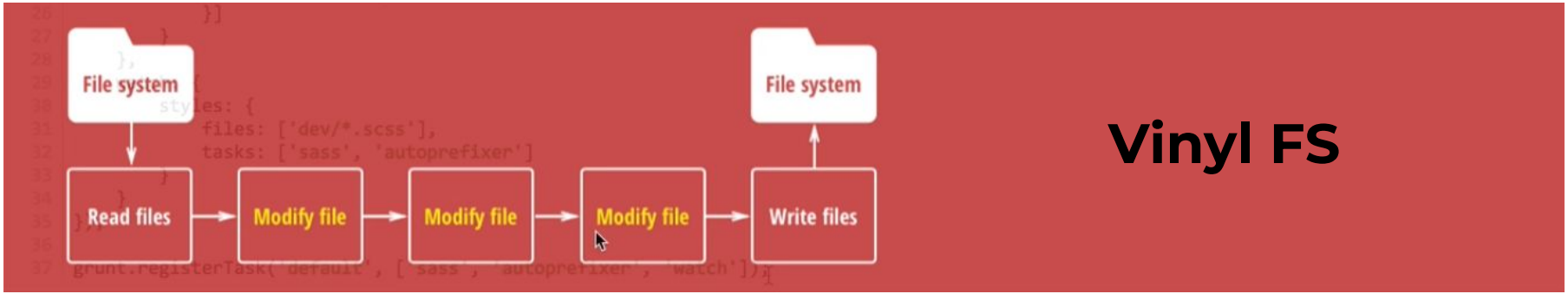
Установка:

npm: npm i gulp-cli -g
npm i gulp -D
touch gulpfile.js

yarn: yarn add gulp
touch gulpfile.js

```
'use strict';

const gulp = require('gulp');
const sass = require('gulp-sass');
const autoprefixer = require('gulp-autoprefixer');

gulp.task(function sass() {
  return gulp.src('dev/*.scss')
    .pipe(sass())
    .pipe(autoprefixer())
    .pipe(gulp.dest('css/styles'));
});

gulp.task('default', gulp.series('sass', function() {
  gulp.watch('dev/*.scss', gulp.series('sass'));
});
```

Gulp


File system
Read files
Modify file
Modify file
Modify file
Write files
File system

Vinyl FS

gulp.src(globs[, options])

Emits files matching provided glob or an array of globs. Returns a stream of Vinyl files that can be piped to plugins.

```
gulp.src('client/templates/*.jade')
  .pipe(jade())
  .pipe(minify())
  .pipe(gulp.dest('build/minified_templates'));
```

gulp.dest(path[, options])

Can be piped to and it will write files. Re-emits all data passed to it so you can pipe to multiple folders. Folders that don't exist will be created.

```
gulp.src('./client/templates/*.jade')
  .pipe(jade())
  .pipe(gulp.dest('./build/templates'))
  .pipe(minify())
  .pipe(gulp.dest('./build/minified_templates'));
```

gulp.task(name [, deps, fn])

Define a task using Orchestrator.

```
gulp.task('somename', function() {
  // Do stuff
});
```

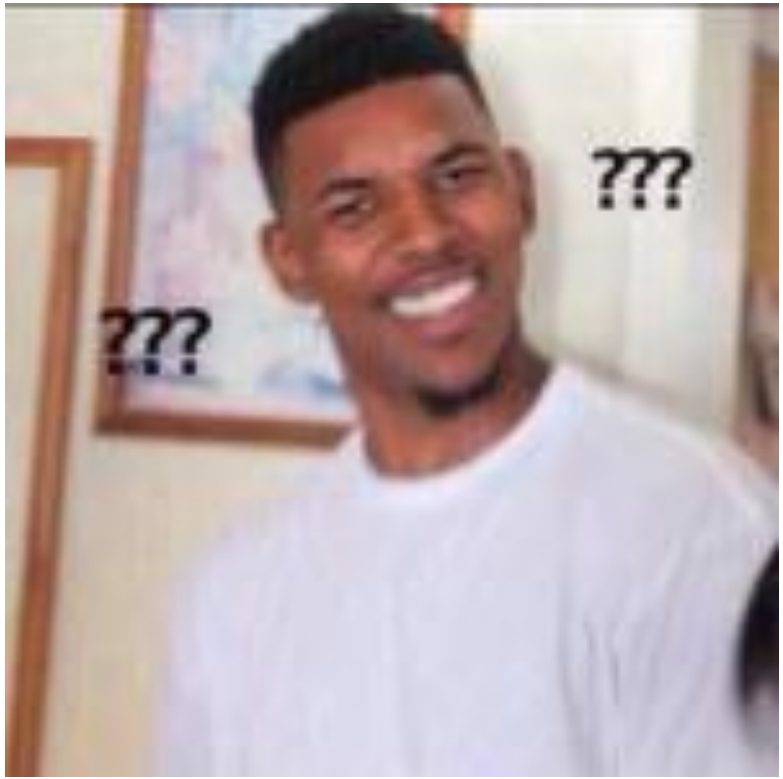
gulp.watch(glob [, opts], tasks) or gulp.watch(glob [, opts, cb])

Watch files and do something when a file changes. This always returns an EventEmitter that emits `change` events.

gulp.watch(glob[, opts], tasks)

```
var watcher = gulp.watch('js/**/*.js', ['uglify','reload']);
watcher.on('change', function(event) {
  console.log('File ' + event.path + ' was ' + event.type + ', running tasks...');
});
```

Webpack



Вопросы?

Ссылки:

https://nodejs.org/en/
https://yarnpkg.com/en/
https://www.npmjs.com/
http://haml.info/
https://pugjs.org/api/getting-started.html
https://emmet.io/
https://sass-lang.com/
http://lesscss.org/
http://stylus-lang.com/
http://postcss.org/
https://www.postcss.parts/
https://gruntjs.com/
https://gulpjs.com/
https://webpack.js.org/