

Kurs Front-End **Developer**
Preprocesory CSS - SASS

PREPROCESORY

Preprocesory CSS to narzędzie pozwalające tworzyć oraz utrzymywać arkusze stylów w sposób dużo lepiej zorganizowany i czytelny. Dzięki nim możemy lepiej zarządzać arkuszem i szybciej dokonywać w nich zmian.

SASS (Syntactical Awesome Stylesheet) jest językiem skryptowym, którego kod przetwarzany jest do plików wynikowych kaskadowych arkuszy stylów - CSS.

Wprowadza on całkiem sporą liczbę udogodnień dla programisty **począwszy** od struktur zagnieżdżonych, **poprzez** mixiny, zmienne, **kończąc** na pętlach zarówno warunkowych jak i iteracyjnych.

INSTALACJA ŚRODOWISKA

Do skompilowania plików napisanych w SASS wykorzystamy narzędzie do automatyzacji pracy jakim jest GULP, którego komponenty pozwalają na pracę z preprocesorami CSS

Do działania GULP wymaga silnika `node.js`. Należy go pobrać i zainstalować ze strony - <https://nodejs.org/en/>

Praca z GULPem odbywa się z wykorzystaniem konsoli.

Pierwszym krokiem jest globalna instalacja GULPa za pomocą komendy `npm install gulp-cli -g` (Windows) lub `sudo npm install gulp-cli -g` (OSX)

INSTALACJA ŚRODOWISKA (I-****)

Następnie stworzymy folder roboczy i w tym folderze inicjujemy nowy projekt, który stworzy plik informacyjny o naszym projekcie w formacie json

```
npm init
```

```
{
  "name": "project",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this ok? (yes) █
```

INSTALACJA ŚRODOWISKA (I-***)

Następnie instalujemy wersję GULPa dla naszego projektu cały czas będąc w folderze roboczym

```
npm install gulp --save-dev
```

Dodatkowo w folderze roboczym tworzymy folder `app` – w którym będziemy przechowywać nasz projekt oraz folderze głównym projektu tworzymy plik `gulpfile.js`, który jest plikiem konfiguracyjnym GULPa

Aby uruchomić kompilowanie SASS w GULPie należy zainstalować odpowiednie rozszerzenie

```
npm install gulp-sass --save-dev
```

INSTALACJA ŚRODOWISKA (I-****)

PRZYKŁADOWA STRUKTURA PROJEKTU

```
| - folder-projektu
  | - app
    | - index.html
    | - js
      | - main.js
    | - scss
      | - styles.scss
    | - css
      | - styles.css
  | - gulpfile.js
  | - package.json
  | - node_modules
```

INSTALACJA ŚRODOWISKA (I-****)

Tworzymy ręcznie plik konfiguracyjny `gulpfile.js`

```
let gulp = require('gulp');
let sass = require('gulp-sass');

gulp.task('sass', () => {
  return gulp.src('app/scss/**/*.scss')
    .pipe(sass())
    .pipe(gulp.dest('app/css'));
});

gulp.task('watch', () => {
  gulp.watch('app/scss/**/*.scss', gulp.series('sass'));
});
```

INSTALACJA ŚRODOWISKA (I-***)

Aby uruchomić śledzenie zmian i automatyczne kompilowanie plików css wykorzystujemy komendę

```
gulp watch
```

Więcej możliwości GULPa i jego konfigurację znajdziecie pod adresem

<https://css-tricks.com/gulp-for-beginners/>

SASS ZMIENNE (I-***)

Zmienne w SASS definiujemy za pomocą znaku \$

Następnie deklarujemy nazwę zmiennej oraz jej wartość.

```
$base-color: #220000;
```

Tak zadeklarowaną zmienną możemy wykorzystywać wielokrotnie w projekcie.

SASS ZMIENNE (I-***)

SCSS

```
$base-color: #220000;
```

```
body {  
  color: $base-color;  
}
```

CSS

```
body {  
  color: #220000;  
}
```

SASS ZAGNIEŻDŻANIE (I-***)

Pisząc arkusz za pomocą preprocesorów należy trzymać się zasady DRY czyli don't repeat yourself (nie powtarzaj się). Aby praktykować tę zasadę kod należy pisać w zagnieżdżonych instrukcjach.

SCSS

```
nav {  
  ul {  
    margin: 0;  
    padding: 0;  
    list-style: none;  
  
    li {  
      display: inline-block;  
    }  
  }  
}
```

SASS ZAGNIEŻDŻANIE (I-***)

CSS

```
nav ul {  
    margin: 0;  
    padding: 0;  
    list-style: none;  
}  
  
nav ul li {  
    display: inline-block;  
}
```

SASS PARTIALS (I-***)

Poszczególne style dla elementów naszego projektu możemy definiować w osobnych plikach. Takie pliki mają nazwę zaczynającą się od znaku podkreślenia „_”.

Przykład:

```
_zmienne.scss
```

Następnie taki plik możemy dołączyć do innych plików naszego projektu za pomocą słowa kluczowego @import

```
@import "zmienne";
```

SASS MIXINS (I-****)

W przypadku powtarzających się zestawów warunków i ich wartości możemy tworzyć tak zwane `mixin`y. Pozwalają one raz zdefiniować zestaw właściwości i wartości, a następnie inkludować go wewnątrz reguły.

//Definicja

```
@mixin nazwa {  
    zestaw właściwości i wartości  
}
```

//Wywołanie w selektorze

```
.single-item {  
    @include nazwa;  
}
```

SASS MIXINS (I-****)

BEGIN NAVIGATION

SCSS

```
@mixin absolute-center {  
  position: absolute;  
  transform: translateY(-50%);  
  top: 50%;  
  right: 0;  
  left: 0;  
  margin: 0 auto;  
  text-align: center;  
}
```

```
.header-caption {  
  @include absolute-center;  
}
```

CSS

```
.header-caption {  
  position: absolute;  
  transform: translateY(-50%);  
  top: 50%;  
  right: 0;  
  left: 0;  
  margin: 0 auto;  
  text-align: center;  
}
```

SASS MIXINS (I-****)

Definiując mixin możemy dynamicznie przypisywać wartości do właściwości za pomocą parametrów:

SCSS

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

CSS

```
.box {  
  -webkit-border-radius: 10px;  
  -moz-border-radius: 10px;  
  -ms-border-radius: 10px;  
  border-radius: 10px;  
}
```


WARSZTATY – Landing Page Finanse

Zakoduj Landing Page z branży finansowej. Wykorzystaj preprocesor SASS i narzędzie automatyzacji zadań GULP.

Fonty:

- Open Sans

Kolory:

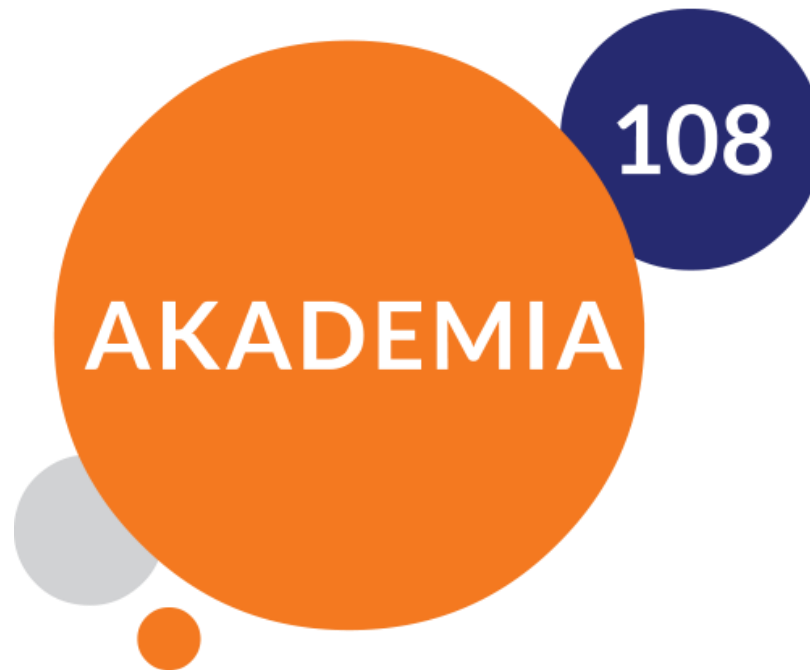
- Niebiesko-Granatowy - #001d8a
- Żółty - #ecd407
- Szary - #292929

Podgląd projektu

- <https://akademia108.pl/kurs-front-end/landing-finanse.png>

Grafiki do projektu dostępne są pod linkiem:

- <https://akademia108.pl/kurs-front-end/grafika-landing-finanse.zip>



Akademia 108

<https://akademia108.pl>