2007

2019

Programowanie dla iOS

TTRules

wersja 2020

TTRules 2020

TTRules 2020

- Dostosowanie starej aplikacji do równie starej specyfikacji:

TTRules 2020

- Dostosowanie starej aplikacji do równie starej specyfikacji:
- Pliki danych w pełni niezależne od aplikacji przechowywane w pliku ZIP

TTRules 2020

- Dostosowanie starej aplikacji do równie starej specyfikacji:
  - Pliki danych w pełni niezależne od aplikacji przechowywane w pliku ZIP
  - Ściąganie aktualizacji z sieci

TTRules 2020

TTRules 2020

- Zmiany w projekcie - usuwamy wszystkie pliki z danymi z pakietu

TTRules 2020

- Zmiany w projekcie - usuwamy wszystkie pliki z danymi z pakietu
- Pakujemy je do pliku ZIP i dodajemy go do pakietu

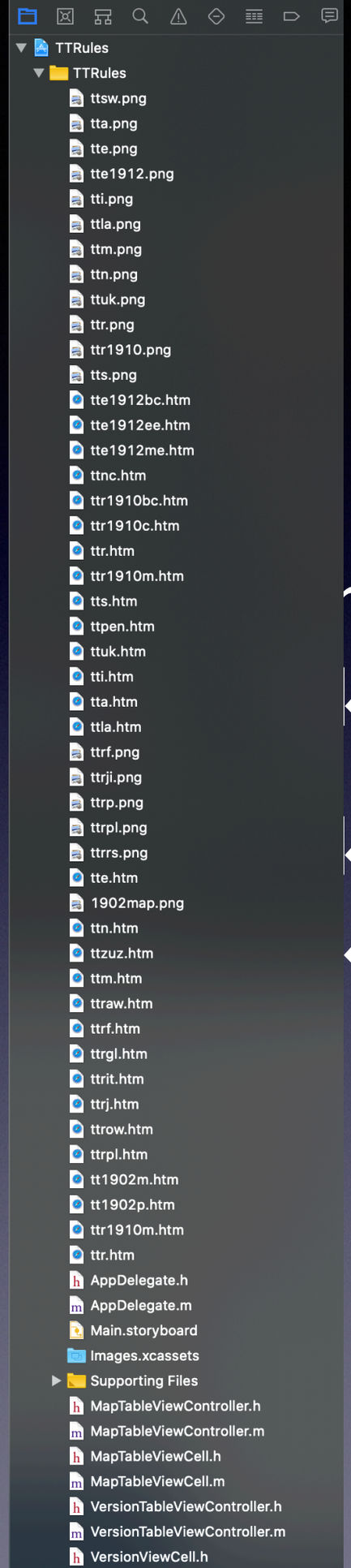
TTRules 2020

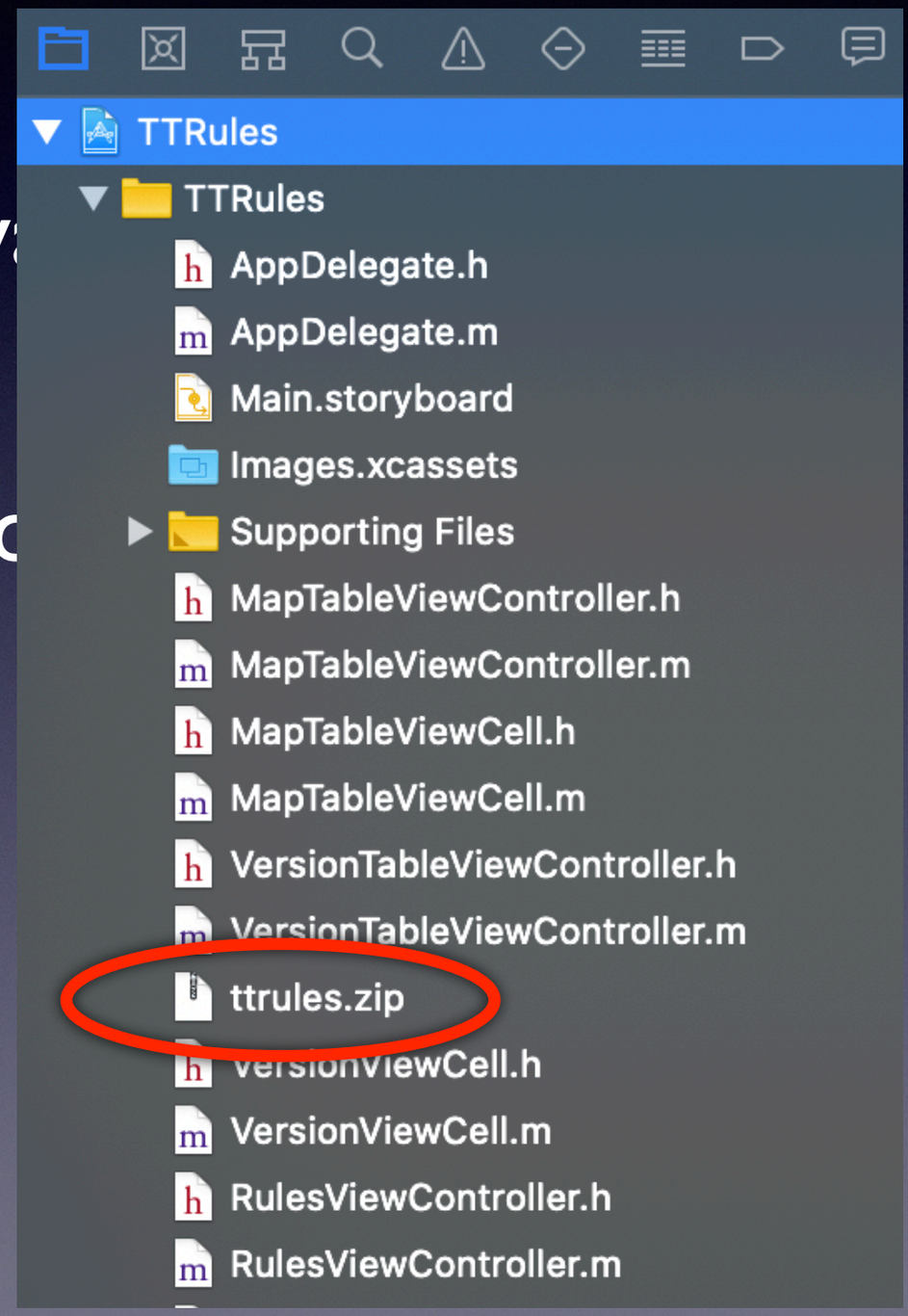
niany w projekcie - usuw

ki z danymi z pakietu

kujemy je do pliku ZIP i c

kietu





TTRules 2020

- Zmiany w projekcie - usuwamy wszystkie pliki z danymi z pakietu
- Pakujemy je do pliku ZIP i dodajemy go do pakietu

TTRules 2020

- Zmiany w projekcie - usuwamy wszystkie pliki z danymi z pakietu
- Pakujemy je do pliku ZIP i dodajemy go do pakietu
- W pliku ZIP dodajemy plik ttrv.plist, w którym będzie informacja o wersji

TTRules 2020

- Zmiany w projekcie - usuwamy wszystkie pliki z danymi z pakietu
- Pakujemy je do pliku ZIP i dodajemy go do pakietu
- W pliku ZIP dodajemy plik ttrv.plist, w którym będzie informacja o wersji

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<array>
    <integer>1</integer>
    <string>2020.04.20</string>
</array>
</plist>
```

TTRules 2020

TTRules 2020

- Musimy rozpakować ZIP'a

TTRules 2020

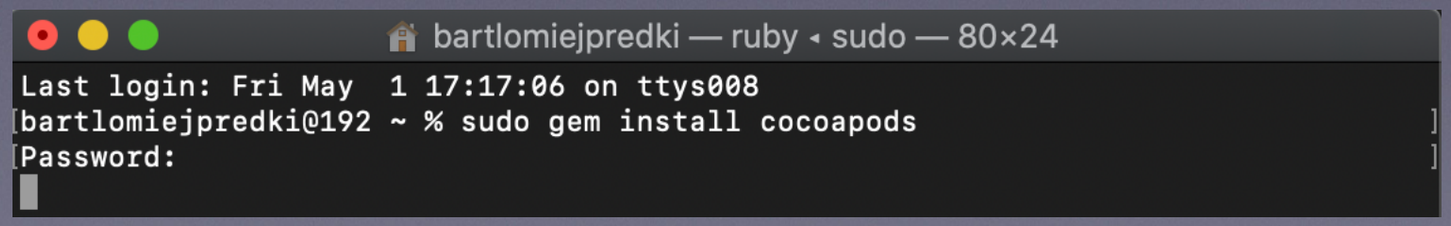
- Musimy rozpakować ZIP'a
- Możemy wykorzystać gotową bibliotekę SSZipArchive (https://github.com/ZipArchive/ZipArchive)

TTRules 2020

- Musimy rozpakować ZIP'a
- Możemy wykorzystać gotową bibliotekę SSZipArchive (https://github.com/ZipArchive/ZipArchive)
- Musimy ją dodać do naszego projektu - wykorzystamy CocoaPods (https://cocoapods.org)

TTRules 2020

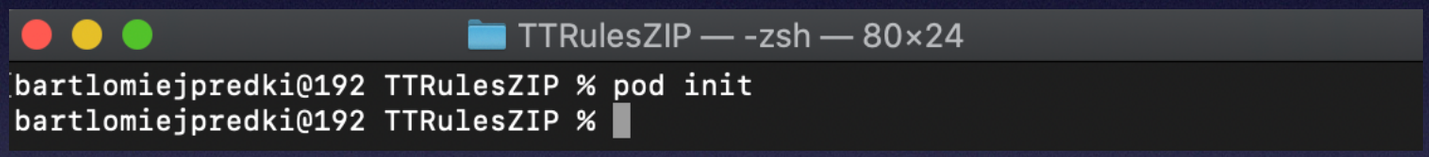
- Musimy rozpakować ZIP'a
- Możemy wykorzystać gotową bibliotekę SSZipArchive (https://github.com/ZipArchive/ZipArchive)
- Musimy ją dodać do naszego projektu - wykorzystamy CocoaPods (https://cocoapods.org)



TTRules 2020

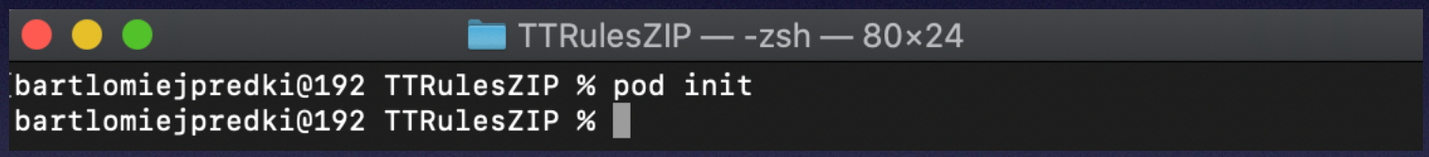
TTRules 2020

- Po zainstalowaniu CocoaPods wchodzimy do katalogu z naszym projektem i wydajemy polecenie:



TTRules 2020

- Po zainstalowaniu CocoaPods wchodzimy do katalogu z naszym projektem i wydajemy polecenie:



- Powoduje to powstanie pliku Podfile w katalogu projektu, do którego dopisujemy:

```
# Uncomment the next line to define a global platform for your project
# platform :ios, '9.0'

target 'TTRules' do
  # Comment the next line if you don't want to use dynamic frameworks
  use_frameworks!

  # Pods for TTRules
  pod 'SSZipArchive'

  target 'TTRulesTests' do
    inherit! :search_paths
    # Pods for testing
  end

end
```

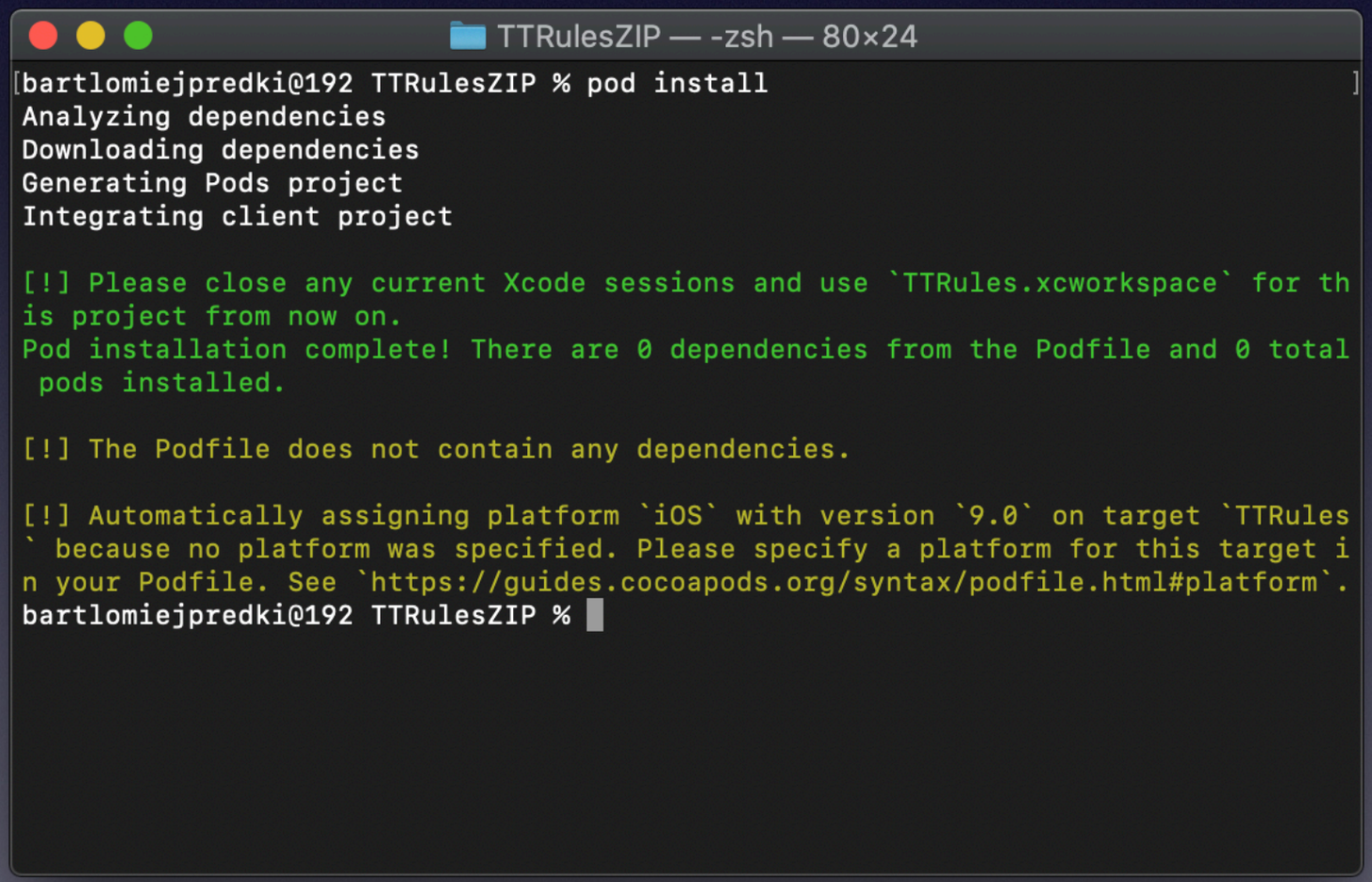
TTRules 2020

TTRules 2020

- Teraz wydajemy polecenie pod install

TTRules 2020

- Teraz wydajemy polecenie pod install

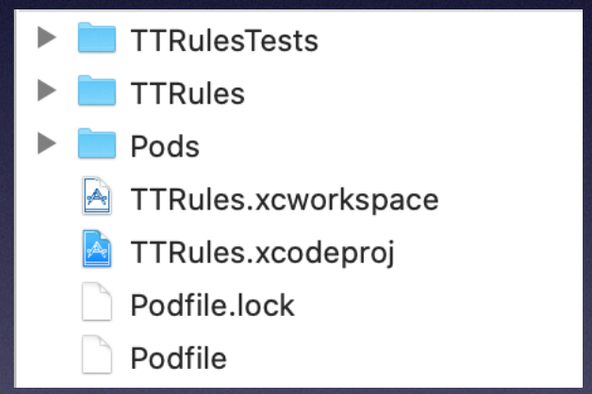


TTRules 2020

- Teraz wydajemy polecenie pod install

TTRulesZIP — -zsh — 80×24

```
[bartlomiejpredki@192 TTRulesZIP % pod install
Analyzing dependencies
Downloading dependencies
Generating Pods project
Integrating client project

[!] Please close any current Xcode sessions and use `TTRules.xcworkspace` for th
is project from now on.
Pod installation complete! There are 0 dependencies from the Podfile and 0 total
 pods installed.

[!] The Podfile does not contain any dependencies.

[!] Automatically assigning platform `iOS` with version `9.0` on target `TTRules
` because no platform was specified. Please specify a platform for this target i
n your Podfile. See `https://guides.cocoapods.org/syntax/podfile.html#platform`.
bartlomiejpredki@192 TTRulesZIP %
```

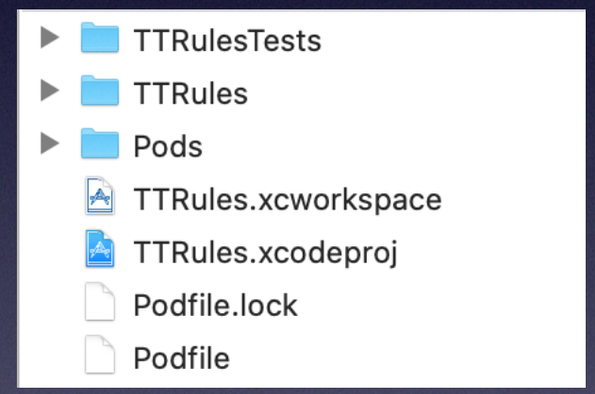


TTRules 2020

- Teraz wydajemy polecenie pod install

TTRulesZIP — -zsh — 80×24

```
[bartlomiejpredki@192 TTRulesZIP % pod install
Analyzing dependencies
Downloading dependencies
Generating Pods project
Integrating client project

[!] Please close any current Xcode sessions and use `TTRules.xcworkspace` for th
is project from now on.
Pod installation complete! There are 0 dependencies from the Podfile and 0 total
 pods installed.

[!] The Podfile does not contain any dependencies.

[!] Automatically assigning platform `iOS` with version `9.0` on target `TTRules
` because no platform was specified. Please specify a platform for this target i
n your Podfile. See `https://guides.cocoapods.org/syntax/podfile.html#platform`.
bartlomiejpredki@192 TTRulesZIP %
```



Od teraz otwieramy plik z rozszerzeniem xcwofrkspace

TTRules 2020

TTRules 2020

- Modyfikujemy metodę viewDidLoad w klasie MapTableViewController
- Ekstrahujemy kod ładujący dane do osobnej metody

```objc
- (void)loadData {
    NSString *docPath =[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
        NSUserDomainMask, YES) firstObject];

    NSString *plistCatPath = [[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
        NSUserDomainMask, YES) firstObject] stringByAppendingPathComponent:@"maps.plist"];
    NSFileManager *fileManager = [NSFileManager defaultManager];


    if (![fileManager fileExistsAtPath:plistCatPath]) {
        //rozpakuj z ZIP'a
        NSString *zipPath = [[NSBundle mainBundle] pathForResource:@"ttrules" ofType:@"zip"];
        if ([fileManager fileExistsAtPath:zipPath]) {
            [SSZipArchive unzipFileAtPath:zipPath toDestination:docPath];
        }

    }
    dict = [[NSDictionary alloc] initWithContentsOfFile:plistCatPath];
    plistCatPath = [[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
        NSUserDomainMask, YES) firstObject] stringByAppendingPathComponent:@"Icons.plist"];
    icons=[[NSDictionary alloc] initWithContentsOfFile:plistCatPath];

    maps=[[NSMutableDictionary alloc] init];
    self.navigationItem.title=@"Select map";
    self.mapArray = dict[@"Maps"];

    for (int i=0; i<_mapArray.count; i++) {
        NSString* map=self.mapArray[i][@"Name"];
        NSDictionary* mapVersions=dict[map];
        maps[map]=[NSNumber numberWithLong:mapVersions.count];
    }

    //Teraz wersja danych
    [self loadVersion:@"ttrv.plist"];
}
```

```objc
- (void)loadData {
    NSString *docPath =[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
        NSUserDomainMask, YES) firstObject];

    NSString *plistCatPath = [[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
        NSUserDomainMask, YES) firstObject] stringByAppendingPathComponent:@"maps.plist"];
    NSFileManager *fileManager = [NSFileManager defaultManager];


    if (![fileManager fileExistsAtPath:plistCatPath]) {
        //rozpakuj z ZIP'a
        NSString *zipPath = [[NSBundle mainBundle] pathForResource:@"ttrules" ofType:@"zip"];
        if ([fileManager fileExistsAtPath:zipPath]) {
            [SSZipArchive unzipFileAtPath:zipPath toDestination:docPath];
        }

    }

    dict = [[NSDictionary alloc] initWithContentsOfFile:plistCatPath];
    plistCatPath = [[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
        NSUserDomainMask, YES) firstObject] stringByAppendingPathComponent:@"Icons.plist"];
    icons=[[NSDictionary alloc] initWithContentsOfFile:plistCatPath];

    maps=[[NSMutableDictionary alloc] init];
    self.navigationItem.title=@"Select map";
    self.mapArray = dict[@"Maps"];

    for (int i=0; i<_mapArray.count; i++) {
        NSString* map=self.mapArray[i][@"Name"];
        NSDictionary* mapVersions=dict[map];
        maps[map]=[NSNumber numberWithLong:mapVersions.count];
    }

    //Teraz wersja danych
    [self loadVersion:@"ttrv.plist"];
}
```

```objc
- (void)viewDidLoad
{
    [super viewDidLoad];

    [self loadData];
}
```

TTRules 2020

TTRules 2020

- Dodajemy metodę loadVersion

TTRules 2020

- Dodajemy metodę loadVersion

```
- (void)loadVersion:(NSString*) fileName {
    NSString *versionPath = [[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
        NSUserDomainMask, YES) firstObject] stringByAppendingPathComponent:fileName];
    if ([[NSFileManager defaultManager] fileExistsAtPath:versionPath]) {
        version = [[NSMutableArray alloc] initWithContentsOfFile:versionPath];
    }
    else {
        version = [[NSMutableArray alloc] init];
        [version addObject:@"0"];
        [version addObject:@"08.09.2014"];
    }
}
```

TTRules 2020

- Dodajemy metodę loadVersion

```
- (void)loadVersion:(NSString*) fileName {
    NSString *versionPath = [[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
        NSUserDomainMask, YES) firstObject] stringByAppendingPathComponent:fileName];
    if ([[NSFileManager defaultManager] fileExistsAtPath:versionPath]) {
        version = [[NSMutableArray alloc] initWithContentsOfFile:versionPath];
    }
    else {
        version = [[NSMutableArray alloc] init];
        [version addObject:@"0"];
        [version addObject:@"08.09.2014"];
    }
}
```

Gdzie version jest polem typu NSMutableArray

TTRules 2020

- Dodajemy metodę loadVersion

```
- (void)loadVersion:(NSString*) fileName {
    NSString *versionPath = [[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
        NSUserDomainMask, YES) firstObject] stringByAppendingPathComponent:fileName];
    if ([[NSFileManager defaultManager] fileExistsAtPath:versionPath]) {
        version = [[NSMutableArray alloc] initWithContentsOfFile:versionPath];
    }
    else {
        version = [[NSMutableArray alloc] init];
        [version addObject:@"0"];
        [version addObject:@"08.09.2014"];
    }
}
```

Gdzie version jest polem typu NSMutableArray

```
@interface MapTableViewController : UITableViewController
{
    NSString* mapKey;
    VersionTableViewController *nextView;
    NSDictionary* dict;
    NSDictionary* icons;
    NSMutableDictionary* maps;
    NSMutableArray* version;
}
```

TTRules 2020

TTRules 2020

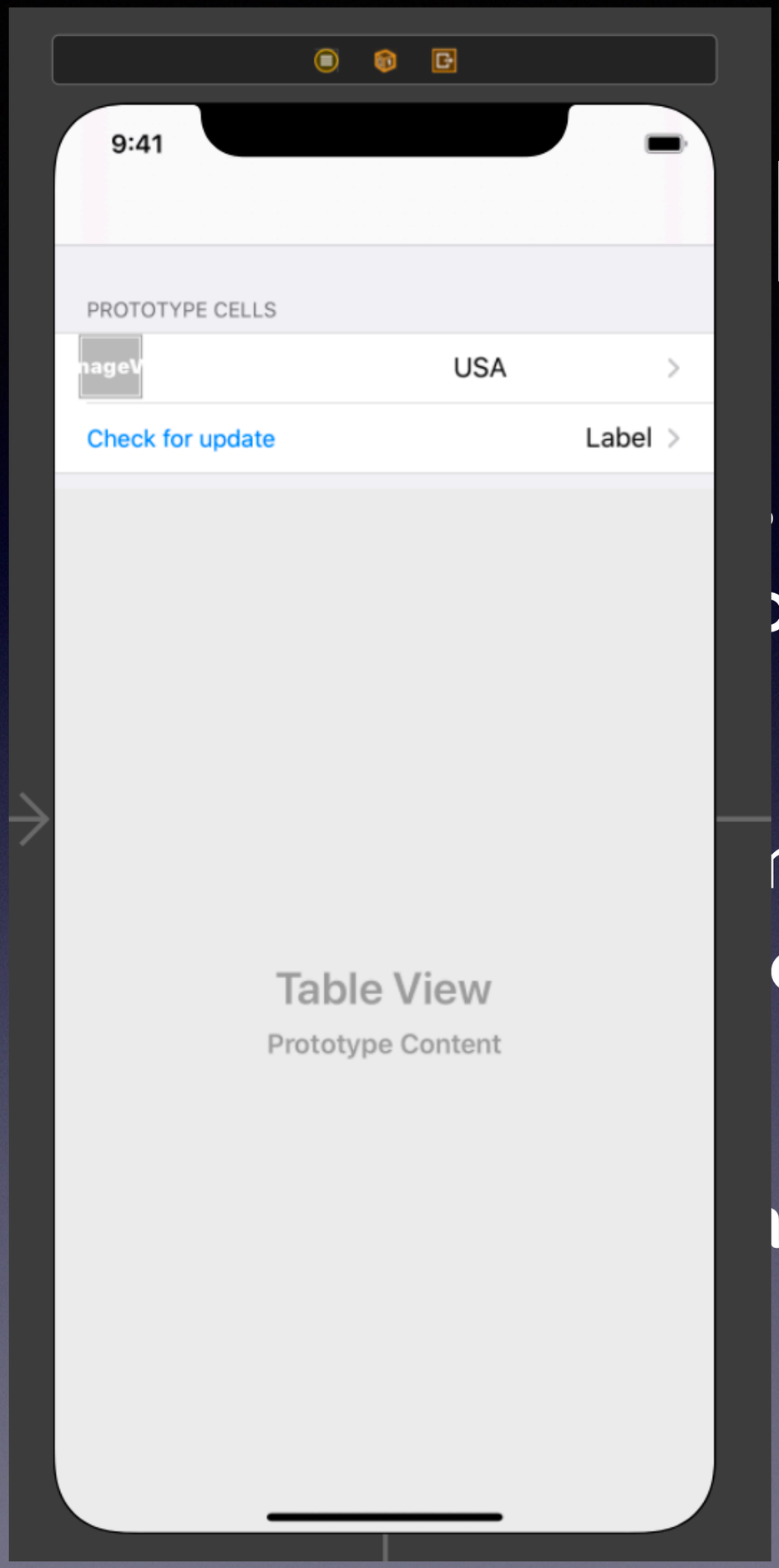
- Możliwość sprawdzania uaktualnienia dodamy jako dodatkową komórkę na końcu tabeli

TTRules 2020

- Możliwość sprawdzania uaktualnienia dodamy jako dodatkową komórkę na końcu tabeli
- Modyfikujemy Storyboard dodając drugą komórkę prototypową o identyfikatorze updateCell

TTRules 2020

- Możliwość sprawdzania uaktualnienia dodamy jako dodatkową komórkę na końcu tabeli
- Modyfikujemy Storyboard dodając drugą komórkę prototypową o identyfikatorze updateCell
- Zawiera ona przycisk (Button) i etykietkę (Label)

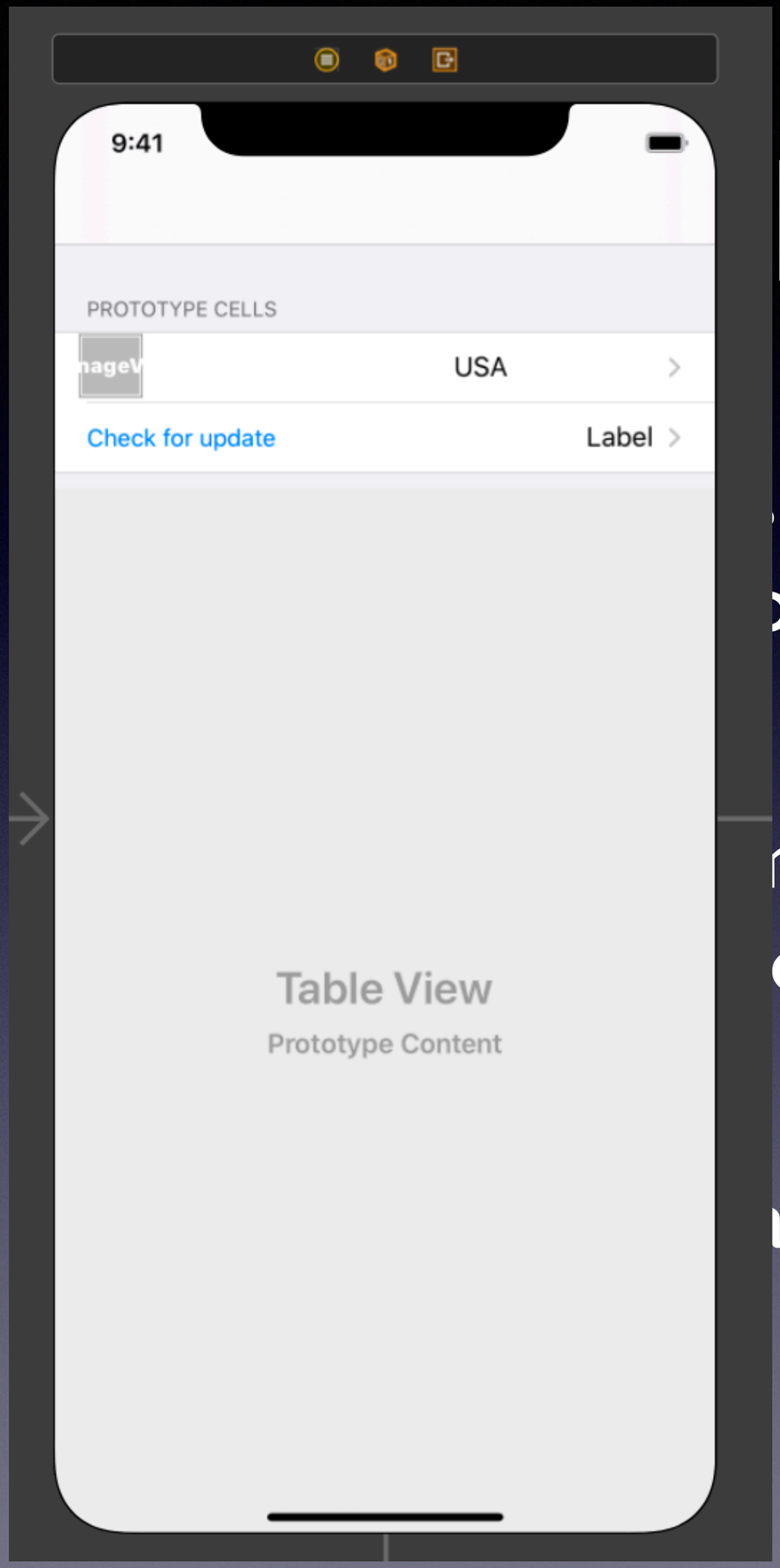


Rules 2020

sprawdzania uaktualnienia
o dodatkową komórkę na końcu

ny Storyboard dodając drugą
ototypową o identyfikatorze

a przycisk (Button) i etykietkę

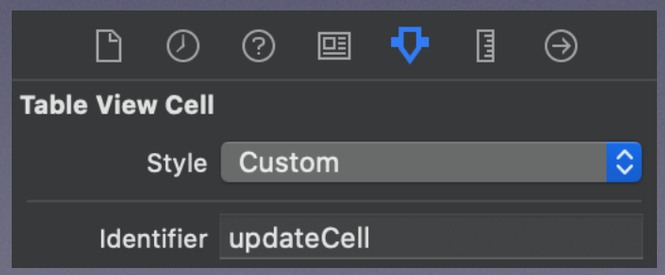


Rules 2020

sprawdzania uaktualnienia
o dodatkową komórkę na końcu

ny Storyboard dodając drugą
ototypową o identyfikatorze

a przycisk (Button) i etykietkę



TTRules 2020

TTRules 2020

- Do sterowania komórką dodajemy kolejną klasę UpdateTableViewCell

TTRules 2020

- Do sterowania komórką dodajemy kolejną klasę UpdateTableViewCell
- Klasa ma jedno gniazdko do etykietki i jedną akcję do przycisku

TTRules 2020

- Do sterowania komórką dodajemy kolejną klasę UpdateTableViewCell
- Klasa ma jedno gniazdko do etykietki i jedną akcję do przycisku
- A także własność wskazującą na nadrzędny kontroler widoku

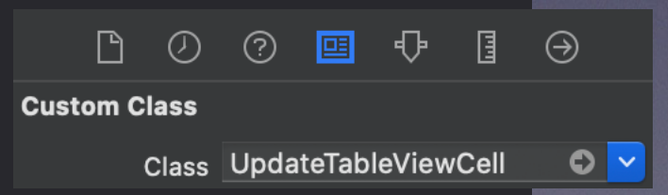
```objc
//
//  UpdateTableViewCell.h
//  TTRules
//
//  Created by Bartłomiej Prędki on 30/04/2020.
//  Copyright © 2020 Bartłomiej Prędki. All rights reserved.
//

#import <UIKit/UIKit.h>
#import "MapTableViewController.h"

NS_ASSUME_NONNULL_BEGIN

@interface UpdateTableViewCell : UITableViewCell
@property (weak, nonatomic) IBOutlet UILabel *versionLabel;
@property (weak, nonatomic) MapTableViewController *controller;

- (IBAction)CheckForUpdate:(id)sender;

@end

NS_ASSUME_NONNULL_END
```

TTRules 2020

- Do sterowania komórką dodajemy kolejną klasę UpdateTableViewCell
- Klasa ma jedno gniazdko do etykietki i jedną akcję do przycisku
- A także własność wskazującą na nadrzędny kontroler widoku

```m
//
//  UpdateTableViewCell.m
//  TTRules
//
//  Created by Bartłomiej Prędki on 30/04/2020.
//  Copyright © 2020 Bartłomiej Prędki. All rights reserved.
//

#import "UpdateTableViewCell.h"

@implementation UpdateTableViewCell

- (void)awakeFromNib {
    [super awakeFromNib];
    // Initialization code
}

- (void)setSelected:(BOOL)selected animated:(BOOL)animated {
    [super setSelected:selected animated:animated];

    // Configure the view for the selected state
}

- (IBAction)CheckForUpdate:(id)sender {
    [self.controller checkForUpdate];
}

@end
```

```objc
//
//  UpdateTableViewCell.m
//  TTRules
//
//  Created by Bartłomiej Prędki on 30/04/2020.
//  Copyright © 2020 Bartłomiej Prędki. All rights reserved.
//

#import "UpdateTableViewCell.h"

@implementation UpdateTableViewCell

- (void)awakeFromNib {
    [super awakeFromNib];
    // Initialization code
}

- (void)setSelected:(BOOL)selected animated:(BOOL)animated {
    [super setSelected:selected animated:animated];

    // Configure the view for the selected state
}

- (IBAction)CheckForUpdate:(id)sender {
    [self.controller checkForUpdate];
}

@end
```



TTRules 2020

TTRules 2020

- Zmieniamy metody obsługi tabel:

```objc
- (NSString *)tableView:(UITableView *)tableView
titleForHeaderInSection:(NSInteger)section
{
    switch (section)
    {
        case 0:
            return @"Official maps";
        case 1:
            return @"Settings";
        default:
            return @"";
    }
}

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    // Return the number of sections.
    return 2;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    // Return the number of rows in the section.
    switch (section) {
        case 0:
            return [self.mapArray count];
        case 1:
            return 1;
        default:
            return 0;
    }
}
```

TTRules 2020

TTRules 2020

- I tworzenie komórek

```objc
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
    *)indexPath
{
    if (indexPath.section == 0) {
        MapTableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"mapCell"
            forIndexPath:indexPath];
        NSString* map=self.mapArray[indexPath.row][@"Name"];

        cell.lblMapName.text=map;
        cell.key=map;
        NSString* imageName=[icons[cell.key] stringByAppendingString: @".png"];
        NSString *imagePath = [[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
            NSUserDomainMask, YES) firstObject] stringByAppendingPathComponent:imageName];
        UIImage* img=[UIImage imageWithContentsOfFile:imagePath];
        cell.imgIcon.image=img;
        return cell;
    }
    else {
        UpdateTableViewCell *cell = [tableView
            dequeueReusableCellWithIdentifier:@"updateCell" forIndexPath:indexPath];
        cell.versionLabel.text = [NSString stringWithFormat:@"%d",[version[0] intValue]];
        cell.controller = self;

        return cell;
    }

}
```

TTRules 2020

TTRules 2020

- Dodamy metody do ściągania plików:

TTRules 2020

- Dodamy metody do ściągania plików:
- Synchronicznego

TTRules 2020

```
- (BOOL)downloadFile:(NSString *)urlToDownload :(NSString *)destinationName {
    NSURL  *url = [NSURL URLWithString:urlToDownload];
    NSData *urlData = [NSData dataWithContentsOfURL:url];
    if ( urlData )
    {
        NSString *filePath = [[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
            NSUserDomainMask, YES) firstObject]
            stringByAppendingPathComponent:destinationName];

        [urlData writeToFile:filePath atomically:YES];
        return YES;
    }
    return NO;
}
```

TTRules 2020

- Dodamy metody do ściągania plików:
- Synchronicznego

TTRules 2020

- Dodamy metody do ściągania plików:
  - Synchronicznego
  - Asynchronicznego

TTRules 2020

```
- (void)downloadFileAsync:(NSString *)urlToDownload :(NSString *)destinationName {
    //download the file in a seperate thread.
    dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
        NSURL  *url = [NSURL URLWithString:urlToDownload];
        NSData *urlData = [NSData dataWithContentsOfURL:url];
        if ( urlData )
        {
            NSString *filePath = [[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
                NSUserDomainMask, YES) firstObject]
                stringByAppendingPathComponent:destinationName];

            //saving is done on main thread
            dispatch_async(dispatch_get_main_queue(), ^{
                [urlData writeToFile:filePath atomically:YES];
                NSString *docPath =[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
                    NSUserDomainMask, YES) firstObject];
                [SSZipArchive unzipFileAtPath:filePath toDestination:docPath];
                [self showAlert:@"Update" :@"New update downloaded" :@"OK" :@""];
                [self loadData];
                [self.tableView reloadData];
            });
        }

    });
}
```

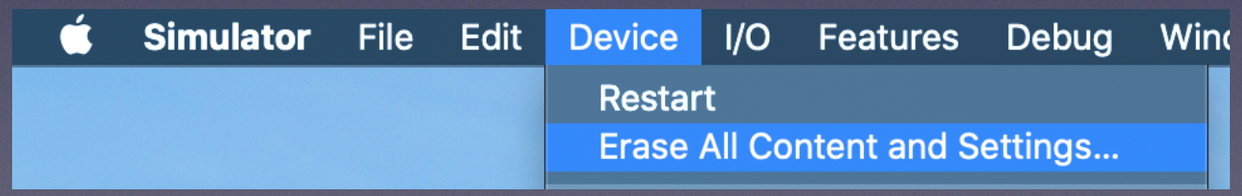
TTRules 2020

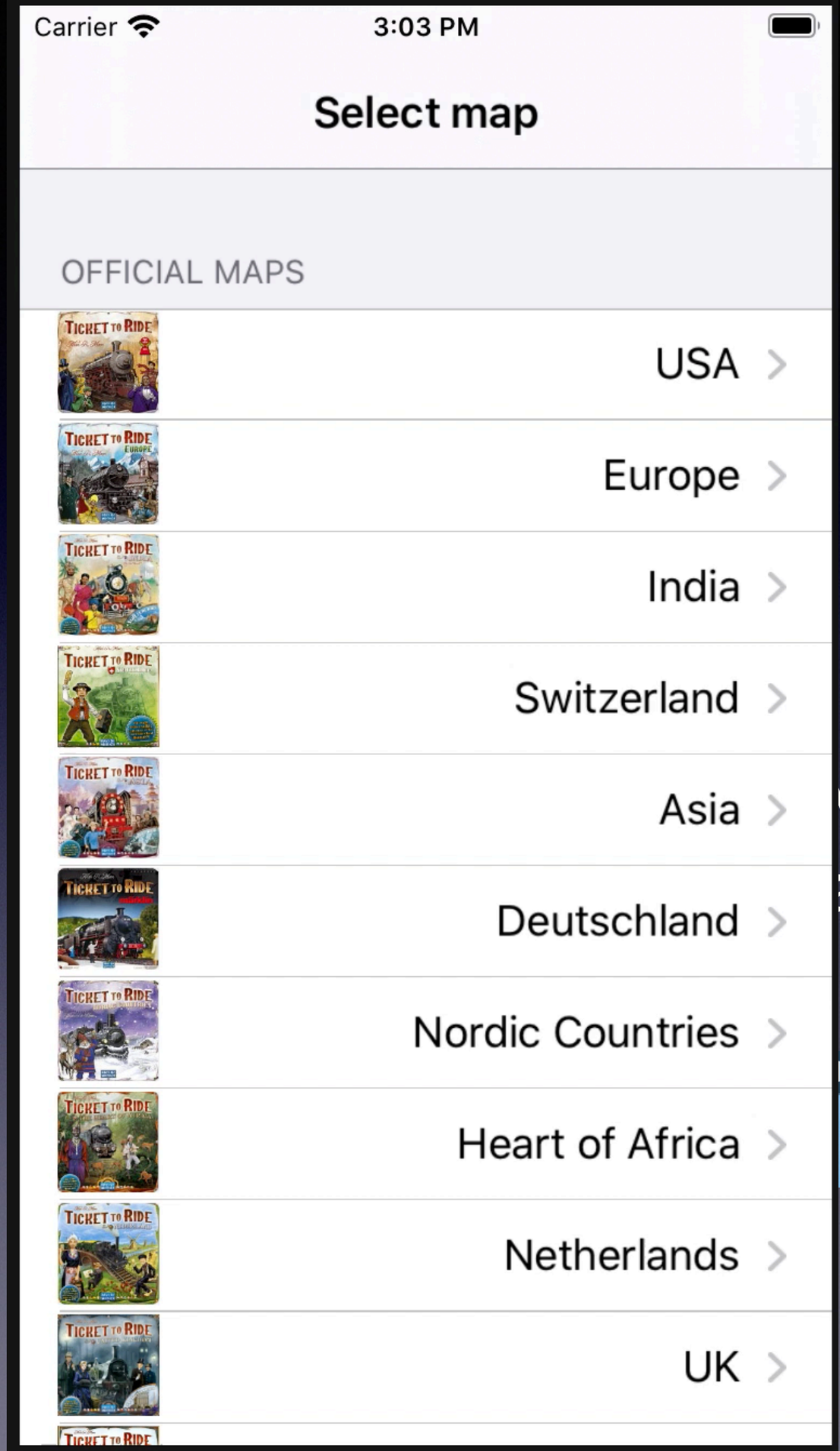
TTRules 2020

- Przy testowaniu warto korzystać z funkcji symulatora kasującej zawartość

TTRules 2020

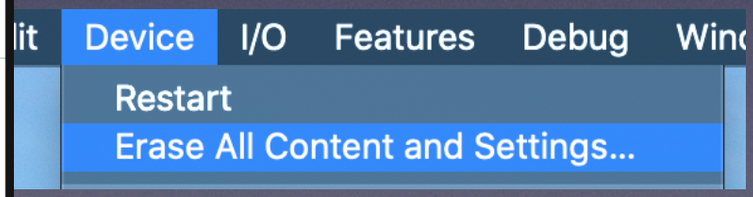
- Przy testowaniu warto korzystać z funkcji symulatora kasującej zawartość





les 2020

warto korzystać z funkcji
ącej zawartość



Carrier 3:03 PM

Select map

OFFICIAL MAPS

- USA
- Europe
- India
- Switzerland
- Asia
- Deutschland
- Nordic Countries
- Heart of Africa
- Netherlands
- UK

les

wartc

ącej z

lit Device

Restart

Erase A

Carrier 7:55 PM

Files Shortcuts Contacts Watch

TTRules

TTRules 2020

TTRules 2020

- I jeszcze metoda checkForUpdate

TTRules 2020

- I jeszcze metoda checkForUpdate

```
-(void) checkForUpdate {
    if ([self downloadFile:@"http://fcds.cs.put.poznan.pl/MyWeb/Media/ttrv.txt"
        :@"ttrv.txt"]) {
        int previous = [version[0] intValue];
        [self loadVersion:@"ttrv.txt"];
        if (previous < [version[0] intValue]) {
            [self downloadFileAsync:@"http://fcds.cs.put.poznan.pl/MyWeb/Media/ttrules.zip"
                :@"ttrules.zip"];
        } else {
            [self showAlert:@"Update" :@"No new update" :@"OK" :@""];
        }
    } else {
        NSLog(@"Problem downloading file");
    }
}
```

SWIFT

SWIFT

Swift (*pol. jerzyk*)

Swift (*pol. jerzyk*)

Swift (*pol. jerzyk*)

- nowy język programowania zaprezentowany latem 2014 r. (prace od 2010 r.)

Swift (*pol. jerzyk*)

- nowy język programowania zaprezentowany latem 2014 r. (prace od 2010 r.)
- przeznaczony do programowania zarówno pod iOS jak i macOS

Swift (*pol. jerzyk*)

- nowy język programowania zaprezentowany latem 2014 r. (prace od 2010 r.)
- przeznaczony do programowania zarówno pod iOS jak i macOS
- bazuje na logice Objective-C bez kompatybilności z C

Swift (*pol. jerzyk*)

- nowy język programowania zaprezentowany latem 2014 r. (prace od 2010 r.)
- przeznaczony do programowania zarówno pod iOS jak i macOS
- bazuje na logice Objective-C bez kompatybilności z C
- należy do języków wykorzystujących LLVM

Swift (*pol. jerzyk*)

- nowy język programowania zaprezentowany latem 2014 r. (prace od 2010 r.)
- przeznaczony do programowania zarówno pod iOS jak i macOS
- bazuje na logice Objective-C bez kompatybilności z C
- należy do języków wykorzystujących LLVM
- obecnie wersja 5.1

RedMonk Q120 Programming Language Rankings

RedMonk

Popularity Rank on Stack Overflow (by # of Tags)

Popularity Rank on GitHub (by # of Projects)

0 25 50 75 100

JavaScript
Java
Python
C#
PHP
C++
CSS
R
Objective-C
C
Swift
Ruby
TypeScript
Scala
Visual Basic
Matlab
PowerShell
Shell
Perl
Go
Haskell
Kotlin
GCC Machine Description
Assembly
Dart
Groovy
PLSQL
Lua
Clojure
Arduino
Rust
F#
Cuda
ColdFusion
ASP
FORTRAN
CoffeeScript
ActionScript
QML
Erlang
Elixir
Scheme
Tcl
Julia
GLSL
OCaml
Processing
Common Lisp
VHDL
Verilog
Racket
XQuery
Smarty
Emacs Lisp
Mathematica
Puppet
PLpgSQL
FreeMarker
D
Liquid
Pascal
Web Ontology Language
Coq
Standard ML
Elm
Apex
Smalltalk
Solidity
Haxe
Raku
SaltStack
Objective-C++
PostScript
Vala
Gherkin
BitBake
WebAssembly
Modelica
Crystal
PureScript
Nix
Nim
MAXScript
API Blueprint
Chapel
Hack
GDScript
Game Maker Language
Logos
ShaderLab
HCL
Vim script
Jsonnet
SourcePawn
SQLPL
GAP
SQF
Eagle
Roff
DM

RedMonk Q120 Programming Language Rankings

RedMonk

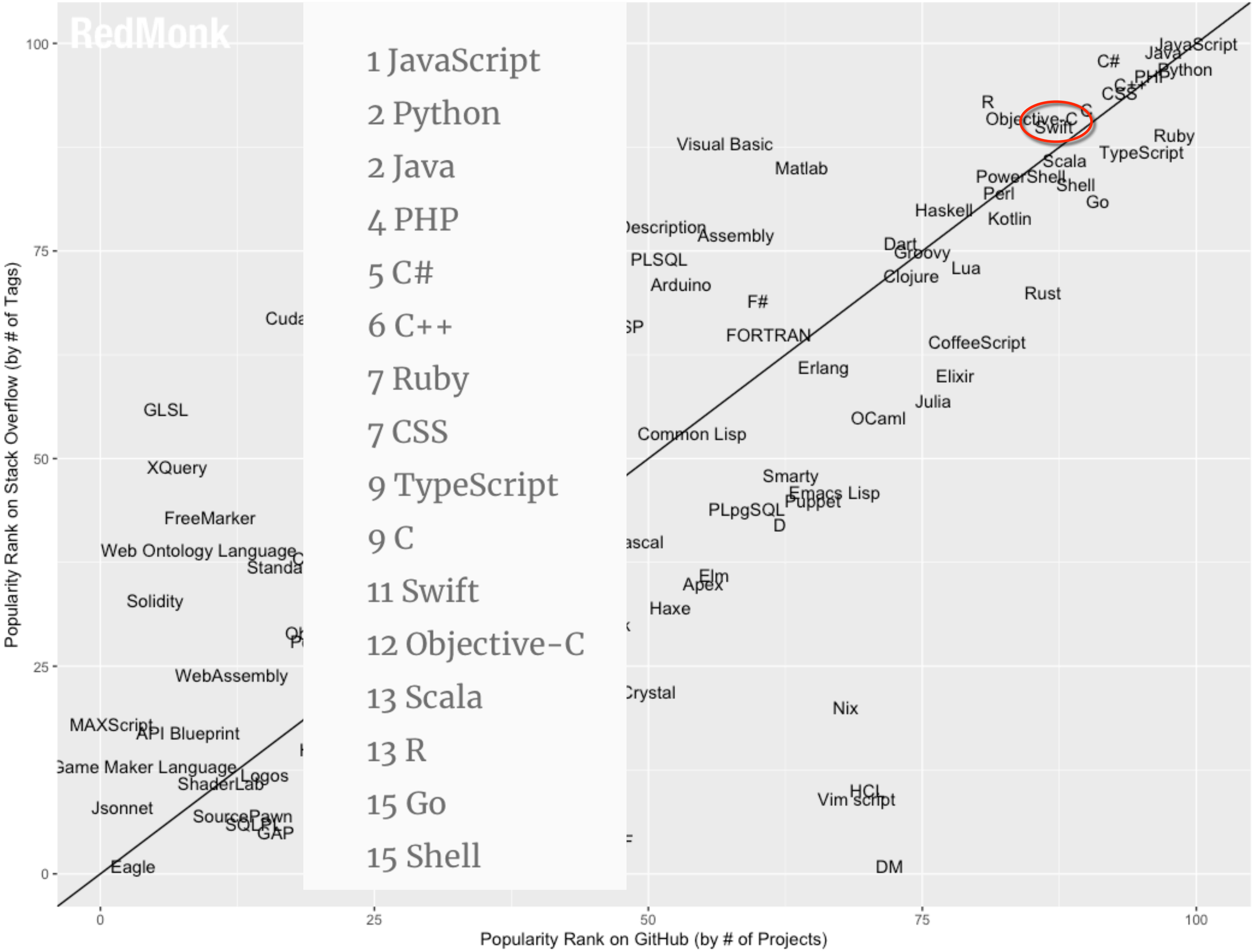
Popularity Rank on Stack Overflow (by # of Tags)

Popularity Rank on GitHub (by # of Projects)

0
25
50
75
100

JavaScript
Java
Python
C#
PHP
C++
CSS
R
Objective-C
C
Swift
Ruby
TypeScript
Scala
PowerShell
Shell
Perl
Go
Haskell
Kotlin
Visual Basic
Matlab
GCC Machine Description
Assembly
Dart
Groovy
Lua
PLSQL
Clojure
Arduino
Rust
F#
Cuda
ColdFusion
ASP
FORTRAN
CoffeeScript
ActionScript
QML
Erlang
Elixir
Scheme
Tcl
Julia
GLSL
OCaml
Processing
Common Lisp
VHDL
Verilog
Racket
XQuery
Smarty
Mathematica
Emacs Lisp
Puppet
PLpgSQL
FreeMarker
D
Liquid
Pascal
Web Ontology Language
Coq
Standard ML
Elm
Apex
Smalltalk
Solidity
Haxe
Raku
SaltStack
Objective-C++
PostScript
Vala
Gherkin
BitBake
WebAssembly
Modelica
Crystal
PureScript
Nix
Nim
MAXScript
API Blueprint
Chapel
Hack
GDScript
Game Maker Language
Logos
ShaderLab
HCL
Vim script
Jsonnet
SourcePawn
SQLPL
GAP
SQF
Eagle
Roff
DM

RedMonk Q120 Programming Language Rankings



1 JavaScript
2 Python
2 Java
4 PHP
5 C#
6 C++
7 Ruby
7 CSS
9 TypeScript
9 C
11 Swift
12 Objective-C
13 Scala
13 R
15 Go
15 Shell

Podobieństwa do Objective-C

Podobieństwa do Objective-C

- typy podstawowe

Podobieństwa do Objective-C

- typy podstawowe
- operatory

Podobieństwa do Objective-C

- typy podstawowe
- operatory
- użycie nawiasów klamrowych { }

Podobieństwa do Objective-C

- typy podstawowe
- operatory
- użycie nawiasów klamrowych `{}`
- użycie nawiasów kwadratowych `[]` (tablice)

Podobieństwa do Objective-C

- typy podstawowe
- operatory
- użycie nawiasów klamrowych `{ }`
- użycie nawiasów kwadratowych `[ ]` (tablice)
- operator przypisania =, porównania ==

Podobieństwa do Objective-C

- typy podstawowe
- operatory
- użycie nawiasów klamrowych `{ }`
- użycie nawiasów kwadratowych `[ ]` (tablice)
- operator przypisania =, porównania ==
- podobne instrukcje (pętle, warunkowe)

Podobieństwa do Objective-C

- typy podstawowe
- operatory
- użycie nawiasów klamrowych `{ }`
- użycie nawiasów kwadratowych `[ ]` (tablice)
- operator przypisania =, porównania ==
- podobne instrukcje (pętle, warunkowe)
- dziedziczenie metod klas i instancji

Podobieństwa do Objective-C

- typy podstawowe
- operatory
- użycie nawiasów klamrowych `{}`
- użycie nawiasów kwadratowych `[]` (tablice)
- operator przypisania =, porównania ==
- podobne instrukcje (pętle, warunkowe)
- dziedziczenie metod klas i instancji
- słowo kluczowe `self`

Różnice vs Objective-C

Różnice vs Objective-C

- nie trzeba używać ;

Różnice vs Objective-C

- nie trzeba używać ;
- brak plików nagłówkowych

Różnice vs Objective-C

- nie trzeba używać ;
- brak plików nagłówkowych
- silne typowanie

Różnice vs Objective-C

- nie trzeba używać ;
- brak plików nagłówkowych
- silne typowanie
- przeciążanie operatorów

Różnice vs Objective-C

- nie trzeba używać ;
- brak plików nagłówkowych
- silne typowanie
- przeciążanie operatorów
- nowy operator === porównujący obiekty

Różnice vs Objective-C

- nie trzeba używać ;
- brak plików nagłówkowych
- silne typowanie
- przeciążanie operatorów
- nowy operator === porównujący obiekty
- pełne wsparcie Unicode w string

Różnice vs Objective-C

- nie trzeba używać ;
- brak plików nagłówkowych
- silne typowanie
- przeciążanie operatorów
- nowy operator === porównujący obiekty
- pełne wsparcie Unicode w string
- inferencja typów

Różnice vs Objective-C

- nie trzeba używać ;
- brak plików nagłówkowych
- silne typowanie
- przeciążanie operatorów
- nowy operator === porównujący obiekty
- pełne wsparcie Unicode w string
- inferencja typów
- programowanie generyczne

Różnice vs Objective-C

- nie trzeba używać ;
- brak plików nagłówkowych
- silne typowanie
- przeciążanie operatorów
- nowy operator === porównujący obiekty
- pełne wsparcie Unicode w string
- inferencja typów
- programowanie generyczne
- typy funkcyjne

Różnice vs Objective-C

Różnice vs Objective-C

- domyślnie nie ma wskaźników

Różnice vs Objective-C

- domyślnie nie ma wskaźników
- przypisanie nie zwraca wartości

Różnice vs Objective-C

- domyślnie nie ma wskaźników
- przypisanie nie zwraca wartości
- brak konieczności używania `break` w instrukcji `switch` (instrukcja `fallthrough`)

Różnice vs Objective-C

- domyślnie nie ma wskaźników
- przypisanie nie zwraca wartości
- brak konieczności używania `break` w instrukcji `switch` (instrukcja `fallthrough`)
- zmienne i stałe są zawsze inicjalizowane

Różnice vs Objective-C

- domyślnie nie ma wskaźników
- przypisanie nie zwraca wartości
- brak konieczności używania `break` w instrukcji `switch` (instrukcja `fallthrough`)
- zmienne i stałe są zawsze inicjalizowane
- przepełnienie zakresu jest wykrywane, chyba, że użyjemy specjalnych operatorów, np. `&+`, `&-`

Playground

Playground

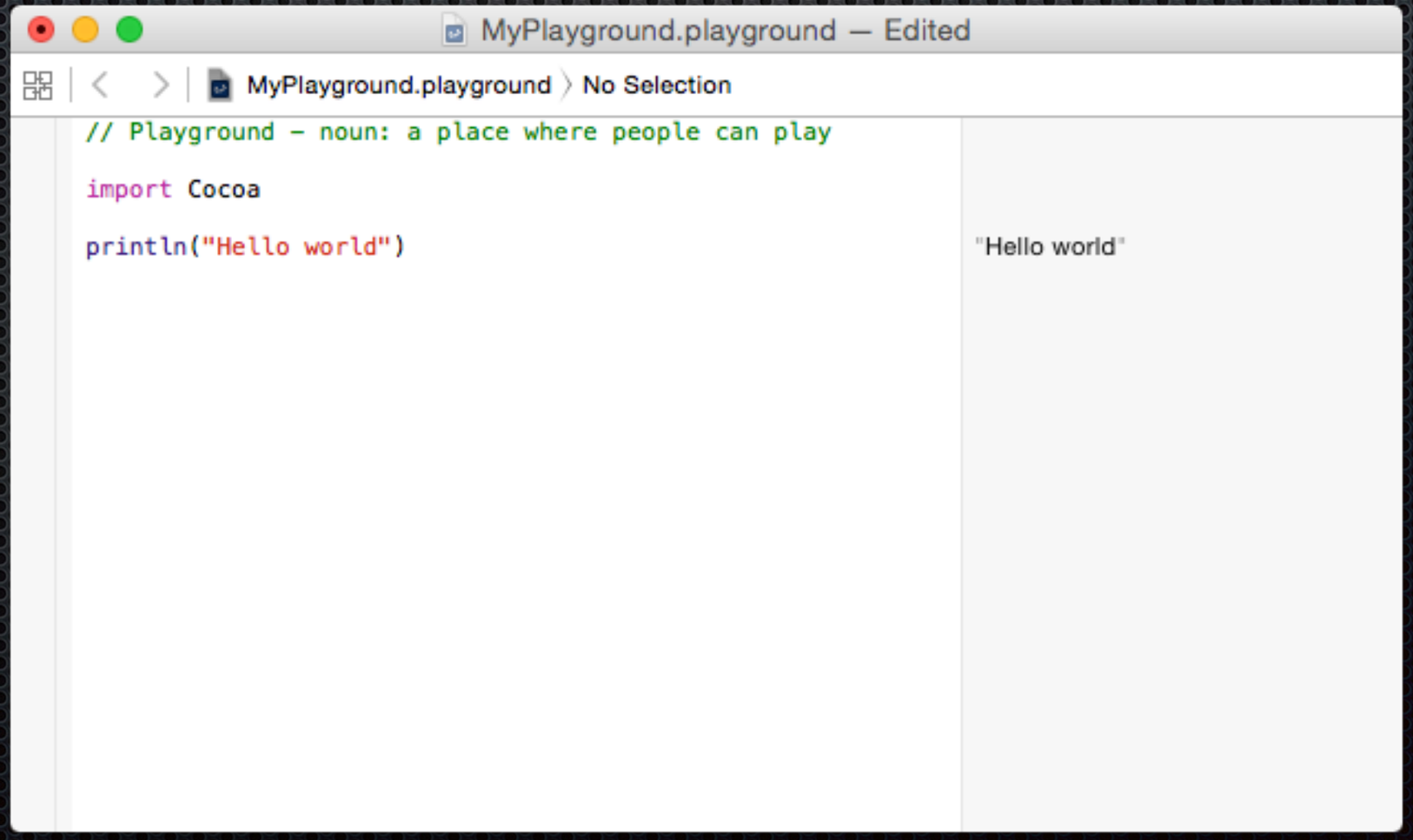
- xCode zawiera moduł Playground

Playground

- xCode zawiera moduł Playground
- służy on do sprawdzania na żywo jak działa kod Swift

Playground

- xCode zawiera moduł Playground
- służy on do sprawdzania na żywo jak działa kod Swift



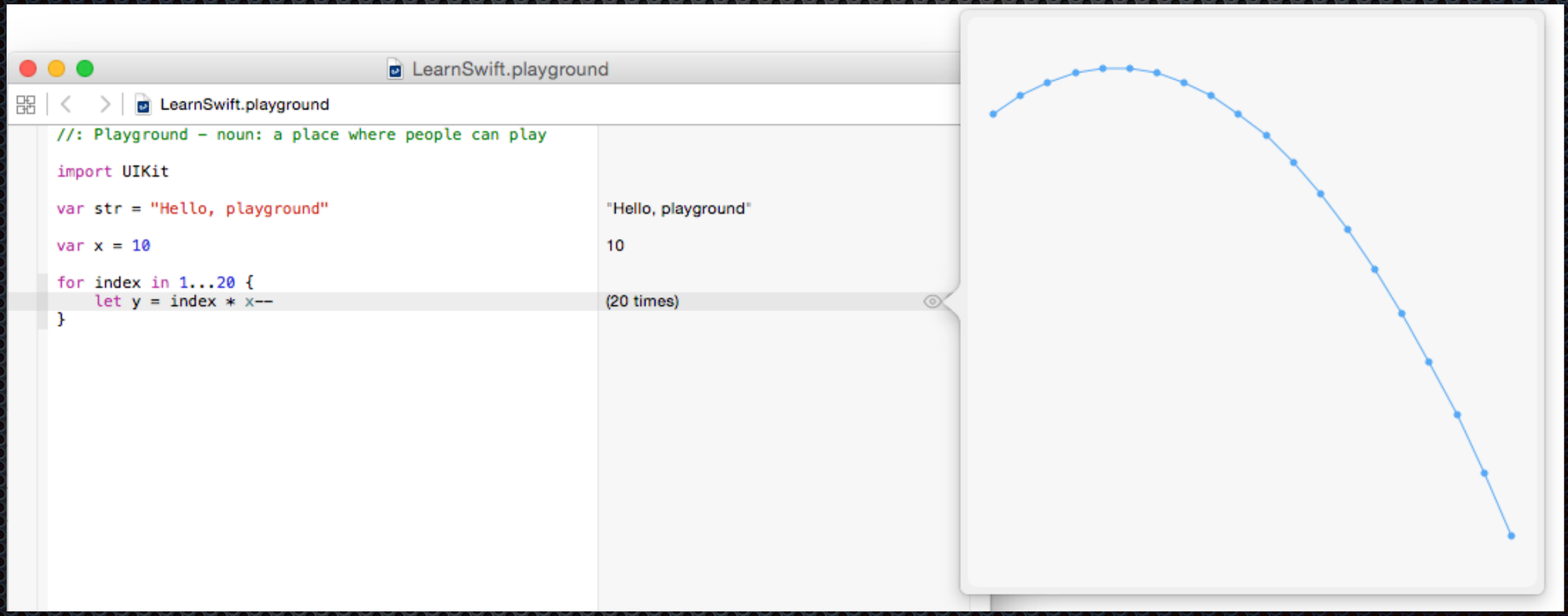
Playground

Playground

- Można na różne sposoby wizualizować działanie kodu

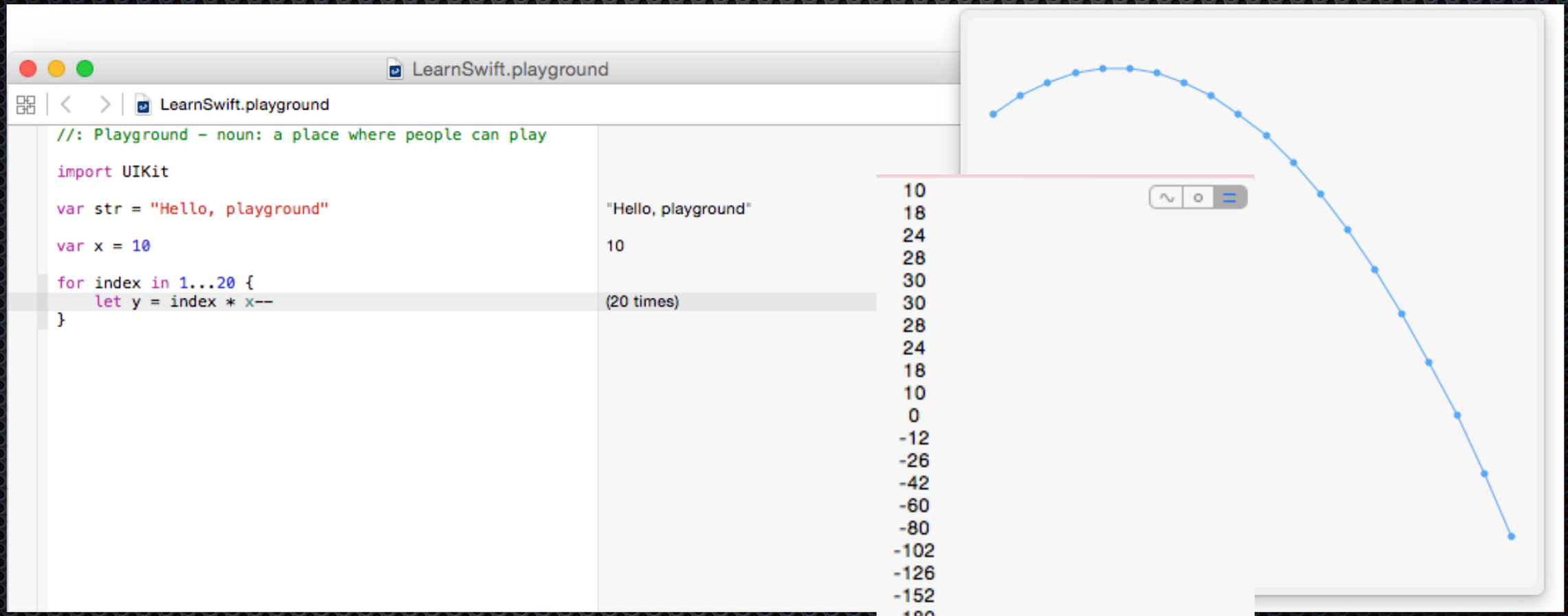
Playground

- Można na różne sposoby wizualizować działanie kodu



Playground

- Można na różne sposoby wizualizować działanie kodu



Playground

Playground

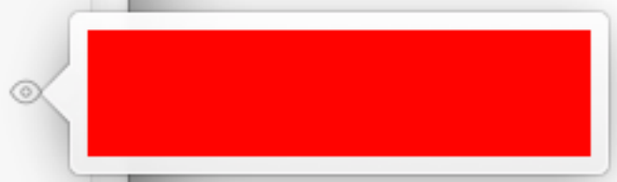
- Można korzystać z większości API, np. UIKit

Playground

- Można korzystać z większości API, np. UIKit

```
let myLabel = UILabel(frame: CGRectMake(0, 0, 200, 50))
myLabel.backgroundColor = UIColor.redColor()
myLabel.text = "Hello Swift"
myLabel.textAlignment = .Center
myLabel.font = UIFont(name: "Georgia", size: 24)
myLabel
```

UILabel
UILabel
UILabel
UILabel
UILabel
UILabel

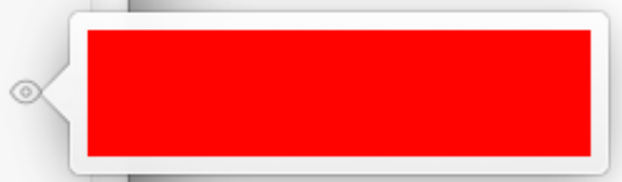


Playground

- Można korzystać z większości API, np. UIKit

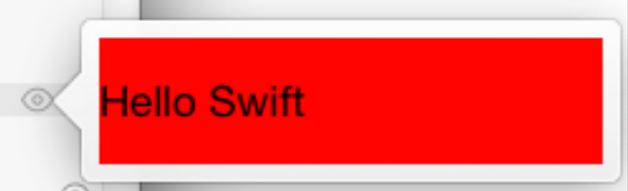
```
let myLabel = UILabel(frame: CGRectMake(0, 0, 200, 50))
myLabel.backgroundColor = UIColor.redColor()
myLabel.text = "Hello Swift"
myLabel.textAlignment = .Center
myLabel.font = UIFont(name: "Georgia", size: 24)
myLabel
```

```
UILabel
UILabel
UILabel
UILabel
UILabel
UILabel
```




```
let myLabel = UILabel(frame: CGRectMake(0, 0, 200, 50))
myLabel.backgroundColor = UIColor.redColor()
myLabel.text = "Hello Swift"
myLabel.textAlignment = .Center
myLabel.font = UIFont(name: "Georgia", size: 24)
myLabel
```

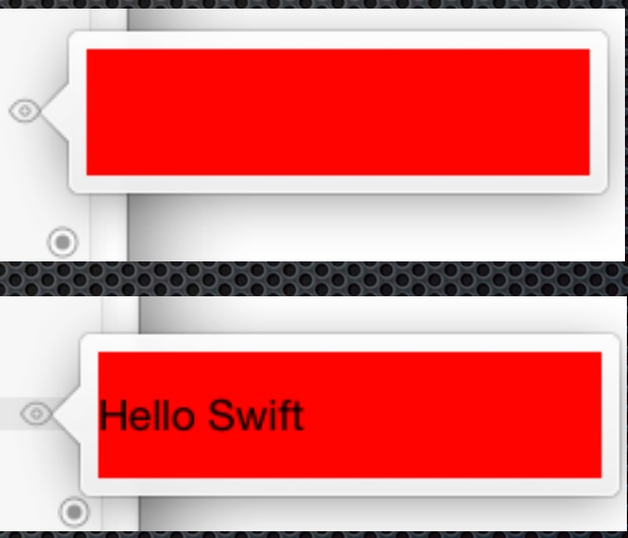

```
UILabel
UILabel
UILabel
UILabel
UILabel
UILabel
```



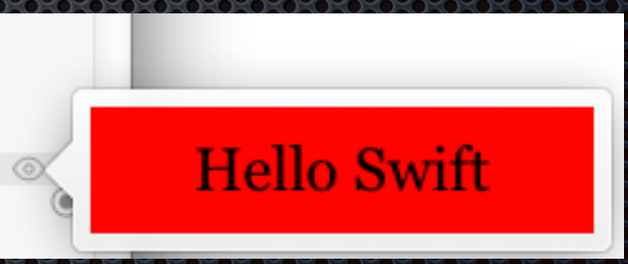
Playground

- Można korzystać z większości API, np. UIKit

```
let myLabel = UILabel(frame: CGRectMake(0, 0, 200, 50))
myLabel.backgroundColor = UIColor.redColor()
myLabel.text = "Hello Swift"
myLabel.textAlignment = .Center
myLabel.font = UIFont(name: "Georgia", size: 24)
myLabel
```




```
let myLabel = UILabel(frame: CGRectMake(0, 0, 200, 50))
myLabel.backgroundColor = UIColor.redColor()
myLabel.text = "Hello Swift"
myLabel.textAlignment = .Center
myLabel.font = UIFont(name: "Georgia", size: 24)
myLabel
```



Zmienne i stałe

Zmienne i stałe

- Słowa kluczowe let i var

Zmienne i stałe

- Słowa kluczowe let i var
- let definiuje stałą

Zmienne i stałe

- Słowa kluczowe let i var
  - let definiuje stałą
  - var definiuje zmienną

```
var myVariable = 42                    42
myVariable = 50                        50
let myConstant = 42                    42
```

Zmienne i stałe

- Słowa kluczowe let i var
  - let definiuje stałą
  - var definiuje zmienną
- podstawienie wartości określa typ zmiennej/stałej dzięki inferencji typów

```
var myVariable = 42                    42
myVariable = 50                        50
let myConstant = 42                    42
```

```
let implicitInteger = 70               70
let implicitDouble = 70.0              70.0
let explicitDouble: Double = 70        70.0
```

Zmienne i stałe

Zmienne i stałe

- Nazwy zmiennych mogą zawierać dowolne znaki Unicode oprócz:

Zmienne i stałe

- Nazwy zmiennych mogą zawierać dowolne znaki Unicode oprócz:
  - spacji

Zmienne i stałe

- Nazwy zmiennych mogą zawierać dowolne znaki Unicode oprócz:
  - spacji
  - symboli matematycznych

Zmienne i stałe

- Nazwy zmiennych mogą zawierać dowolne znaki Unicode oprócz:
  - spacji
  - symboli matematycznych
  - linii i prostokątów

Zmienne i stałe

- Nazwy zmiennych mogą zawierać dowolne znaki Unicode oprócz:
  - spacji
  - symboli matematycznych
  - linii i prostokątów
- Nie mogą się zaczynać od cyfry

Zmienne i stałe

- Nazwy zmiennych mogą zawierać dowolne znaki Unicode oprócz:
  - spacji
  - symboli matematycznych
  - linii i prostokątów
- Nie mogą się zaczynać od cyfry

```
let π = 3.14159
let 你好 = "你好世界"
let 🐶🐮 = "dogcow"
```

Typy podstawowe

Typy podstawowe

- `Int8`

Typy podstawowe

- `Int8`
- `UInt8`

Typy podstawowe

- `Int8`
- `UInt8`
- `Int` (Int32, Int64)

Typy podstawowe

- `Int8`
- `UInt8`
- `Int` (Int32, Int64)
- `UInt` (UInt32, UInt64)

Typy podstawowe

- `Int8`
- `UInt8`
- `Int` (Int32, Int64)
- `UInt` (UInt32, UInt64)
- `Double` (64 bity - 15 cyfr)

Typy podstawowe

- `Int8`
- `UInt8`
- `Int` (Int32, Int64)
- `UInt` (UInt32, UInt64)
- `Double` (64 bity - 15 cyfr)
- `Float` (32 bity - 6 cyfr)

Typy podstawowe

- `Int8`
- `UInt8`
- `Int` (Int32, Int64)
- `UInt` (UInt32, UInt64)
- `Double` (64 bity - 15 cyfr)
- `Float` (32 bity - 6 cyfr)
- `String`

Typy podstawowe

- `Int8`
- `UInt8`
- `Int` (Int32, Int64)
- `UInt` (UInt32, UInt64)
- `Double` (64 bity - 15 cyfr)
- `Float` (32 bity - 6 cyfr)
- `String`
- `Bool` (true lub false)

Typy podstawowe

Typy podstawowe

- Dla których ma to sens mają własności max i min

```
let tooBig: Int8 = Int8.max
```

Typy podstawowe

- Dla których ma to sens mają własności max i min

```
let tooBig: Int8 = Int8.max
```

- Można stosować aliasy nazw typów

```
typealias AudioSample = UInt16
```

Operatory

Operatory

- Podstawowe operatory: +,-,*,/,% (modulo)

Operatory

- Podstawowe operatory: +,-,*,/,% (modulo)
- Operatory złożone: +=, -=, *=, /=, %=, &&, ||

Operatory

- Podstawowe operatory: +,-,*,/,% (modulo)
- Operatory złożone: +=, -=, *=, /=, %=, &&, ||
- Od Swift 3 nie ma operatorów inkrementacyjnych: ++, —

Operatory

- Podstawowe operatory: +,-,*,/,% (modulo)
- Operatory złożone: +=, -=, *=, /=, %=, &&, ||
- Od Swift 3 nie ma operatorów inkrementacyjnych: ++, —
- Operatory porównania: ==, >, <, >=, <=, !=

Operatory

- Podstawowe operatory: +,-,*,/,% (modulo)
- Operatory złożone: +=, -=, *=, /=, %=, &&, ||
- Od Swift 3 nie ma operatorów inkrementacyjnych: ++, —
- Operatory porównania: ==, >, <, >=, <=, !=
- Operatory zakresu: x...y, x..<y

Operatory

- Podstawowe operatory: +,-,*,/,% (modulo)
- Operatory złożone: +=, -=, *=, /=, %=, &&, ||
- Od Swift 3 nie ma operatorów inkrementacyjnych: ++, —
- Operatory porównania: ==, >, <, >=, <=, !=
- Operatory zakresu: x...y, x..<y
- Operatory bitowe: &, |, ^, <<, >>, &=, |=, ^=, <<=, >>=

Operatory typów

Operatory typów

- Operator `is` sprawdza, czy obiekt jest danego typu i ma wartość boolowską

Operatory typów

- Operator `is` sprawdza, czy obiekt jest danego typu i ma wartość boolowską
- Operator `as?` rzutuje obiekt na typ opcjonalny, a gdy jest to niemożliwe podstawia `nil`

Operatory typów

- Operator `is` sprawdza, czy obiekt jest danego typu i ma wartość boolowską
- Operator `as?` rzutuje obiekt na typ opcjonalny, a gdy jest to niemożliwe podstawia `nil`
- Operator `as!` wymusza rzutowanie na typ, a jeżeli jest niemożliwe to występuje błąd wykonania

Operatory typów

- Operator `is` sprawdza, czy obiekt jest danego typu i ma wartość boolowską
- Operator `as?` rzutuje obiekt na typ opcjonalny, a gdy jest to niemożliwe podstawia `nil`
- Operator `as!` wymusza rzutowanie na typ, a jeżeli jest niemożliwe to występuje błąd wykonania
- Operator `as` można stosować przy rzutowaniu w górę

Zmienne i stałe

Zmienne i stałe

- wartości nigdy nie są domyślnie konwertowane do innych typów - trzeba stworzyć nową wartość

```
let label = "The width is "
let width = 94
let widthLabel = label + String(width)
```

```
"The width is "
94
"The width is 94"
```

Zmienne i stałe

- wartości nigdy nie są domyślnie konwertowane do innych typów - trzeba stworzyć nową wartość

```
let label = "The width is "
let width = 94
let widthLabel = label + String(width)
```

```
"The width is "
94
"The width is 94"
```

- wartości w stringach można osadzać łatwiej

```
let apples = 3
let oranges = 5
let appleSummary = "I have \(apples) apples."
let fruitSummary = "I have \(apples + oranges) pieces
    of fruit."
```

```
3
5
"I have 3 apples."
"I have 8 pieces of fruit."
```

Zmienne i stałe

- wartości nigdy nie są domyślnie konwertowane do innych typów - trzeba stworzyć nową wartość

```
let label = "The width is "
let width = 94
let widthLabel = label + String(width)
```

```
"The width is "
94
"The width is 94"
```

- wartości w stringach można osadzać łatwiej

```
let apples = 3
let oranges = 5
let appleSummary = "I have \(apples) apples."
let fruitSummary = "I have \(apples + oranges) pieces
    of fruit."
```

```
3
5
"I have 3 apples."
"I have 8 pieces of fruit."
```

- typy opcjonalne (nullowe) - dodajemy `?` wtedy zmienna może mieć wartość `nil`

```
var optionalString: String? = "Hello"

optionalString == nil
```

Typy opcjonalne

Typy opcjonalne

- W pracy ze Swiftem bardzo często operujemy na typach opcjonalnych

Typy opcjonalne

- W pracy ze Swiftem bardzo często operujemy na typach opcjonalnych
- Pomaga w tym instrukcja guard

Typy opcjonalne

- W pracy ze Swiftem bardzo często operujemy na typach opcjonalnych
- Pomaga w tym instrukcja guard

```
func submit() {
    guard let name = nameField.text else {
        show("No name to submit")
        return
    }

    guard let address = addressField.text else {
        show("No address to submit")
        return
    }

    guard let phone = phoneField.text else {
        show("No phone to submit")
        return
    }

    sendToServer(name, address: address, phone: phone)
}
```

Typy opcjonalne

```
func tappedSubmitButton() {
    guard let name = nameField.text where isValid(name) else {
        show("name failed validation")
        return
    }

    submit(name)
}

func isValid(name: String) -> Bool {
    // check the name is between 4 and 16 characters
    if !(4...16 ~= name.characters.count) {
        return false
    }

    // check that name doesn't contain whitespace or newline characters
    let range = name.rangeOfCharacterFromSet(.whitespaceAndNewlineCharacterSet())
    if let range = range where range.startIndex != range.endIndex {
        return false
    }

    return true
}
```

Tablice i słowniki

Tablice i słowniki

- Deklaruje się je za pomocą nawiasów kwadratowych

Tablice i słowniki

- Deklaruje się je za pomocą nawiasów kwadratowych
- Dostęp do elementów też przez nawiasy

```
var shoppingList = ["catfish", "water", "tulips",
    "blue paint"]
shoppingList[1] = "bottle of water"
```

```
["catfish", "water", "tulips", "blue...
"bottle of water"
```

Tablice i słowniki

- Deklaruje się je za pomocą nawiasów kwadratowych
- Dostęp do elementów też przez nawiasy

```
var shoppingList = ["catfish", "water", "tulips",
    "blue paint"]
shoppingList[1] = "bottle of water"
```

```
["catfish", "water", "tulips", "blue...
"bottle of water"
```

- Słowniki klucz:wartość

```
var occupations = [
    "Malcolm": "Captain",
    "Kaylee": "Mechanic",
]
occupations["Jayne"] = "Public Relations"
```

```
["Kaylee": "Mechanic", "Malcolm"...
{Some "Public Relations"}
```

Tablice i słowniki

- Deklaruje się je za pomocą nawiasów kwadratowych
- Dostęp do elementów też przez nawiasy

```
var shoppingList = ["catfish", "water", "tulips",
    "blue paint"]
shoppingList[1] = "bottle of water"
```

```
["catfish", "water", "tulips", "blue...
"bottle of water"
```

- Słowniki klucz:wartość

```
var occupations = [
    "Malcolm": "Captain",
    "Kaylee": "Mechanic",
]
occupations["Jayne"] = "Public Relations"
```

```
["Kaylee": "Mechanic", "Malcolm"...
{Some "Public Relations"}
```

- Aby utworzyć puste korzystamy ze składni konstruktorów

```
let emptyArray = [String]()
let emptyDictionary = [String: Float]()
```

```
0 elements
0 key/value pairs
```

```
shoppingList = []
occupations = [:]
```

```
0 elements
0 key/value pairs
```

Krotki

Krotki

- Wartości można grupować w krotki (tuples)

```
let http404Error = (404, "Not Found")
// http404Error is of type (Int, String), and
        equals (404, "Not Found")
```

Krotki

- Wartości można grupować w krotki (tuples)

```
let http404Error = (404, "Not Found")
// http404Error is of type (Int, String), and
        equals (404, "Not Found")
```

- Krotka może zawierać dowolną kombinację typów

Krotki

- Wartości można grupować w krotki (tuples)

```
let http404Error = (404, "Not Found")
// http404Error is of type (Int, String), and
        equals (404, "Not Found")
```

- Krotka może zawierać dowolną kombinację typów

- Krotkę można zdekomponować na składowe

```
let (statusCode, statusMessage) = http404Error
println("The status code is \(statusCode)")
// prints "The status code is 404"
println("The status message is \(statusMessage)")
// prints "The status message is Not Found"
```

Krotki

Krotki

- niepotrzebne wartości można zastąpić podkreśleniem

```
let (justTheStatusCode, _) = http404Error
println("The status code is \
        (justTheStatusCode)")
// prints "The status code is 404"
```

Krotki

- niepotrzebne wartości można zastąpić podkreśleniem

```
let (justTheStatusCode, _) = http404Error
println("The status code is \
        (justTheStatusCode)")
// prints "The status code is 404"
```

- do składowych można się odwołać przez indeks

```
println("The status code is \(http404Error.0)")
// prints "The status code is 404"
println("The status message is \
        (http404Error.1)")
// prints "The status message is Not Found"
```

Krotki

Krotki

- składowe mogą mieć swoje etykiety

Krotki

- składowe mogą mieć swoje etykiety

```
let http200Status = (statusCode: 200,
        description: "OK")
```

Krotki

- składowe mogą mieć swoje etykiety

```
let http200Status = (statusCode: 200,
        description: "OK")
```

```
println("The status code is \
        (http200Status.statusCode)")
// prints "The status code is 200"
println("The status message is \
        (http200Status.description)")
// prints "The status message is OK"
```

Instrukcje

Instrukcje

- instrukcje warunkowe `if` i `switch`

Instrukcje

- instrukcje warunkowe `if` i `switch`
  - warunek musi być zawsze boolowski

Instrukcje

- instrukcje warunkowe `if` i `switch`
  - warunek musi być zawsze boolowski
- pętle `for-in`, `for`, `while`, `do-while`

Instrukcje

- instrukcje warunkowe `if` i `switch`
  - warunek musi być zawsze boolowski
- pętle `for-in`, `for`, `while`, `do-while`

```
let individualScores = [75, 43, 103, 87, 12]
var teamScore = 0
for score in individualScores {
    if score > 50 {
        teamScore += 3
    } else {
        teamScore += 1
    }
}
println(teamScore)
```

```
[75, 43, 103, 87, 12]
0


(3 times)

(2 times)


"11"
```

Instrukcja warunkowa

Instrukcja warunkowa

- Jeżeli chcemy sprawdzić, czy zmienna ma wartość w instrukcji warunkowej można wykorzystać operator let

Instrukcja warunkowa

- Jeżeli chcemy sprawdzić, czy zmienna ma wartość w instrukcji warunkowej można wykorzystać operator let

```
var optionalName: String? = "John Appleseed"
var greeting = "Hello!"
if let name = optionalName {
    greeting = "Hello, \(name)"
}
```

```
{Some "John Appleseed"}
"Hello!"

"Hello, John Appleseed"
```

Instrukcja warunkowa

- Jeżeli chcemy sprawdzić, czy zmienna ma wartość w instrukcji warunkowej można wykorzystać operator let

```
var optionalName: String? = "John Appleseed"
var greeting = "Hello!"
if let name = optionalName {
    greeting = "Hello, \(name)"
}
```

{Some "John Appleseed"}
"Hello!"

"Hello, John Appleseed"

```
var optionalName: String? = nil
var greeting = "Hello!"
if let name = optionalName {
    greeting = "Hello, \(name)"
}
```

nil
"Hello!"

Instrukcja `switch`

Instrukcja `switch`

- Nie jest ograniczona do typów prostych

Instrukcja `switch`

- Nie jest ograniczona do typów prostych
- Nie jest ograniczona do równości

Instrukcja `switch`

- Nie jest ograniczona do typów prostych
- Nie jest ograniczona do równości
- Musi być wyczerpująca - klauzula `default`

Instrukcja `switch`

- Nie jest ograniczona do typów prostych
- Nie jest ograniczona do równości
- Musi być wyczerpująca - klauzula `default`

```
let vegetable = "red pepper"
switch vegetable {
case "celery":
    let vegetableComment = "Add some raisins and make ants on a log."
case "cucumber", "watercress":
    let vegetableComment = "That would make a good tea sandwich."
case let x where x.hasSuffix("pepper"):
    let vegetableComment = "Is it a spicy \(x)?"
default:
    let vegetableComment = "Everything tastes good in soup."
}
```

"red pepper"

"Is it a spicy red pepper?"

Instrukcja `switch`

- Nie jest ograniczona do typów prostych
- Nie jest ograniczona do równości
- Musi być wyczerpująca - klauzula `default`

```
let vegetable = "red pepper"
switch vegetable {
case "celery":
    let vegetableComment = "Add some raisins and make ants on a log."
case "cucumber", "watercress":
    let vegetableComment = "That would make a good tea sandwich."
case let x where x.hasSuffix("pepper"):
    let vegetableComment = "Is it a spicy \(x)?"
default:
    let vegetableComment = "Everything tastes good in soup."
}
```

"red pepper"

"Is it a spicy red pepper?"

```
var temperature = 83

switch (temperature)
{
case 0...49:
    println("Cold")
case 50...79:
    println("Warm")
case 80...110:
    println("Hot")
default:
    println("Temperature out of range")
}
```

83

"Hot"

Instrukcja `switch`

Instrukcja `switch`

- W przeciwieństwie do innych języków Swift automatycznie kończy `switch` po znalezieniu pasującego przypadku

Instrukcja `switch`

- W przeciwieństwie do innych języków Swift automatycznie kończy **`switch`** po znalezieniu pasującego przypadku
- Można użyć **`break`**, gdy tego potrzeba

Instrukcja `switch`

- W przeciwieństwie do innych języków Swift automatycznie kończy **`switch`** po znalezieniu pasującego przypadku
- Można użyć **`break`**, gdy tego potrzeba
- Można to zmienić instrukcją **`fallthrough`**

Instrukcja `switch`

- W przeciwieństwie do innych języków Swift automatycznie kończy **`switch`** po znalezieniu pasującego przypadku
- Można użyć **`break`**, gdy tego potrzeba
- Można to zmienić instrukcją **`fallthrough`**

```
var temperature = 60
switch (temperature)
{
case 0...49 where temperature % 2 == 0:
    println("Cold and even")
    fallthrough
case 50...79 where temperature % 2 == 0:
    println("Warm and even")
    fallthrough
case 80...110 where temperature % 2 == 0:
    println("Hot and even")
    fallthrough
default:
    println("Temperature out of range or odd")
}
```

```
60

"Warm and even"

"Hot and even"

"Temperature out of range or odd"
```

Instrukcja `for-in`

Instrukcja `for-in`

- Umożliwia iterowanie po tablicach i słownikach

Instrukcja `for-in`

- Umożliwia iterowanie po tablicach i słownikach

```
let interestingNumbers = [
    "Prime": [2, 3, 5, 7, 11, 13],
    "Fibonacci": [1, 1, 2, 3, 5, 8],
    "Square": [1, 4, 9, 16, 25],
]
var largest = 0
for (kind, numbers) in interestingNumbers {
    for number in numbers {
        if number > largest {
            largest = number
        }
    }
}
println(largest)
```

```
["Prime": [2, 3, 5, 7, 11,...

0

(8 times)

"25"
```

Instrukcja `for-in`

- Umożliwia iterowanie po tablicach i słownikach

```
let interestingNumbers = [
    "Prime": [2, 3, 5, 7, 11, 13],
    "Fibonacci": [1, 1, 2, 3, 5, 8],
    "Square": [1, 4, 9, 16, 25],
]
var largest = 0
for (kind, numbers) in interestingNumbers {
    for number in numbers {
        if number > largest {
            largest = number
        }
    }
}
println(largest)
```

```
["Prime": [2, 3, 5, 7, 11,...
0
(8 times)
"25"
```

```
let interestingNumbers = [
    "Prime": [2, 3, 5, 7, 11, 13],
    "Fibonacci": [1, 1, 2, 3, 5, 8],
    "Square": [1, 4, 9, 16, 25],
]
var largest = 0
var k = "None"
for (kind, numbers) in interestingNumbers {
    for number in numbers {
        if number > largest {
            largest = number
            k=kind
        }
    }
}
println("\(largest) in \(k)")
```

```
["Prime": [2, 3, 5, 7, 11,...
0
"None"
(8 times)
(8 times)
"25 in Square"
```

Pętle

..<

...

Pętle

- `while`, `do-while`

```
var n = 2
while n < 100 {
    n = n * 2
}
println(n)

var m = 2
do {
    m = m * 2
} while m < 100
println(m)
```

```
2

(6 times)

"128"

2

(6 times)

"128"
```

- `..<`

- `...`

Pętle

- while, do-while

```
var n = 2
while n < 100 {
    n = n * 2
}
println(n)

var m = 2
do {
    m = m * 2
} while m < 100
println(m)
```

```
2

(6 times)

"128"

2

(6 times)

"128"
```

- for

..<

...

Pętle

- while, do-while

```
var n = 2
while n < 100 {
    n = n * 2
}
println(n)

var m = 2
do {
    m = m * 2
} while m < 100
println(m)
```

```
2

(6 times)

"128"

2

(6 times)

"128"
```

- for

```
var firstForLoop = 0
for i in 0...4 {
    firstForLoop += i
}
println(firstForLoop)
```

```
0

(5 times)

"10"
```

..<

...

Pętle

- while, do-while

```
var n = 2
while n < 100 {
    n = n * 2
}
println(n)

var m = 2
do {
    m = m * 2
} while m < 100
println(m)
```

```
2

(6 times)

"128"

2

(6 times)

"128"
```

- for

```
var firstForLoop = 0
for i in 0..<4 {
    firstForLoop += i
}
println(firstForLoop)

var secondForLoop = 0
for var i = 0; i < 4; ++i {
    secondForLoop += i
}
println(secondForLoop)
```

```
0

(4 times)

"6"

0

(4 times)

"6"
```

..<

```
var firstForLoop = 0
for i in 0...4 {
    firstForLoop += i
}
println(firstForLoop)
```

```
0

(5 times)

"10"
```

...

Pętle

Pętle

- przerwanie pętli - break

```
var j = 10

for var i = 0; i < 100; ++i
{
    j += j

    if j > 100 {
        break
    }

    println("j = \(j)")
}
```

Pętle

- przerwanie pętli - break

```
var j = 10

for var i = 0; i < 100; ++i
{
    j += j

    if j > 100 {
        break
    }

    println("j = \(j)")
}
```

- zakończenie bieżącego przebiegu - continue

```
var i = 1

while i < 20
{
    ++i

    if (i % 2) != 0 {
        continue
    }

    println("i = \(i)")
}
```

Funkcje

Funkcje

- Do deklaracji funkcji służy słowo kluczowe func

Funkcje

- Do deklaracji funkcji służy słowo kluczowe func
- Argumenty oddzielone przecinkami, poprzedzone etykietami i dwukropkiem

Funkcje

- Do deklaracji funkcji służy słowo kluczowe func
- Argumenty oddzielone przecinkami, poprzedzone etykietami i dwukropkiem
- Typ funkcji po argumentach oddzielony ->

Funkcje

- Do deklaracji funkcji służy słowo kluczowe func
- Argumenty oddzielone przecinkami, poprzedzone etykietami i dwukropkiem
- Typ funkcji po argumentach oddzielony ->

```
func greet(name: String, day: String) -> String {
    return "Hello \(name), today is \(day)."
}
greet("Bob", "Tuesday")
```

"Hello Bob, today is Tues...

"Hello Bob, today is Tues...

Funkcje

Funkcje

- Argumenty funkcji mogą mieć etykiety

Funkcje

- Argumenty funkcji mogą mieć etykiety
- Trzeba z nich wtedy korzystać w trakcie wywołania

```
func buildMessage(#name: String, #count: Int) -> String {
    return("\(name), you are customer number \(count)")
}

var message = buildMessage(name: "John", count: 100)
```

"John, you are customer number 100"

"John, you are customer number 100"

Funkcje

- Argumenty funkcji mogą mieć etykiety
- Trzeba z nich wtedy korzystać w trakcie wywołania

```
func buildMessage(#name: String, #count: Int) -> String {
    return("\(name), you are customer number \(count)")
}

var message = buildMessage(name: "John", count: 100)
```

```
"John, you are customer number 100"


"John, you are customer number 100"
```

- Argumenty mogą mieć wartości domyślne

```
func buildMessage(count: Int, name: String = "Customer") -> String {
    return ("\(name), you are customer number \(count)")
}

var message = buildMessage(100)
println(message)

message=buildMessage(50, name: "John")
```

```
(2 times)


"Customer, you are customer number 100"
"Customer, you are customer number 100"

"John, you are customer number 50"
```

Funkcje

Funkcje

- Funkcja może mieć wiele wartości, identyfikowanych przez etykiety lub indeks

Funkcje

- Funkcja może mieć wiele wartości, identyfikowanych przez etykiety lub indeks

```
func calculateStatistics(scores: [Int]) -> (min: Int, max: Int, sum: Int) {
    var min = scores[0]                                                    5
    var max = scores[0]                                                    5
    var sum = 0                                                            0

    for score in scores {
        if score > max {
            max = score                                                    100
        } else if score < min {
            min = score                                                    3
        }
        sum += score                                                       (5 times)
    }

    return (min, max, sum)                                                 (.0 3, .1 100, .2 120)
}
let statistics = calculateStatistics([5, 3, 100, 3, 9])                    (.0 3, .1 100, .2 120)
println(statistics.sum)                                                    "120"
println(statistics.2)                                                      "120"
```

Funkcje

Funkcje

- Liczba argumentów funkcji może być zmienna pod warunkiem, że są tego samego typu

Funkcje

- Liczba argumentów funkcji może być zmienna pod warunkiem, że są tego samego typu

```
func sumOf(numbers: Int...) -> Int {
    var sum = 0
    for number in numbers {
        sum += number
    }
    return sum
}
sumOf()
sumOf(42, 597, 12)
sumOf(10,20,30,40,50,60,70,90)
```

(3 times)

(11 times)

(3 times)

0
651
370

Funkcje

Funkcje

- Domyślnie argumenty są przekazywane jako stałe - nie można ich modyfikować

Funkcje

- Domyślnie argumenty są przekazywane jako stałe - nie można ich modyfikować
- Można to zmienić deklarując argument jako zmienną

Funkcje

- Domyślnie argumenty są przekazywane jako stałe - nie można ich modyfikować
- Można to zmienić deklarując argument jako zmienną
- Nie zmienia to wartości poza funkcją

Funkcje

- Domyślnie argumenty są przekazywane jako stałe - nie można ich modyfikować
- Można to zmienić deklarując argument jako zmienną
- Nie zmienia to wartości poza funkcją

```
func calcuateArea (var length: Float, var width: Float) -> Float {
    length = length * 2.54
    width = width * 2.54
    return length * width
}

println(calcuateArea(10, 20))
```

```
25.4
50.8
1290.32


"1290.32"
```

Funkcje

Funkcje

- Argumenty wyjściowe można uzyskać przez słowo kluczowe `inout`

Funkcje

- Argumenty wyjściowe można uzyskać przez słowo kluczowe **`inout`**
- Przekazywany argument musi być wtedy poprzedzony **&**

Funkcje

- Argumenty wyjściowe można uzyskać przez słowo kluczowe **`inout`**
- Przekazywany argument musi być wtedy poprzedzony **&**

```
var myValue = 10

func doubleValue (inout value: Int) -> Int {
    value += value
    return(value)
}

println("Before function call myValue = \(myValue)")

println("doubleValue call returns \(doubleValue(&myValue))")

println("After function call myValue = \(myValue)")
```

```
10


20
20


"Before function call myValue = 10"

"doubleValue call returns 20"

"After function call myValue = 20"
```

Do zobaczenia

速