

# SwiftUI



ЕГОР ПЕТРОВ



**ЕГОР ПЕТРОВ**

iOS Engineer, фирма Agora



# План занятия

1. [Предыстория SwiftUI](#)
2. [SwiftUI. Основы](#)
3. [Базовые компоненты SwiftUI](#)
4. [UIKit и SwiftUI](#)
5. [Кроссплатформенная разработка](#)
6. [Итоги](#)
7. [Домашнее задание](#)



# Предыстория SwiftUI

---

# История SwiftUI

**Swift** – язык программирования

**SwiftUI** – фреймворк для создания iOS-приложений



---

# Что такое SwiftUI



**SwiftUI**  
Better apps. Less code.

# Что такое SwiftUI

**SwiftUI** – это набор инструментов пользовательского интерфейса. Он может похвастаться декларативным синтаксисом, стилем программирования, уже популярным среди веб-разработчиков, использующих JavaScript.

**Важно**, что SwiftUI дает возможности для разработки приложений для iOS 13 и новее (без обратной совместимости).

На сегодня функционал SwiftUI не такой широкий, как в UIKit, но потенциал есть для его развития, Apple готовы делать серьезные вложения в развитие фреймворка.



# Основы SwiftUI



# Основы SwiftUI

Подход к разработке пользовательского интерфейса в **SwiftUI** полностью отличается от **UIKit**.

Он использует декларативный синтаксис для определения представлений исключительно в коде Swift.

```
13     var body: some View {  
14         VStack {  
15             Slider(  
16                 value: $speed,  
17                 in: 0...100,  
18                 onEditingChanged: { editing in  
19                     isEditing = editing  
20             }  
21         )  
22         Text("\(speed)")  
23             .foregroundColor(isEditing ? .red :  
24                 .blue)  
25     }
```



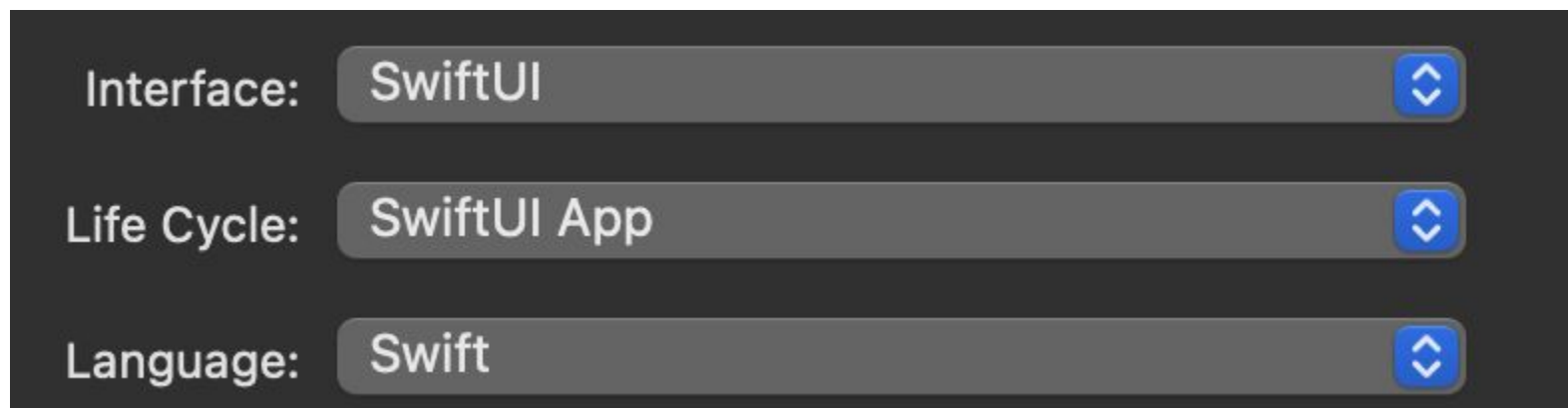
# Основы SwiftUI

**SwiftUI** – это фреймворк, написанный для Swift. Он использует преимущества Swift, чтобы сделать его декларативный синтаксис читабельным.

SwiftUI поощряет использование протоколов вместо наследования классов за счет использования протоколов для основных строительных блоков платформы, таких как View.

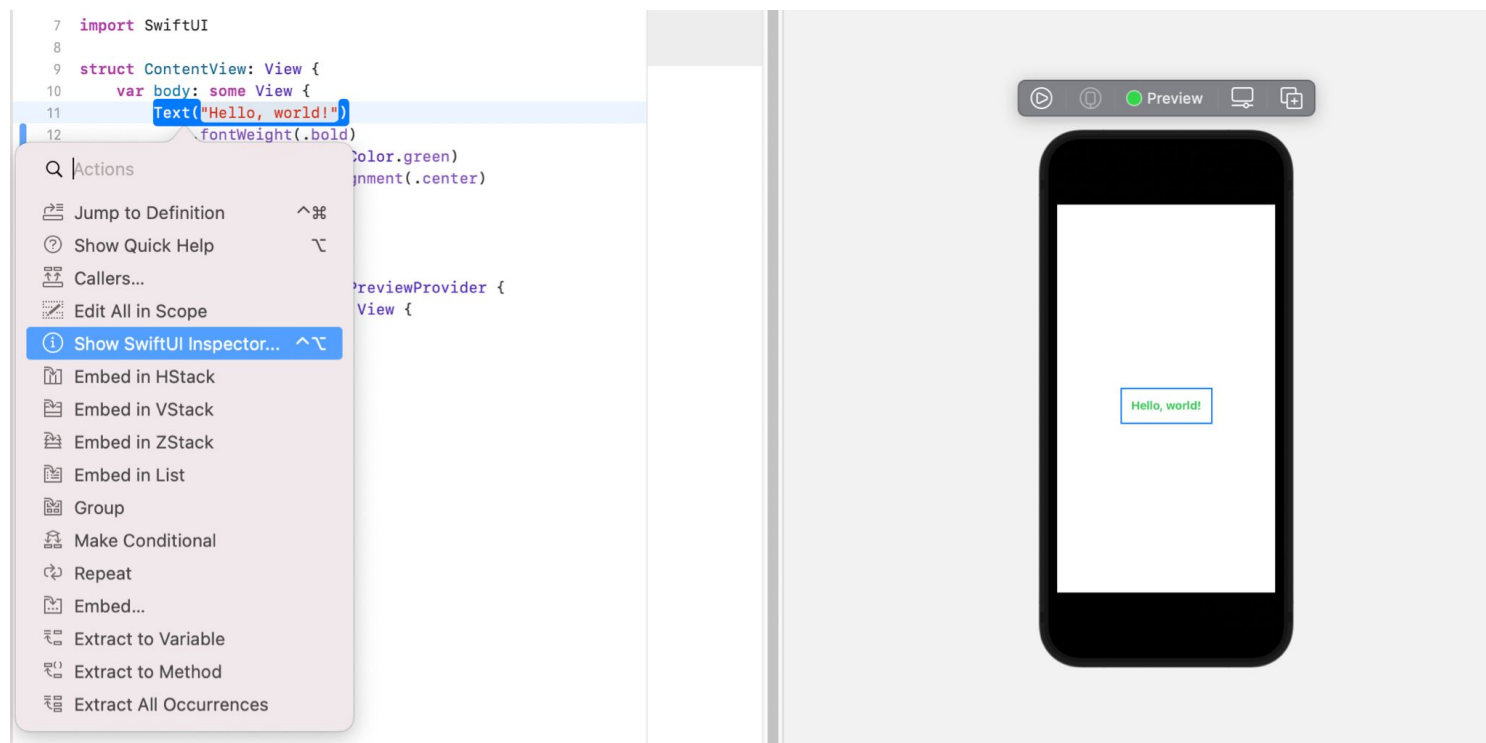
# Основы SwiftUI – создаем первое демо

Для создания на SwiftUI при выборе опций для проекта необходимо выбрать:



# SwiftUI Inspector

На демо можно изменять свойства текста с помощью SwiftUI inspector **command+click** (на Text) → **SwiftUI Inspector**.



# Модификаторы

**Модификатор** принимает представление (или другой модификатор). Он делает его копию и возвращает измененный вид после внесения изменений в его стили.

SwiftUI предоставляет ряд встроенных модификаторов – `font()`, `background()` и прочие.

Модификаторы вы можете найти [по ссылке](#) (для каждого компонента на отдельной странице).

# Модификаторы

Иногда порядок применения модификаторов представления влияет на результат пользовательского интерфейса.

Пример:

Если сначала добавить к тексту `padding()`, а после `background(Color.orange)` мы получим один бэкграунд, а если поменять их местами, то другой. Попробуйте сделать это в своем проекте и посмотреть на результат.

# Кастомные модификаторы

Мы можем создавать свои модификаторы:

```
struct Title: ViewModifier {  
    func body(content: Content) -> some View {  
        content  
            .font(.largeTitle)  
            .foregroundColor(.white)  
            .padding()  
            .background(Color.blue)  
            .clipShape(RoundedRectangle(cornerRadius: 10))  
    }  
}
```

# Кастомные модификаторы

Теперь с применением точечного синтаксиса у нас есть возможность применить модификатор `Title` к тексту. Модификатор так и называется – `modifier`.

```
Text("Hello World")  
    .modifier>Title()
```





# Контейнеры SwiftUI

Для помещения нескольких объектов на экран необходимо использовать стеки (stack) SwiftUI.

Использование стеков в SwiftUI позволяет упорядочить несколько представлений в одно согласованное представление с определенными свойствами.

---

# Контейнеры SwiftUI

- **HStack** позволяет расположить свои дочерние представления по горизонтали.
- **VStack** позволяет расположить свои дочерние представления по вертикальной линии.
- **ZStack** позволяет перекрывать свои дочерние представления друг над другом.

Стеки могут быть дополнительно настроены с выравниванием и интервалом, чтобы изменить их внешний вид.



# Базовые компоненты SwiftUI

# Базовые компоненты



---

# Базовые компоненты SwiftUI

**Базовые компоненты в SwiftUI** – это визуальные строительные блоки пользовательского интерфейса вашего приложения.

Используйте их для отображения содержимого вашего приложения на экране.

- Представления – текст, изображения, фигуры, пользовательские рисунки и композиции из любого из них вместе.
- Элементы управления позволяют взаимодействовать с пользователем с помощью согласованных API-интерфейсов, которые адаптируются к их платформе и контексту.

---

# Примеры базовых компонентов

- Picker
- Toggle
- Slider
- Table
- Form
- Alert и прочие

---

## Базовые компоненты – демо

- Form и Toggle
- Slider и Text
- Buttons и alerts



# UIKit и SwiftUI



# UIKit в SwiftUI

Существует способ использовать вместе с SwiftUI представления из исходных фреймворков. Например, из UIKit.

Это осуществляется в форме таких протоколов, как `UIViewRepresentable` – это оболочка для представления UIKit, которую вы используете для интеграции этого представления в иерархию представлений SwiftUI.

Больше [по ссылке](#)


# Использование UIKit компонентов

SwiftUI без проблем работает с существующими фреймворками пользовательского интерфейса на всех платформах Apple.

Например, вы можете разместить представления UIKit и контроллеры представлений внутри представлений SwiftUI и наоборот.

- SwiftUI дает возможность использовать представления из исходных фреймворков, таких как UIKit. Это осуществляется в форме таких протоколов, как `UIViewRepresentable`.
- Представление SwiftUI может соответствовать этому протоколу и, реализуя его функцию, оборачивать UIView таким образом, чтобы он работал в новой структуре.

Больше [по ссылке](#)



# Кроссплатформенная разработка



# Кроссплатформенная разработка

**SwiftUI** — это инновационный, исключительно простой способ создания пользовательских интерфейсов на всех платформах Apple с помощью Swift.

Создавайте пользовательские интерфейсы для любого устройства Apple, используя всего один набор инструментов и API.

---

# Кроссплатформенная разработка, преимущества SwiftUI

1. Поддержка: SwiftUI поддерживает все платформы Apple (iOS, iPadOS, macOS, watchOS и tvOS).
2. Быстрая перезагрузка: вы видите всю свою работу сразу на canvas в Live Preview.
3. SwiftUI с Combine: согласно документации Apple. Платформа Combine предоставляет декларативный API Swift для обработки значений во времени.
4. SwiftUI очень быстро развивается. Скоро вы сможете создавать не только приложения для iOS и других платформ Apple, но и для устройств Android, Windows.

---

# WWDC: дополнительные материалы

- [SwiftUI Essentials](#)
- [SwiftUI on watchOS](#)
- [Stacks, Grids, and Outlines in SwiftUI](#)
- [Demystify SwiftUI](#)
- [What's new in SwiftUI](#)

---

# Дополнительные материалы

- [SwiftUI: Getting Started | raywenderlich.com](https://raywenderlich.com/swiftui/getting-started)

Документация Apple:

- [SwiftUI](https://developer.apple.com/documentation/swiftui)
- [SwiftUI Tutorials](https://developer.apple.com/tutorials/swiftui/)
- [Views and Controls](https://developer.apple.com/documentation/uikit/views_and_controls/)

# Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** учебной группы.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.



**Задавайте вопросы и  
пишите отзыв о лекции!**

**Егор Петров**