Redux Introduction

|  |  |  |  |
| --- | --- | --- | --- |
| № урока: | 1 | Курс: | React Advanced |
|  | | |  |
| Средства обучения: | | | Текстовый редактор или IDE, браузер, Node.js |

# Обзор, цель и назначение урока

В этом уроке мы познакомимся с библиотекой Redux, узнаем где и почему ее стоит использовать, ознакомимся с ключевыми преимуществами этой библиотеки перед Flux, поймем основные концепции управления потоком данных в Redux и узнаем о ее ключевых принципах в построении React приложений.

**Изучив материал данного занятия, учащийся сможет:**

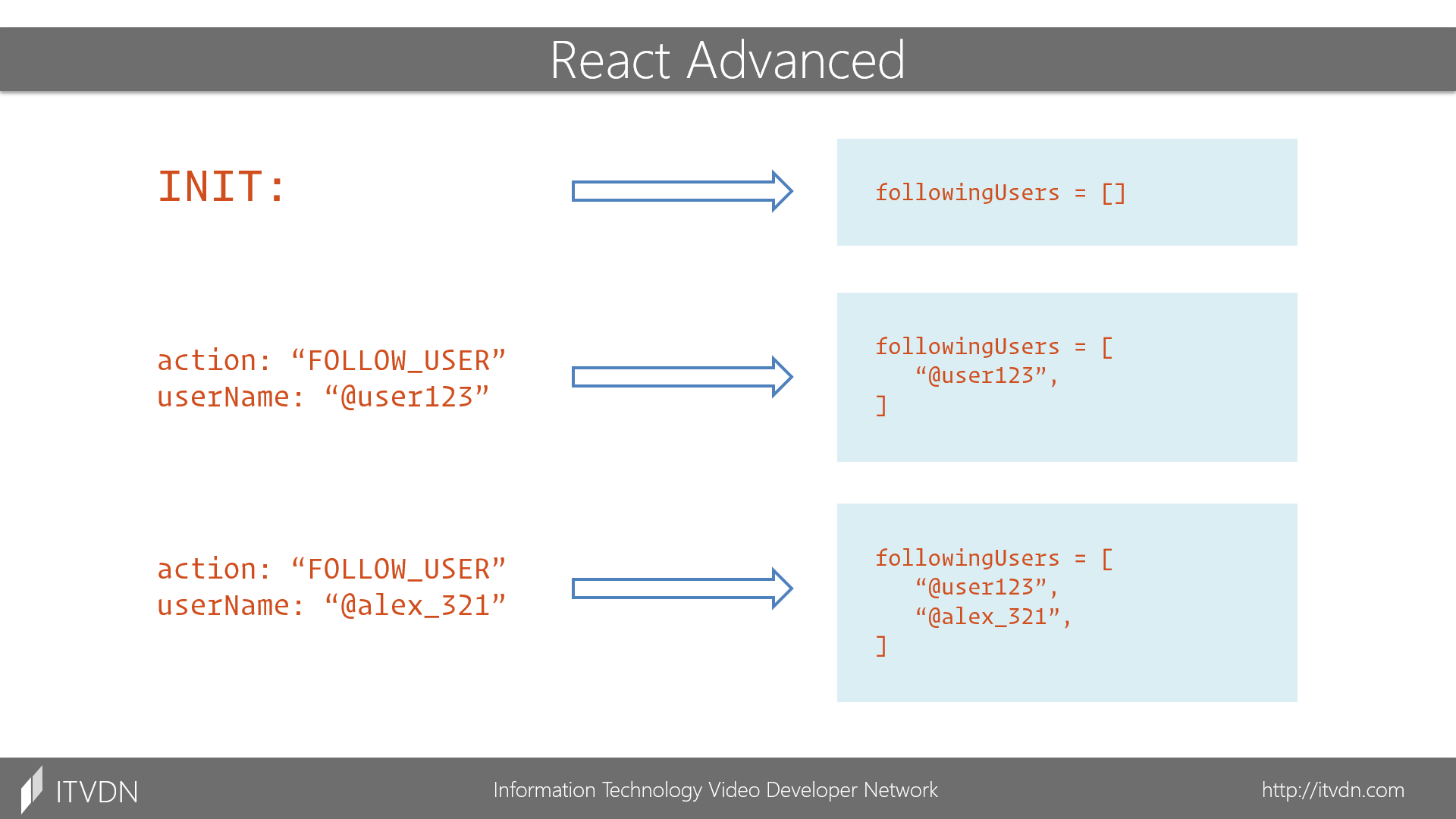
* Иметь представление о том, как осуществляется управление потоком данных в Redux
* Понимать в каких случаях нужен Redux, а когда можно обойтись и без него
* Узнать о ключевых принципах и аспектах Redux
* Понимать общую картину управления data-flow в приложениях с использованием Redux

# Содержание урока

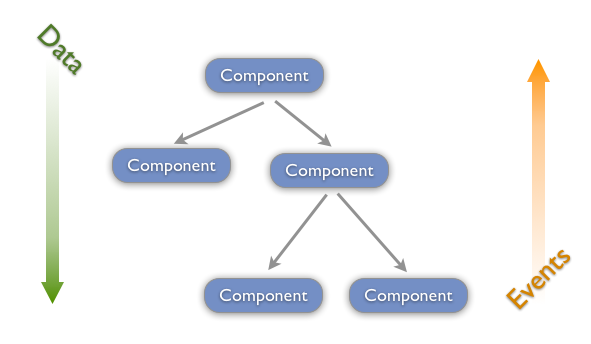
1. Redux. Его преимущества и особенности
2. Когда стоит использовать Redux
3. Основные принципы Redux
4. Концепция управления потоком данных в Redux

# Резюме

**Redux** – это библиотека для удобного управления потоком данных (data-flow) в React приложениях. Он позиционирует себя, как предсказуемый контейнер состояния для вашего JS приложения и предлагает думать о приложении, как о каком-то начальном состоянии, которое изменяется последовательностью неких действия (или *Actions* в терминологии Redux).



Redux, как и Flux, использует *One way data-binding* (однонаправленный поток данных), это значит, что все данные «проваливаются» вниз, а события за счет коллбэков «всплывают» наверх.



Одними из основных фич в Redux являются *Hot Module Replacement* и *Time Travelling*, о которых Дэн Абрамов (автор Redux) на конференции React-Europe 2015 рассказывал и показывал на примерах ([ссылка](https://youtu.be/xsSnOQynTHs)). Это очень мощные инструменты, которые ускоряют разработку и делают ее приятной для вас.

**Hot Module Replacement** (сокращ. HMR) – это такой механизм, который позволяет вам делать какие-то изменения в коде какого-либо компонента и в браузере без перезагрузки эти изменения применятся конкретно к этому компоненту, не затрагивая весь остальной *state* и другие части UI отображения в вашем приложении ([ссылка на пример](http://i.imgur.com/yZCVLEC.gif)).

**Time Travelling** – позволяет нам отслеживать изменения в нашем приложении в контексте времени и благодаря специальному расширению в браузере даёт возможность «путешествовать во времени» перемещаясь вперед/назад или вовсе сбрасывать весь *state* до изначального вида ([ссылка на пример](http://imgur.com/a/sOwSx)).

**Redux Features:**

* Hot Module Replacement
* Time Travel
* Single Store (*Single Source of Truth*)
* Меньшее кол-во boilerplate кода
* Immutable Store (неизменный)
* Удобное и просто тестирование
* Просто и минималистичный API
* Точки расширения (middlewares)

**Нужен ли вам Redux?**

Не существует серебрянной пули для решения всех проблем, всегда нужно искать компомиссы, особенно это касается веб-разработки. Есть задачи, в которых Redux может и не понадобится, но если ваше приложение становится большим и сложным, то Redux упростит вам жизнь.

**Случаи, когда Redux нужен:**

* У вас есть гонки состояний (Race conditions)
* Ваш data-flow сложный и запутанный
* Большое кол-во действий (Actions)
* Вы нуждаетесь в переиспользовании одних

и тех же данных в разных местах приложения

**Redux: три основных принципа:**

1. Store – это единый источник правды (*Single Source of Truth*)
2. Actions вызывают изменения (но не делают их)
3. Reducers делают изменения и меняют Store

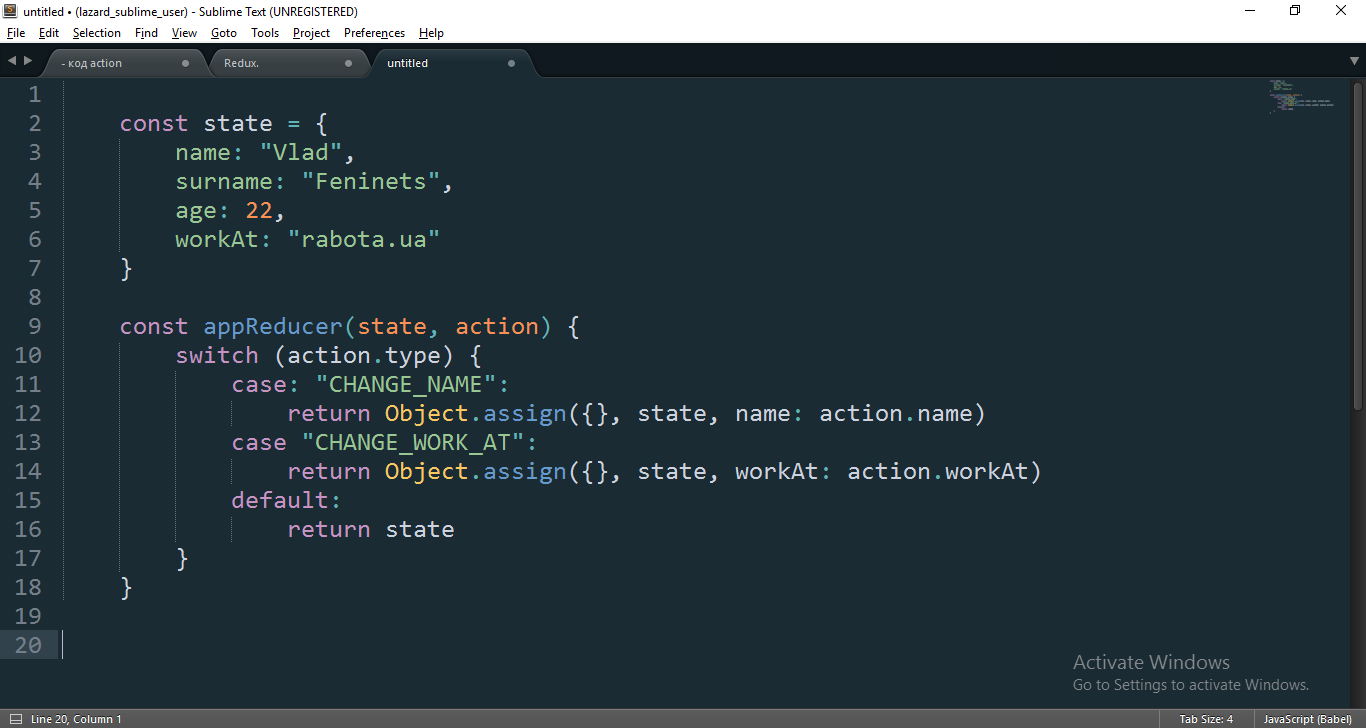
**Actions** – начнём с того, что это всего-навсего обычный (*plain*) Javascript объект. *Action* говорит о каких-то изменениях в вашем приложении, но не совершает их. Есть определенное [соглашение](https://github.com/acdlite/flux-standard-action), которое гласит, что у *Action* должно быть обязательно поле “type” и опционально какой-то “payload” с какими-либо данными (который может быть чем-угодно: строкой, объектом, числом и т.д.). Вызывается *Action* с помощью “store.dispatch({…})”.

Также есть такое понятие, как *Action Creators* – это функции, которые могут принимать параметры и возвращать объекты с нужными данными.

**Actions** всего лишь описывают какие-то изменения, а уже производят их

непосредственно **Reducers**.

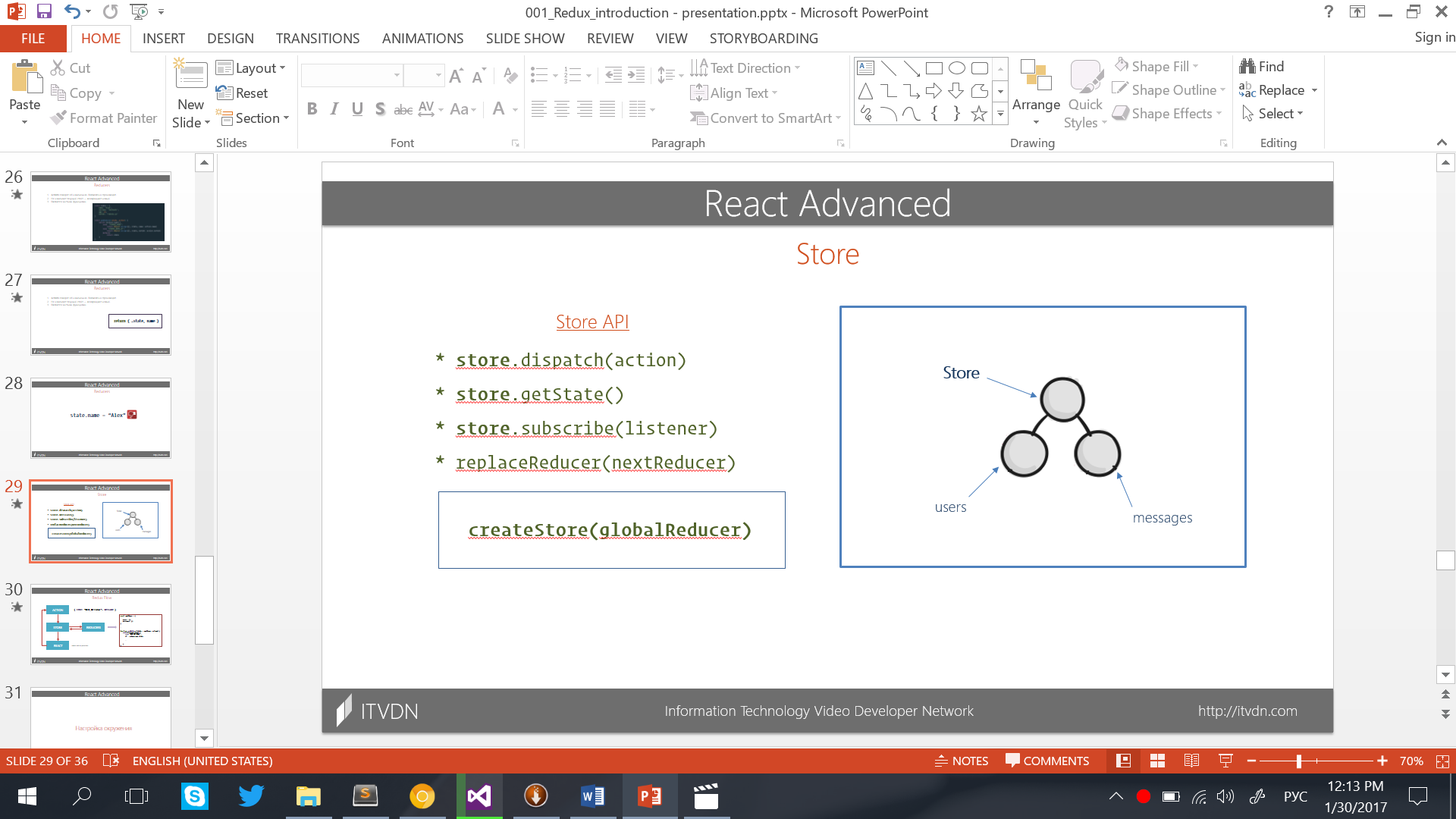
**Reducers** всегда должны быть чистыми функциями (*pure functions*) и возвращать новый объект с изменениями, а не делать их в текущем.



Не важно каким образом вы это будете делать, с помощью Object.assign() или spread-оператора ( { …state, newState } ) – главное всегда вернуть новый объект, это концептуально для Redux.

Давайте теперь перейдем к **Store.**

Сам по себе **Store** – это один большой объект, но он может быть декомпозирован, например, в одном объекте вы храните данные по юзерам, а в другом сообщения.



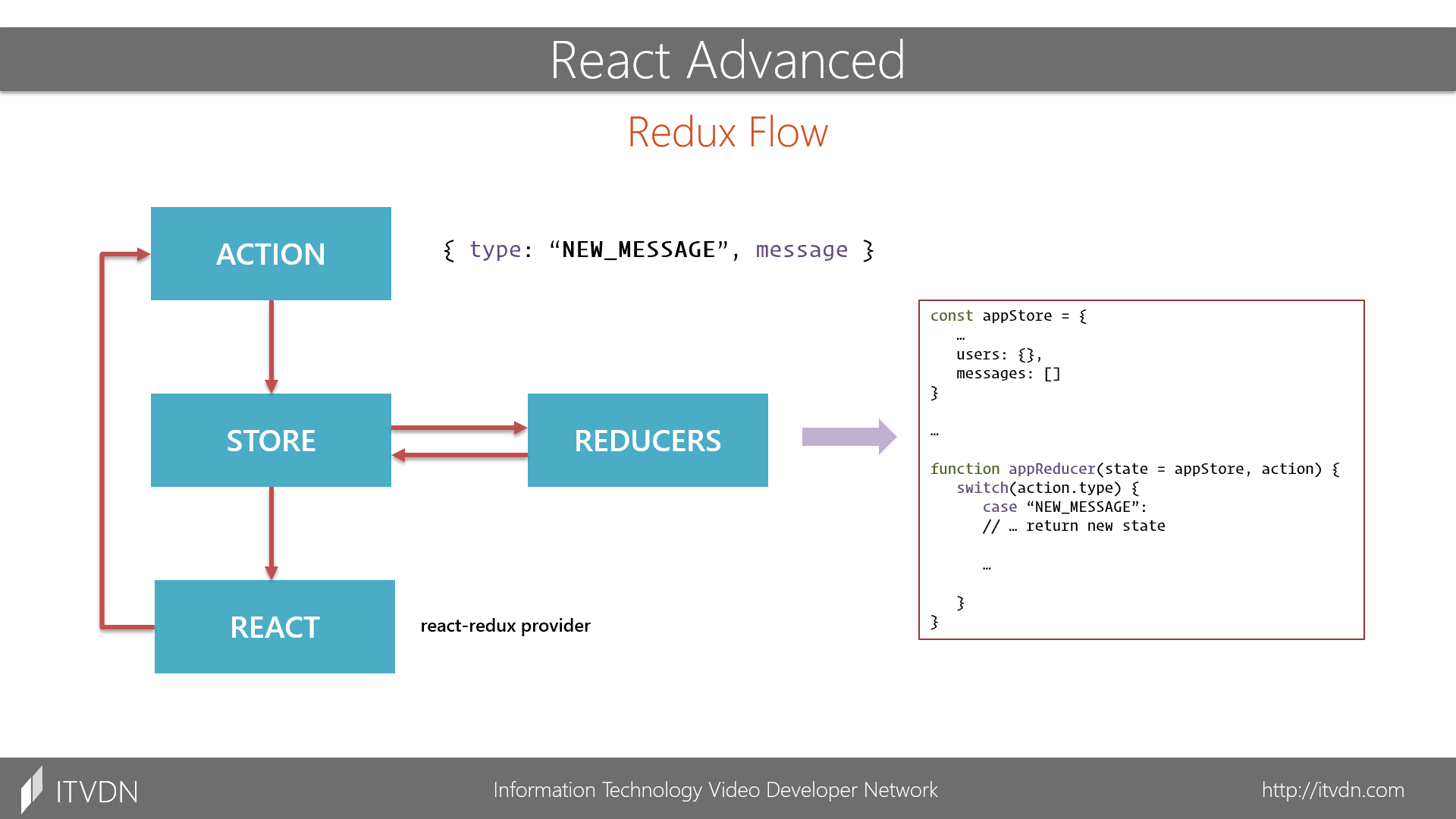
**Store** объединяет наши **Actions** и **Reducers.**

**Store** берет на себя такие задачи, как:

* Хранения состояния вашего приложения
* Предоставляет возможность обновления состояния при помощи метода store.dispatch
* Предоставляет доступ к чтению состояния с помощью метода store.getState
* С помощью метода store.subscribe можно подписывать слушателей, которые будут оповещены о изменениях в вашем **Store.**
* И метод replaceReducer нужен для корректной работы HMR.

Создается **Store** при помощи метода createStore, который в кач-ве аргумента принимает Top-Level Reducer (глобальный), который в свою очередь может импортировать другие редьюсеры (за счёт этого и возможно декомпозиция **Store**).

**Data-Flow в Redux**



# Закрепление материала

* Назовите как минимум 3 особенности Redux
* Для чего нужен Redux?
* Назовите 3 принципа Redux
* Что делают Actions?
* Что делают Reducers?
* Перечислите как минимум 2 метода Store и расскажите, что они делают

# Рекомендуемые ресурсы

Цикл статей по ES2015 от Ильи Кантора

<http://learn.javascript.ru/es-modern>

Лучшее из ES2015 (LearnAcademy)

<https://www.youtube.com/playlist?list=PLoYCgNOIyGACDQLaThEEKBAlgs4OIUGif>

Курс React Essential на ITVDN

<https://itvdn.com/ru/video/react-js-essential>

Выступление Дэна Абрамова на React Europe 2015

<https://youtu.be/xsSnOQynTHs>

Официальная документация Redux

<http://redux.js.org/>

Redux на русском

<https://gkp43215.gitbooks.io/redux-in-russian/>

Небольшой туториал по Redux

<https://maxfarseer.gitbooks.io/redux-course-ru/content/>