1) Для чего в WPF используются ресурсы? Каким образом можно определить новый ресурс и управлять им? Опишите назначение класса ResourceDictionary. Зачем каждый элемент имеет собственную коллекцию ресурсов?

Они позволяют определить объект любого типа один раз, а затем многократно использовать его в коде по мере необходимости, чаще всего - **для** **настройки** **визуальных** **элементов** **пользовательского** **интерфейса**. **ResourceDictionary** — Это репозиторий для ресурсов, используемых Xamarin.Forms приложением. Типичные ресурсы, которые хранятся в **ResourceDictionary** **ResourceDictionary** включения, шаблонах элементов управления, шаблонах данных, цветах и конвертерах

2) Какая разница между статическими и динамическими ресурсами?

**Динамический** **ресурс** подхватит это изменение, а **статический** не будет иметь ни малейшего понятия о том, что его кисть была заменена в коллекции Resources какой-то другой, и, следовательно, продолжит пользоваться исходной кистью ImageBrush.

3) Что такое триггеры ? Для чего в WPF используются триггеры? Назовите основные типы триггеров.

Тригерры позволяют декларативно задать некоторые действия, которые выполняются при изменении свойств стиля. Существует три вида триггеров:

* **Триггеры свойств**: вызываются в ответ на изменения свойствами зависимостей своего значения
* **Триггеры данных**: вызываются в ответ на изменения значений любых свойств (они необязательно должны быть свойствами зависимостей)
* **Триггеры событий**: вызываются в ответ на генерацию событий
* **Мультитриггеры**: вызываются при выполнении ряда условий

Я видел следующие триггеры:

* Trigger
* DataTrigger
* MultiTrigger
* MultiDataTrigger
* EventTrigger

4) Что такое локализация и как ее обеспечить.

**Локализация** — это не просто перевод сайта на другой язык, а пошаговое создание копии проекта с учётом особенностей определённого региона. В процессе **локализации** у сайта появляется новая языковая версия в виде поддомена или подпапки. Главная задача — максимально адаптировать проект под культурные и поведенческие характеристики целевой аудитории.

5) Что такое тема? Опишите процесс создания темы на основе ресурсов и стилей.

**Темы** создаются **на** **основе** элементов управления

6) Что такое шаблон и как его создать?

Все визуальные элементы в WPF уже имеют встроенные шаблоны, которые определяют визуальное дерево, структуру и даже поведение элементов. Однако мощь шаблонов состоит в том, что мы можем их переопределить по своему вкусу.

7) Зачем нужны пользовательские элементы управления? Как создать собственный элемент. Опишите члены класса UserControl.

8) Что такое привязка данных? В чем разница между следующими режимами привязки: OneWay, TwoWay и OneTime.

В WPF привязка (binding) является мощным инструментом программирования, без которого не обходится ни одно серьезное приложение.

Привязка подразумевает взаимодействие двух объектов: источника и приемника. Объект-приемник создает привязку к определенному свойству объекта-источника. В случае модификации объекта-источника, объект-приемник также будет модифицирован.

Некоторые свойства класса **Binding**:

* **ElementName**: имя элемента, к которому создается привязка
* **IsAsync**: если установлено в True, то использует асинхронный режим получения данных из объекта. По умолчанию равно False
* **Mode**: режим привязки
* **Path**: ссылка на свойство объекта, к которому идет привязка
* **TargetNullValue**: устанавливает значение по умолчанию, если привязанное свойство источника привязки имеет значение null
* **RelativeSource**: создает привязку относительно текущего объекта
* **Source**: указывает на объект-источник, если он не является элементом управления.
* **XPath**: используется вместо свойства path для указания пути к xml-данным

**Режимы привязки**

Свойство **Mode** объекта Binding, которое представляет режим привязки, может принимать следующие значения:

* **OneWay**: свойство объекта-приемника изменяется после модификации свойства объекта-источника.
* **OneTime**: свойство объекта-приемника устанавливается по свойству объекта-источника только один раз. В дальнейшем изменения в источнике никак не влияют на объект-приемник.
* **TwoWay**: оба объекта - применки и источник могут изменять привязанные свойства друг друга.
* OneWayToSource: объект-приемник, в котором объявлена привязка, меняет объект-источник.
* **Default**: по умолчанию (если меняется свойство TextBox.Text, то имеет значение TwoWay, в остальных случаях OneWay).

9) Объясните назначение интерфейса INotifyPropertyChanged

Интерфейс INotifyPropertyChanged: уведомляет клиента об изменении определенного значения свойства. Общее использование: при загрузке данных своевременно обновлять соответствующее имя загрузки данных. При использовании функции своевременно выводите соответствующее сообщение об ошибке.