**Отчет по Главе 6: Стили (Styles)**

**1. Введение**

Данная глава посвящена механизму стилизации элементов пользовательского интерфейса в .NET MAUI. Стили играют ключевую роль в создании консистентного и профессионального внешнего вида приложений, позволяя определять наборы свойств и применять их к элементам управления многократно. Использование стилей упрощает процесс разработки, облегчает редизайн и поддержку приложений, а также способствует разделению логики представления от визуального оформления. Освоение стилей является важным шагом для разработки качественных настольных и мобильных приложений.

**2. Основная часть**

**2.1. Основные концепции и принципы**

* **Объект Style**: Определяет коллекцию объектов Setter, каждый из которых устанавливает определенное свойство (Property) элемента управления в указанное значение (Value).
* **TargetType**: Обязательный атрибут для Style, указывающий тип элемента управления, к которому этот стиль может быть применен.
* **Размещение стилей:**
  + **Ресурсы приложения (Application.Resources в App.xaml):** Глобальные стили, доступные во всем приложении. Часто выносятся в отдельные словари ресурсов (например, Resources/Styles/Styles.xaml).
  + **Ресурсы страницы (ContentPage.Resources):** Локальные стили, доступные только на данной странице.
  + **Ресурсы элемента управления:** Стили, определенные в ресурсах конкретного элемента управления (менее распространено для общих стилей).
* **Типы стилей:**
  + **Явные стили (Explicit Styles):** Определяются с атрибутом x:Key и применяются к элементам с помощью Style="{StaticResource ИмяСтиля}".
  + **Неявные стили (Implicit Styles):** Определяются без x:Key, но с указанием TargetType. Такой стиль автоматически применяется ко всем элементам указанного типа в области видимости ресурсов, если им не задан явный стиль.
* **Наследование стилей (BasedOn):** Позволяет создавать новые стили на основе существующих, наследуя их сеттеры и добавляя или переопределяя некоторые из них. BasedOn="{StaticResource ИмяБазовогоСтиля}".
* **Динамические стили (DynamicResource):** В отличие от StaticResource, привязка к DynamicResource позволяет стилю (или отдельным его значениям) изменяться во время выполнения приложения.
* **Классы стилей (StyleClass):** Позволяют применять один или несколько "стилевых классов" к элементу. Это обеспечивает более гибкое смешивание стилей. Классы стилей определяются как обычные Style с x:Key, и затем этот ключ используется в атрибуте StyleClass.
* **Стили для конкретных платформ (OnPlatform):** Позволяет определять разные значения свойств стиля для разных операционных систем.

**2.2. Примеры кода с пояснениями**

Демонстрация стилей была реализована на странице StylesDemoPage.xaml с использованием стилей, определенных как глобально в App.xaml, так и локально на странице.

**2.2.1. Глобальный стиль в App.xaml**  
В App.xaml был определен явный стиль для кнопок:

Generated xml

<!-- Фрагмент App.xaml -->

<Application.Resources>

<ResourceDictionary>

<!-- ... MergedDictionaries ... -->

<Style x:Key="GlobalButtonStyle" TargetType="Button">

<Setter Property="BackgroundColor" Value="{StaticResource PrimaryColor}"/>

<Setter Property="TextColor" Value="White"/>

<Setter Property="CornerRadius" Value="8"/>

<Setter Property="Padding" Value="14,10"/>

<Setter Property="FontAttributes" Value="Bold"/>

</Style>

</ResourceDictionary>

</Application.Resources>

*Пояснение:* Этот стиль GlobalButtonStyle делает кнопки единообразными по всему приложению, если он применен явно.

**2.2.2. Стили на странице StylesDemoPage.xaml**

Generated xml

<!-- Фрагменты StylesDemoPage.xaml -->

<ContentPage.Resources>

<Style x:Key="PageHeaderLabelStyle" TargetType="Label">

<Setter Property="FontSize" Value="Title"/>

<Setter Property="FontAttributes" Value="Bold"/>

<!-- ... -->

</Style>

<!-- Неявный стиль для Entry -->

<Style TargetType="Entry">

<Setter Property="BackgroundColor" Value="AliceBlue"/>

<!-- ... -->

</Style>

<!-- Наследуемый стиль -->

<Style x:Key="InheritedButtonStyle" TargetType="Button" BasedOn="{StaticResource GlobalButtonStyle}">

<Setter Property="BorderColor" Value="Yellow"/>

<Setter Property="BorderWidth" Value="2"/>

<Setter Property="TextColor" Value="DarkBlue"/>

</Style>

<!-- Стиль, используемый как класс стиля -->

<Style x:Key="ErrorLabelClassStyle" TargetType="Label">

<Setter Property="TextColor" Value="Red"/>

<Setter Property="FontAttributes" Value="Italic"/>

</Style>

</ContentPage.Resources>

<!-- Применение стилей -->

<Label Text="Demonstration of Styles" Style="{StaticResource PageHeaderLabelStyle}"/>

<Button Text="Global Styled Button" Style="{StaticResource GlobalButtonStyle}" />

<Button Text="Page Explicit Styled Button" Style="{StaticResource InheritedButtonStyle}" />

<Entry Placeholder="This Entry uses implicit style" />

<Label Text="This is an error label." StyleClass="ErrorLabelClassStyle" />

<!-- Платформо-зависимый стиль -->

<Button Text="Platform-Specific Button">

<Button.Style>

<Style TargetType="Button" BasedOn="{StaticResource GlobalButtonStyle}">

<Setter Property="BackgroundColor">

<OnPlatform x:TypeArguments="Color">

<On Platform="Android" Value="LimeGreen"/>

<On Platform="iOS" Value="OrangeRed"/>

<On Platform="Windows" Value="DodgerBlue"/>

</OnPlatform>

</Setter>

</Style>

</Button.Style>

</Button>

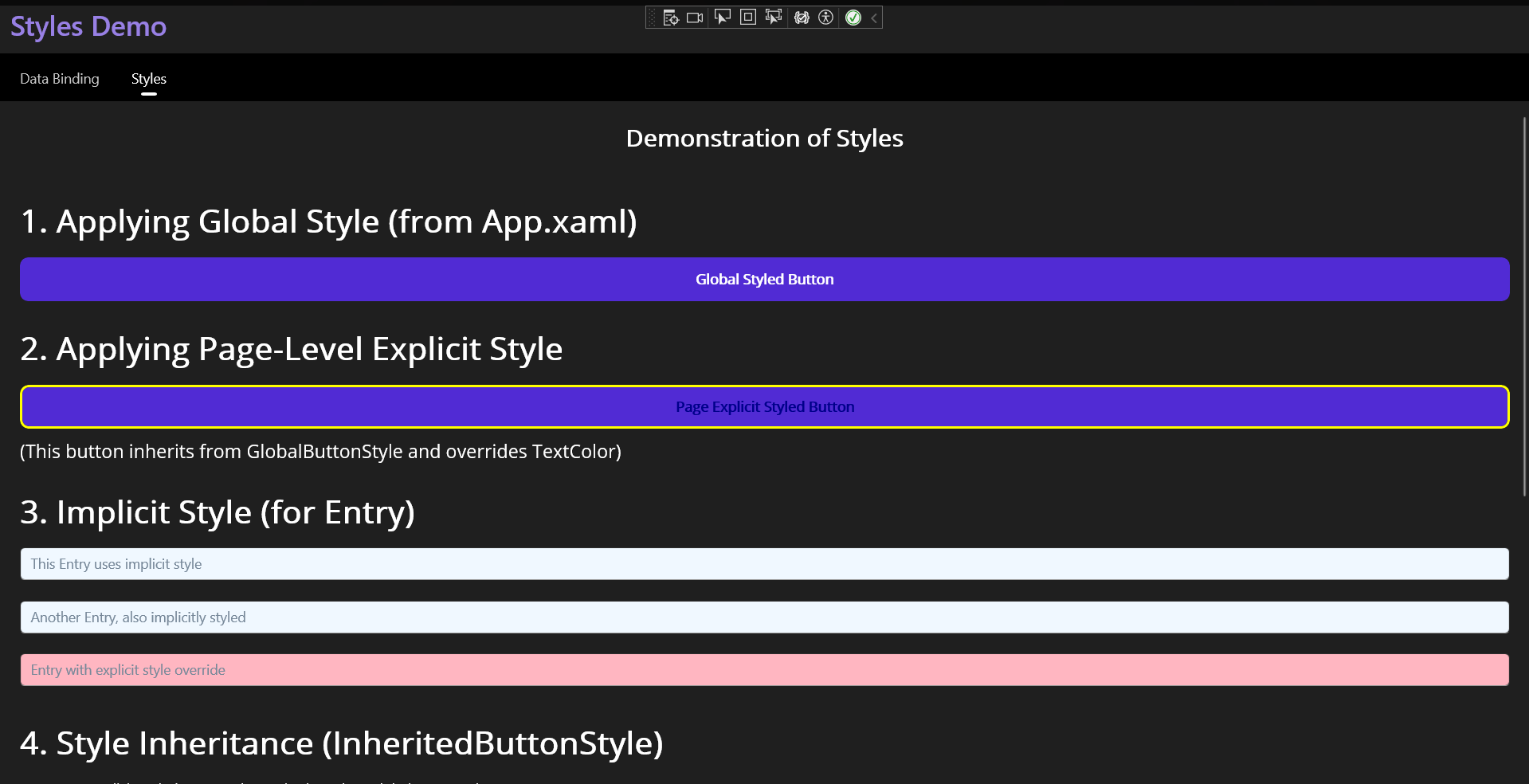
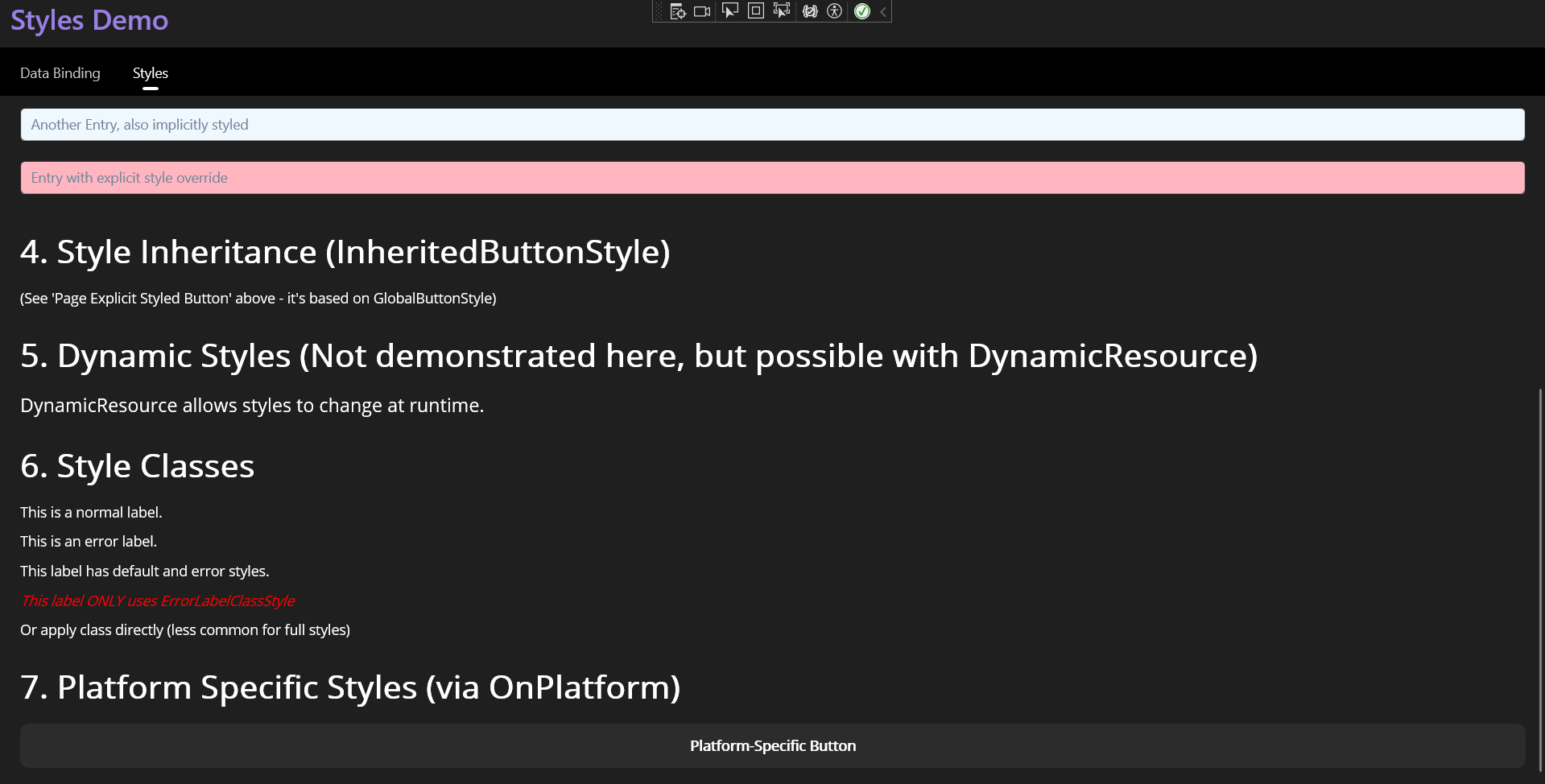
*Пояснение:* На странице определены явные стили (PageHeaderLabelStyle, InheritedButtonStyle), неявный стиль для Entry и стиль ErrorLabelClassStyle, который применяется через атрибут StyleClass. Также демонстрируется платформо-зависимое изменение BackgroundColor для кнопки.

**3. Заключение**

Использование стилей в .NET MAUI является мощным средством для управления внешним видом приложения. Возможность определять глобальные и локальные, явные и неявные стили, наследовать их и применять классы стилей, а также адаптировать под конкретные платформы, предоставляет разработчикам гибкость и контроль над UI. Это не только улучшает визуальную составляющую настольных и мобильных приложений, но и способствует лучшей организации кода и упрощает его дальнейшее сопровождение.

**4. Скриншоты и примеры**

Стили были продемонстрированы на странице StylesDemoPage.xaml, запущенной на платформе Windows.

 *"Рис. 6.1 и 6.2. Демонстрация применения различных типов стилей на странице StylesDemoPage."*

**Процесс выполнения примеров:**

1. Создана страница StylesDemoPage.xaml.
2. В App.xaml добавлен глобальный стиль GlobalButtonStyle.
3. На странице StylesDemoPage.xaml в секции ресурсов определены явные, неявные, наследуемые стили и стиль для использования с StyleClass.
4. Различные элементы управления (Label, Button, Entry) на странице были стилизованы с использованием определенных стилей.
5. Продемонстрировано применение платформо-зависимого стиля для кнопки.
6. Страница добавлена в навигацию AppShell.xaml и протестирована.  
   Все стили применялись корректно, демонстрируя ожидаемое изменение внешнего вида элементов управления. Неявные стили автоматически применялись к Entry, а явные и наследуемые стили корректно изменяли вид кнопок и меток.

**5. Документация**

* **Ссылка на GitHub с исходным кодом:** https://github.com/RaniZee/.NET-MAUI-Showcase-Cross-Platform-Applications
* **Описание выполненных шагов:**
  1. Изучены материалы по теме "Стили" (глава 6) с ресурса [metanit.com/sharp/maui/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fmetanit.com%2Fsharp%2Fmaui%2F).
  2. Создана страница StylesDemoPage для демонстрации стилей.
  3. Реализован пример глобального стиля в App.xaml.
  4. На странице StylesDemoPage продемонстрированы явные, неявные, наследуемые стили, применение класса стиля (StyleClass) и платформо-зависимые стили (OnPlatform).
  5. Проведено тестирование применения стилей на платформе Windows.
  6. Подготовлены скриншоты.