Windows Presentation Foundation (WPF) is a graphical user interface (GUI) framework used to build rich, modern desktop applications for Windows in the .NET ecosystem. It is part of the .NET Framework and also supported in .NET Core and later versions like .NET 5, 6, and 7.

**Key Features:**

1. **Vector-Based Rendering**: WPF uses DirectX for rendering, enabling smooth, high-quality visuals that scale well for different screen sizes and resolutions. This is a key advantage over the pixel-based rendering of older technologies like WinForms.
2. **Separation of UI and Logic (MVVM)**: WPF encourages a clean separation between the UI (View) and the application logic (ViewModel) through the Model-View-ViewModel (MVVM) design pattern. This makes WPF apps more maintainable and testable.
3. **XAML (Extensible Application Markup Language)**: WPF interfaces are typically defined using XAML, a declarative XML-based language. XAML makes it easy to create complex UIs by defining layouts, styles, animations, and more. You can also use code-behind (C#) for more dynamic operations.
4. **Rich UI and Controls**: WPF includes a vast range of built-in controls like buttons, text boxes, grids, and sliders. It also supports more complex controls like trees, menus, and tabs. Developers can easily extend or customize these controls.
5. **Data Binding**: WPF has powerful data binding capabilities, allowing the UI elements to be connected directly to data sources. It supports one-way, two-way, and one-time bindings, making it easy to keep the UI in sync with data changes.
6. **Styles, Templates, and Themes**: WPF allows extensive customization of the look and feel of applications. You can define styles and templates that control the appearance and behavior of controls, creating highly polished and consistent UIs.
7. **Multimedia and Animations**: WPF makes it easy to integrate multimedia (like video and audio) and create animations. For instance, you can animate controls or elements with built-in support for transitions and effects.
8. **Resolution Independence**: Since it uses vector-based graphics, WPF UIs scale better across different display resolutions and DPI settings, making it suitable for high-DPI screens.

**Use Cases:**

* Applications that require rich user interfaces with complex layouts, animations, and media.
* Enterprise-level desktop applications with a modern look and feel.
* Data-driven applications that benefit from the MVVM architecture and WPF’s robust data binding capabilities.

**Comparison to WinForms:**

* **WPF** is more modern, scalable, and suited for complex, visually-rich applications.
* **WinForms** is simpler and more appropriate for rapid, lightweight desktop application development, but lacks the visual and architectural flexibility of WPF.

Though WPF is primarily for Windows desktop applications, it is still widely used for building business applications, internal tools, and other software where a rich user interface is required.