Xamarin Development Cheat Sheet

A quick reference guide for Xamarin developers, covering essential concepts, components, and APIs for building cross-platform mobile applications with C#.

Xamarin.Forms Basics

Layouts

StackLayout	Arranges child views in a single line, either horizontally or vertically.
Grid	Arranges child views in rows and columns.
AbsoluteLayout	Positions child views using absolute coordinates.
RelativeLayout	Positions child views relative to each other or the parent layout.
FlexLayout	Arranges child views using flexible box layout model, similar to CSS Flexbox.
ContentView	Presents a single piece of content. Used for creating custom, reusable controls.

Common Controls

Label	Displays text.
Button	Performs an action when clicked.
Entry	Allows the user to enter single-line text.
Editor	Allows the user to enter multi-line text.
Image	Displays an image.
ListView	Displays a collection of data in a scrollable list.

Data Binding

Data binding allows you to synchronize data between a UI control and a data source (e.g., a property in your view model).

Example:

```
<Label Text="{Binding Name}" />
```

Use `INotifyPropertyChanged` interface in your view model to notify the UI when a property value changes.

```
public class MyViewModel :
INotifyPropertyChanged
{
    public event
PropertyChangedEventHandler
PropertyChanged;
    private string _name;
    public string Name
    {
        get { return _name; }
        set
        {
            _name = value;

PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(nameof(Name)));
        }
    }
}
```

Platform-Specific Code

DependencyService

The `DependencyService` allows you to access platform-specific functionality from your shared Xamarin.Forms code.

1. Define an interface in your shared code:

```
public interface IPlatformService
{
    string GetPlatformName();
}
```

2. Implement the interface in each platform project:

```
// iOS
[assembly:
Xamarin.Forms.Dependency(typeof(iOSPl
atformService))]
public class iOSPlatformService :
IPlatformService
{
    public string GetPlatformName()
=> "iOS";
}

// Android
[assembly:
Xamarin.Forms.Dependency(typeof(Droid
PlatformService))]
public class DroidPlatformService :
IPlatformService
{
    public string GetPlatformName()
=> "Android";
}
```

3. Consume the service in your shared code:

```
string platformName =
DependencyService.Get<IPlatformServic
e>().GetPlatformName();
```

Platform-Specific Views

You can create custom views for each platform to leverage platform-specific features and APIs.

1. Create a custom renderer:

```
// Custom renderer for iOS
public class CustomEntryRenderer :
EntryRenderer
{
    protected override void
OnElementChanged(ElementChangedEventArgs
<Entry> e)
    {
        base.OnElementChanged(e);

        if (Control != null)
        {
            // Customize the native
control
            Control.BorderStyle =
UITextBorderStyle.Line;
        }
    }
}
```

2. Register the renderer:

```
[assembly:
ExportRenderer(typeof(CustomEntry),
typeof(CustomEntryRenderer))]
namespace MyApp.iOS
{
    ...
}
```

Conditional Compilation

You can use conditional compilation directives to write platform-specific code within your shared codebase.

```
#if __IOS__
    // iOS specific code
    Console.WriteLine("Running on iOS");
#elif __ANDROID__
    // Android specific code
    Console.WriteLine("Running on
Android");
#endif
```

Navigation

Page Navigation

NavigationPage	Provides navigation and history management for a stack of pages.
PushAsync()	Pushes a new page onto the navigation stack. `await Navigation.PushAsync(new MyPage());`
PopAsync()	Removes the top page from the navigation stack. `await Navigation.PopAsync();`
PopToRootAsync()	Pops all pages off the navigation stack except the root page. `await Navigation.PopToRootAsync();`

Modal Navigation

PushModalAsync()	Presents a page modally. `await Navigation.PushModalAsync(new MyModalPage());`
PopModalAsync()	Dismisses the current modal page. `await Navigation.PopModalAsync();`

TabbedPage & MasterDetailPage

TabbedPage: Represents a multi-page interface where each child page is accessed through a tab.

MasterDetailPage: Manages two panes of information: a master pane that presents data at a high level, and a detail pane that displays details about information in the master pane.

Essentials

Xamarin.Essentials

Xamarin.Essentials provides APIs for accessing native device features in a cross-platform manner. Install the NuGet package to use it.

```
// Example: Get device information
string model = DeviceInfo.Model;
string platform =
DeviceInfo.Platform.ToString();
string version =
DeviceInfo.VersionString;
```

Common APIs

Connectivity	Checks network connectivity status.
Geolocation	Retrieves the device's current location.
Preferences	Stores and retrieves app preferences.
SecureStorage	Securely stores sensitive data.
Share	Shares text and files with other apps.
VersionTracking	Tracks app version information.

Example Usage

```
// Example: Check connectivity
var current =
Connectivity.NetworkAccess;

if (current == NetworkAccess.Internet)
{
    Console.WriteLine("Internet
access");
}

// Example: Get last known location
var location = await
Geolocation.GetLastKnownLocationAsync();

if (location != null)
{
    Console.WriteLine($"Latitude:
{location.Latitude}, Longitude:
{location.Longitude}");
}

//Example: Storing preferences
Preferences.Set("my_key", "My Value");
var myValue = Preferences.Get("my_key",
"default_value");
```