# 5 REASONS

## Why C# is the Clear Favorite

for Enterprise Client Development

As enterprise mobility gives rise to new business models, choosing the right programming language has become extremely important. Instead of focusing on quick wins, companies should consider the enterprise client platform as part of their long term vision and a point of differentiation.

There are two major choices to consider while selecting an enterprise client platform. The first is leveraging HTML5 and its ability to support major platforms, vendors and devices. These include the Mobile Enterprise Application Platform (MEAP) suite of products that enable. HTML5 based application development.The second option is to use C# which is a part of Microsoft's .NET framework and supports object-oriented programming (OOP). It is also possible to extend C# with the Xamarin platform to develop cross platform applications from a single code.

While each of these options have their own advantages, let's take a look at the top 5 reasons why C# may be a better option for companies in the long run.

**Microsoft Windows is here to stay:** Whether businesses like it or not, Windows powered desktops, laptops, tablets, and devices will continue to be a part of enterprises for a long time to come. While iOS and Android powered devices are also likely to make an entry, a robust enterprise client strategy has to evolve around the Microsoft Windows platform. Moreover, Windows 10 and its promise of universal apps also tilts the balance in favor of C#. In fact, using C# and XAML will be one of the most preferred ways to develop Windows 10 applications. With the support of Xamarin, this means that C# will be the top choice for companies looking to build a wide breadth of applications across Windows 10, Android, and iOS.

**User experience matters:** As companies increasingly leverage enterprise mobility, they will look towards offering richer user experiences to differentiate themselves. This places priority on natively compiled applications that take advantage of every feature and gesture of the device. In this regard, developing native applications using C# offers a clear advantage to companies.

**Native applications require an object-oriented approach:** There is a vast difference in the native app structure and life-cycle of each device operating system. For example an iOS app has UIApplication, AppDelegate, UIViewController, and UIWindows objects as part of its structure.

In addition, it has specific events and different execution states. In contrast, Android applications use an activity object and the associated events and application states. Similarly a Windows 10 application has a different object model and events. However, in order for a native application to be truly successful, the same object model, life-cycle, and event handling need to be inherited. Such an objected-oriented approach is possible only with C# and not HTML5.

**Developer productivity depends on reuse:** The variations in object model, life-cycle events, and application states across operating systems, can make it challenging to deliver consistent user experience across devices. However native app development using C# and Xamarin will help the companies write highly reusable codes across devices. Inheritance, polymorphism, and encapsulation OOP concepts are already and integral part of C#. When used in conjunction with design patterns such as MVVM, C# can be used to write highly reusable, native and cross-platform codes. C# is also a strong server side programming language for developing REST based web service end points. This means that developers can improve productivity by using a single code base for end-to-end app development.

**Adaptability is the key to success:** The previous wave of web applications started in early 2000, with most of them sustaining their existence for more than 10 years by managing multiple changes to the design patterns and frameworks. For instance, Java EE web development started with core SERVLETS, JSPs and then evolved to STRUTS and JSF. Similarly the back end code transitioned from DAO and EJB to Entity Frameworks. This means that the base programming platform has to be adaptable to new design thoughts and frameworks that may arise in the future. The current wave of enterprise client development targeting universal devices is expected to last till 2020. C# is therefore a better choice for companies looking to gain long-term success.

*Over the last 15 years Java and C# have been competing to win a greater share of the enterprise space. Both programming languages are equally strong in their developer support, OOP principles and productivity needs. However, compared to C#, Java does not have a universal client strategy. Hence C# seems to be the natural choice for future enterprise client development. Moreover, since .NETCore is now open source, even enterprises that were sceptical about using a commercial platform are now considering C# to mobilize their enterprise applications.*