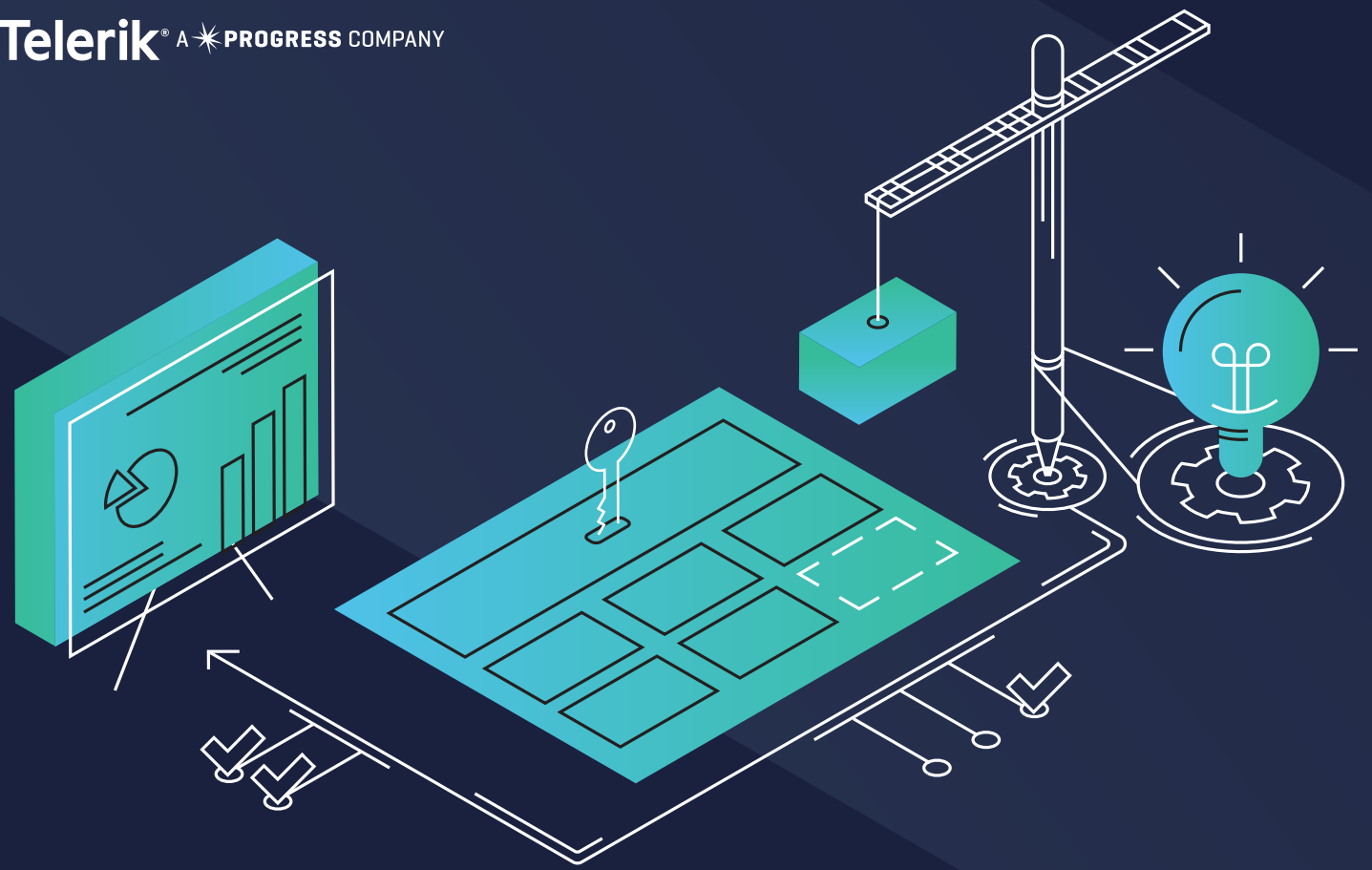
Telerik® A PROGRESS COMPANY



THE 2016 .NET COMMUNITY REPORT

Developer views on key technologies and innovations across web, desktop and mobile

Table of Contents

This report provides insight into a survey we conducted through the online properties of Telerik, a Progress company, including telerik.com and developer.telerik.com. The purpose of this survey was to get a sense for what .NET developers were thinking about on a range of topics such as desktop, mobile and web development. Survey responses capture the pulse of the .NET developer community going into 2016.

Conducted in English and publicly available for six weeks in January and February 2016, the survey reached a .NET developer audience of 1,862 developers; 1,017 responded to all questions.

Survey Demographics

.NET developers

How many years of development experience do you have?

Less than 1 year	4%
1-2 years	8%
3-5 years	15%
6-10 years	21%
11-20 years	27%
More than 20 years	25%

What's your highest educational degree in the field of software development?

College degree	48%
Master's or PhD	27%
I'm self-taught (no official degree)	20%
On-the-job training	5%

Where do you work?

Office	58%
Remote	9%
Mixed	33%

What is your role?

Developer	49%
Architect	14%
Management role	22%
Other	15%

Highlights

1. 30% of respondents are targeting Windows desktop, 45% desktop web and 49% responsive web

2. 42% of respondents building desktop applications use Windows Forms, 46% WPF and 8% UWP

3. 70% of respondents would use Angular if they were building web applications with JavaScript; 30% are divided among many alternatives

4. 61% of respondents are writing vanilla CSS, 15% Sass and 24% LESS

5. 43% of respondents use MSBuild, 37% do not use a task runner and 20% are split among alternatives

6. 52% of .NET developers believe Xamarin/C# is the best option to go cross-platform mobile; 22% would not use a cross-platform tool

7. Although mobile device and app platform choices are varied, 63% of respondents haven't used a Mac

Web Developers

Web developers have a lot of choices available in ASP.NET, including front-end core frameworks, user interface libraries, server-side frameworks, package managers and task runners. Respondents were asked their preferences on several key frameworks and tools for ASP.NET web development (client and server-side). The following results provide insight into how the .NET community views web development, and how each tool fits into the overall ecosystem.

Mobile Developers

For .NET developers, the proliferation of mobile apps and a vast new audience creates an interesting frontier. The encouraging trend, however, is tools maturity; no longer do you need to code to app platform specifics—cross-platform abstractions have come of age. While .NET developers continue to use a mixed bag of devices, there are clear trends regarding what development paradigms suit cross-platform needs. Developers appear to embrace the "mobile first, cloud first" mantra, while staying rooted with preferences and practicalities.

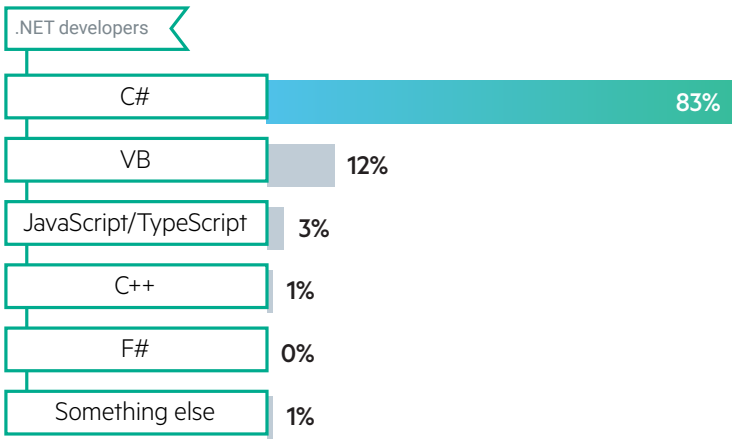
Desktop Developers

Desktop development continues to play a significant role amongst survey respondents, who broadly use technologies such as Windows Forms and WPF for building desktop applications. With the introduction of runtime environments such as UWP and Electron, developers have a wide range of languages and environments to target, to satisfy expanding requirements such as multiplatform support.

Technologies Developers Love or Hate

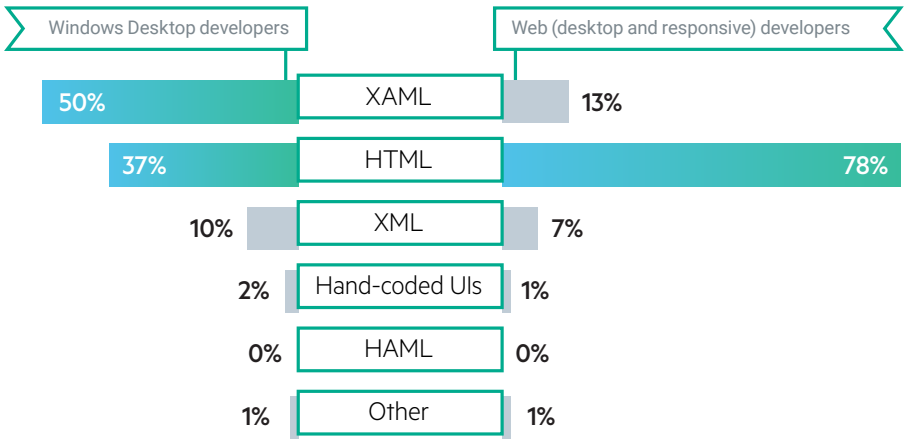
What's Your Primary Programming Language?

It's not surprising to see C# as the dominant language used by survey respondents—it's the de facto language among .NET developers, particularly those targeting the desktop through Windows Forms, WPF and UWP. Its popularity is widespread, as evidenced by survey results from Stack Overflow. We recommend developers continue to invest in their skills by learning what's coming with C# 6. Read Essential C# 6 Features You Need to Know!



Which Declarative Markup Language Do You Prefer?

Desktop applications built with HTML haven't been the norm. This result is largely skewed in its favour, because of its use in web development. Still, HTML is the preferred markup language of 37% of respondents who classify themselves as desktop developers, and 78% of respondents who classify themselves as web developers. Platforms such as UWP and Electron enable developers to build desktop applications with web technologies such as HTML, contributing to this result. XAML remains popular as a markup language because it's used predominately to build WPF and UWP applications.



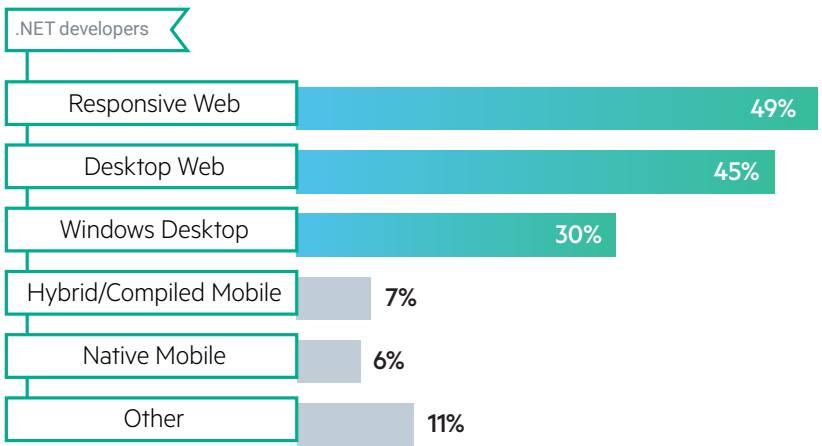
What Are You Primarily Developing Apps For?

Web Developers

Considering both desktop web and responsive web are web-based applications, as a whole, developers prefer the web as their platform of choice. Secondly, a little over half of those web developers are using responsive web techniques to create apps capable of desktop and mobile. Given Windows desktop, desktop web and responsive web combined make up most of the results (98% of total responses), only a minority of developers surveyed (approximately 2%) are building applications targeting mobile exclusively.

Desktop Developers

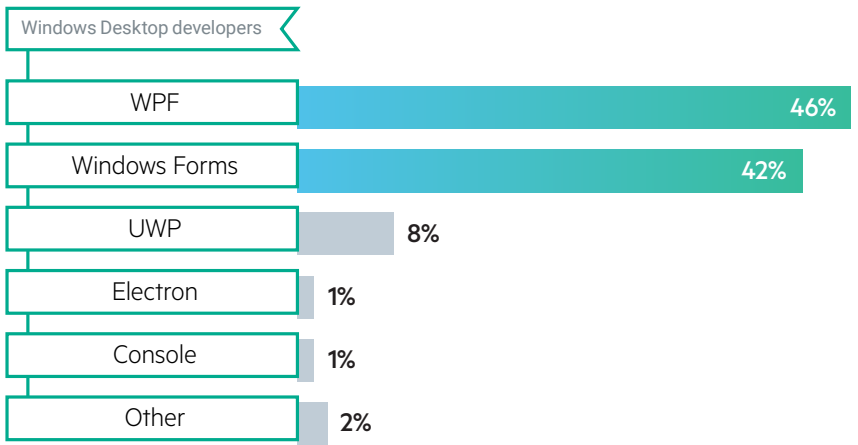
Thirty percent of all respondents are developing Windows-based desktop apps. Many developers are building or maintaining applications that leverage Windows Forms (42%), WPF (46%) or UWP (8%) as the supporting framework. Desktop development as a primary target isn't nearly as popular as web development; however, it currently outranks development for mobile devices. These results are similar to survey figures obtained from Stack Overflow.



What Technology Would You Choose if Building for Windows Desktop?

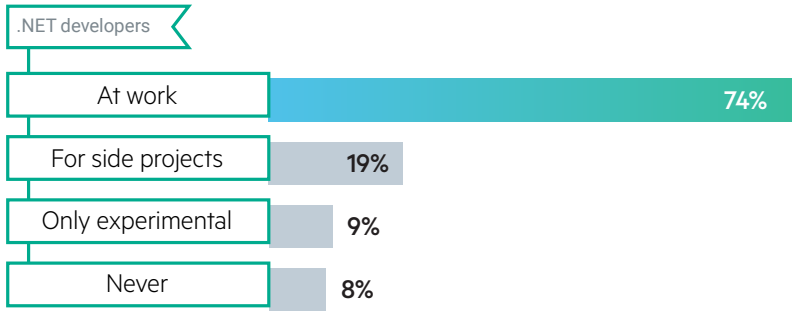
A significant number of respondents would build applications based on Windows Forms if given the choice for targeting Windows. This is somewhat surprising when you consider Windows Forms is under maintenance mode, with no new features being added. For larger organizations, which may have longer iteration cycles or legacy OS requirements, Windows Forms remains a viable platform for building desktop applications.

Many desktop developers build applications using WPF—it's a mature platform with broad runtime support across Windows. As WPF employs XAML to define the user interface, WPF developers can apply their skills and experience toward building UWP-based applications, as well. This provides a nice onramp for organizations wanting to transition to UWP, which is part of Windows 10 and Windows 10 Mobile.



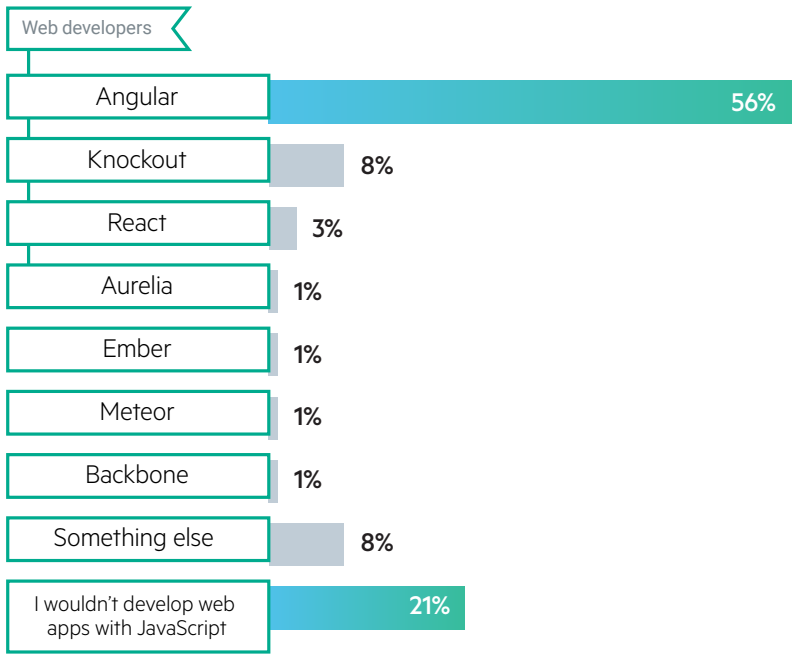
Do You Use JavaScript...

It's hard to imagine the web without JavaScript. JavaScript is becoming the ubiquitous programming language; so the results here aren't surprising. Because ASP.NET MVC doesn't abstract the use of JavaScript for client-side development like ASP.NET web forms, model-view-controller (MVC) has likely helped .NET developers become more comfortable with JavaScript. Another major contributor is the abundance of JavaScript libraries and frameworks in software development, .NET or otherwise.



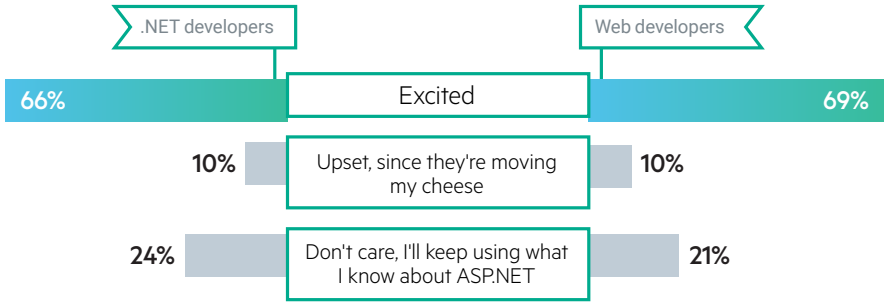
Your Chosen JavaScript Core Framework for Web Development Would Be...

Angular, an open-source web application framework mainly maintained by Google, is the popular core framework among .NET developers. This may be due to Microsoft's endorsement and commitment to its future. The Angular framework for client-side (MVC) and model–view–viewmodel (MVVM) architectures, coupled with dependency injection (DI), promote a separation of concerns and testability important to .NET developers. It's also worth mentioning the response, "I wouldn't develop web apps with JavaScript," appears to indicate a preference among developers to use the server for an application's logic and view rendering.



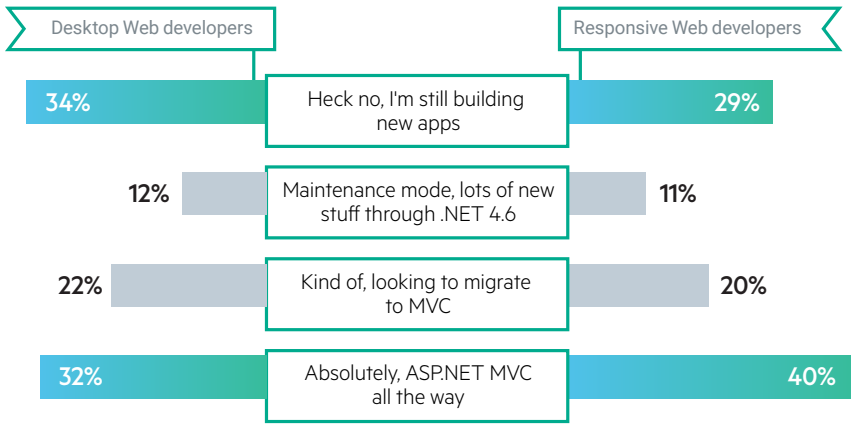
All of the Changes Coming in ASP.NET Make You Feel...

Excitement is building around ASP.NET because of a massive shift in Microsoft's approach. Microsoft is gearing ASP.NET to be the application development platform that can be written anywhere (Mac, Linux, or Windows) and run anywhere. This includes not only ASP.NET libraries but development tools, compilers, SQL Server and Visual Studio Code—a lightweight text editor with lots of promise.



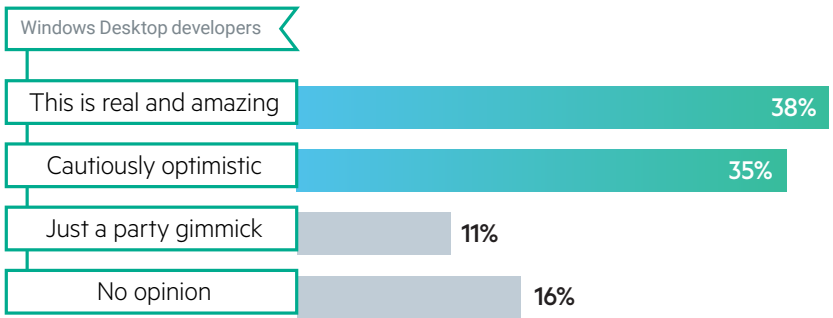
"ASP.NET WebForms is Dead"

While ASP.NET MVC is preferred, traditional ASP.NET Web Forms is still popular. For developers targeting desktop web, 32% prefer ASP.NET MVC. This number increases to 40% for developers targeting responsive web. For ASP.NET Web Forms, there are still quite a few developers working on new apps and some additional apps in maintenance mode. Until developers stop looking for a painless migration path and actually make the leap, ASP.NET Web Forms will stick around. The lack of necessity may contribute to the closeness of the results here, since ASP.NET Web Forms is completely capable, tested and a "known entity," .NET developers may be reluctant to abandon ASP.NET Web Forms completely. In addition, Microsoft keeps pushing ASP.NET Web Forms forward with updates such as Async Model Binding, HTTP2 and more.



How Do You Feel about Being Able to Build and Run .NET Apps Cross-Platform?

Developers have mixed opinions about targeting desktop applications supporting different platforms and form factors. UWP provides a platform for achieving cross-platform support in desktop applications. However, it only targets Windows-based platforms. Projects such as Mono offer cross-platform desktop application support through various GUI toolkits. Currently, the scope of .NET Core is limited to providing a framework for Windows Store, Windows Phone and ASP.NET applications. Microsoft's recent acquisition of Xamarin improves the possibility of cross-platform .NET desktop applications via Xamarin.Mac. Watching these platforms converge may prove interesting.

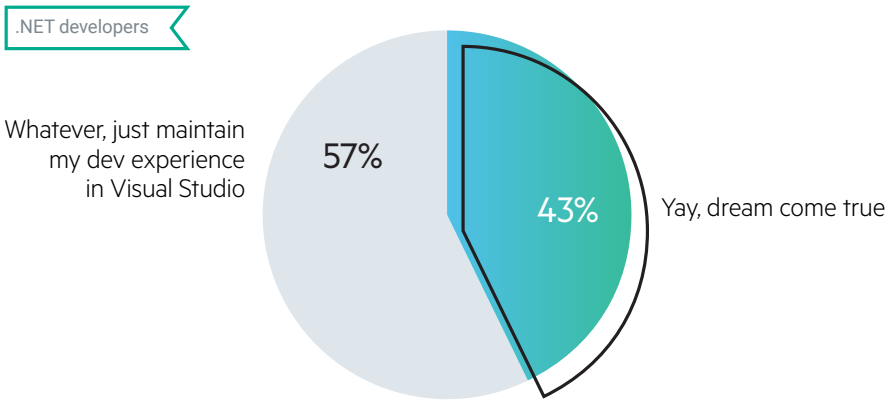


Universal Windows Platform (UWP)

The Universal Windows Platform (UWP) was first introduced in Windows 8 as Windows Runtime. At the core of UWP apps is the idea that users want their experiences to be mobile across all their devices, and they want to use whatever device is most convenient or productive for the task at hand. UWP apps have the potential to run on any Windows-powered device, from desktop and mobile devices to the IoT device family (wearables or household appliances). Each UWP app uses .NET Core as its runtime and .NET Native as its default toolchain.

.NET Core is Now Open Source and Cross-Platform. How Do You Feel about That?

While .NET Core is relatively new and remains under development, a significant portion of respondents appear enthusiastic about its potential for future .NET development. Microsoft has published information about .NET Core at dotnet.github.io.

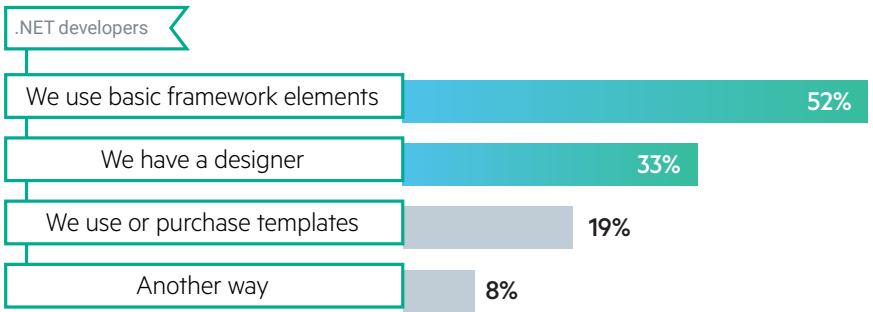


.NET Core

.NET Core is a cross-platform implementation of .NET primarily driven by ASP.NET 5 workloads, but also by the desire to have a modern runtime that's modular and provides features and libraries to be selected based on the application's needs. .NET Core consists of the CoreCLR runtime and the CoreFX framework libraries.

How Do You Handle the Creation of AppLayouts/Screens?

Of developers surveyed, 33% use a designer for handling application layouts, while the majority leverage a framework such as Bootstrap. Bootstrap's popularity may be due to it being the default HTML/CSS framework for ASP.NET MVC applications; however, Bootstrap is quite popular in general.

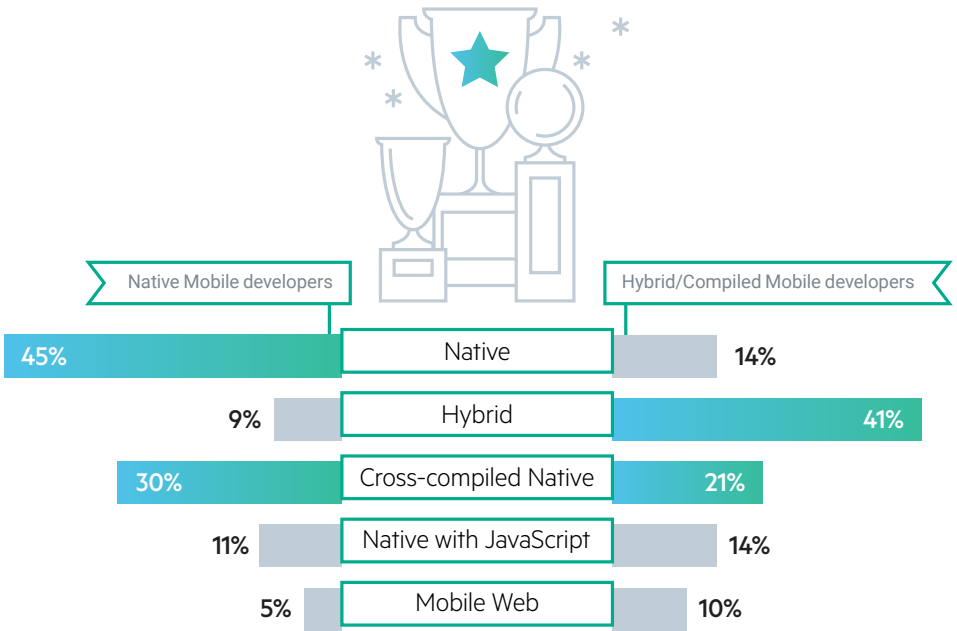


Devices and Experiences

What Would You Choose for Building Your Dream Mobile App?

If presented with the choice, respondents prefer to remain with development strategies that align to their role. For example, 45% of native mobile developers said they stay with native and 30% use cross-compiled native. On the other hand, of hybrid/compiled mobile developers, 41% said they stay with hybrid and 20% use cross-compiled native.

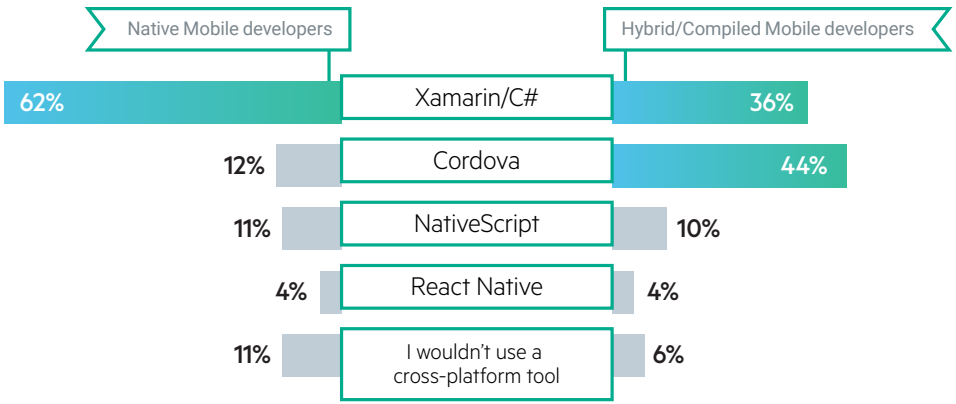
Native mobile apps are enticing, because UWP provides a way to target Windows 10-based devices with a single app. And, native iOS/Android development is no longer a far cry. Cross-compiled native apps have gained a lot of popularity among .NET developers, thanks to the success of Xamarin and the familiarity of C# language with the preferred IDE. Responsive mobile web apps still have their place among mobile technologies and may be suited for simple line-of-business apps. Another popular trend involves "JavaScript Native" apps—using web technologies to build native cross-platform apps, with frameworks such as NativeScript and React Native.



How Would You Go about Building a Cross-Platform Mobile App?

Overall, respondents favor cross-platform development with Xamarin. However, the data paints a different picture: Xamarin is the strongest preference (62%) amongst native mobile developers. Apache Cordova, on the other hand, is the top choice for hybrid/compiled mobile developers (44%).

Native mobile developers choosing Xamarin isn't surprising, given their love for C# and the convenience of using Visual Studio as an IDE. The recent Microsoft acquisition of Xamarin validates the story. However, Xamarin isn't the only cross-platform approach for .NET developers. Building hybrid mobile apps is still a legitimate strategy, thanks to now-polished tooling around Visual Studio Cordova integration and Telerik AppBuilder. If you like TypeScript and a XAML-like UI markup for mobile apps, NativeScript provides a refreshing alternative for cross-platform development, while building truly native mobile apps. (Note that survey results were collected prior to the Microsoft acquisition of Xamarin.)

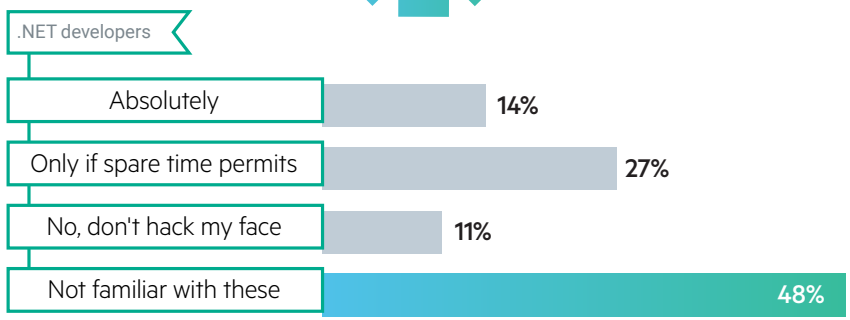


Xamarin

The Xamarin platform consists of a number of elements that enable you to develop apps for iOS and Android: C# language + Mono .NET framework + Compiler + IDE tools (Xamarin Studio IDE and the Xamarin plug-in for Visual Studio). In addition, because the underlying language is C# with the .NET framework, projects can be structured to share code for deploying to Windows Phone.

Will You Be Building Apps for HoloLens or Windows IoT Core?

The frontiers for running .NET code are fast expanding. From the holographic, mesmerizing digital world of the HoloLens to the uniqueness of Windows-based IoT devices, your code brings to life a wide variety of apps. Is such app development cool? You bet! But immediate return on investment is probably not high—and that may explain why many .NET developers are doing such cutting-edge development in their spare time or as a hobby. As futuristic devices become mainstream, cutting-edge development will likely also gain traction.



HoloLens

Microsoft HoloLens is a fully untethered, holographic computer that enables you to interact with high-definition holograms. It is enabled by Windows 10, the first platform to support holographic computing with APIs that enable gaze, gesture, voice and environmental understanding on an untethered device.

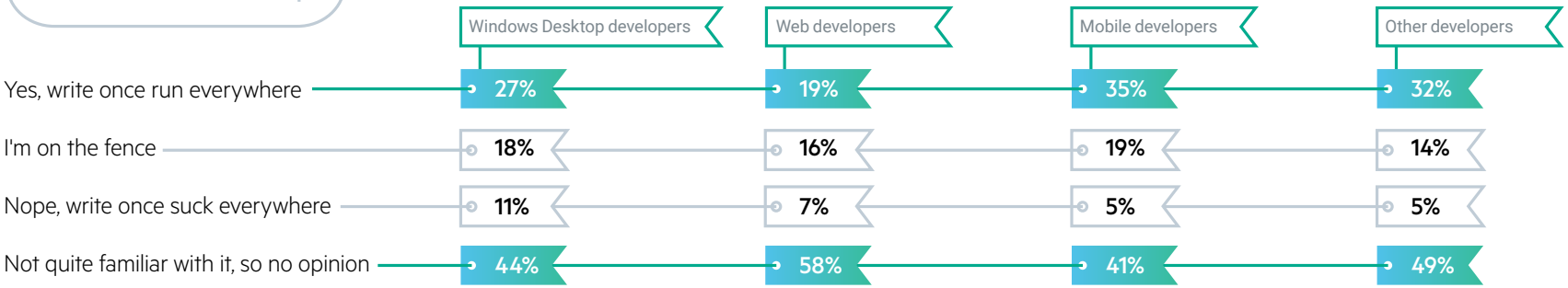
Do You Buy Into the Microsoft UWP Apps Paradigm?

Mobile Developers

With the Universal Windows Platform, Microsoft promises you can write an app once and run it on every Windows 10 device—phones, tablets, PCs and even an Xbox. Microsoft continues to push hard on UWP with Windows 10 adoption. However, reality suggests UWP is not catching up with developers as fast as Microsoft would hope. It may be a chicken-and-egg problem: app quality and Windows Store success will drive developer adoption of UWP.

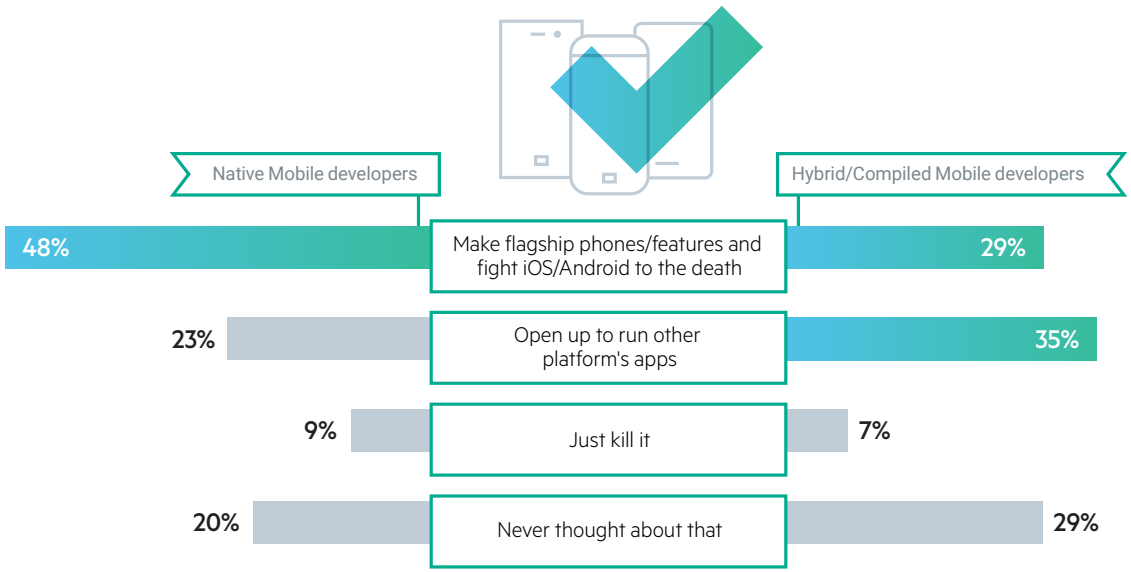
Desktop Developers

The majority of respondents stated they were not familiar with the UWP application paradigm. However, looking at the data by role, mobile developers were most favorable toward UWP development, with 35% stating they agreed with the paradigm. The second most favorable were Windows desktop developers (27%) followed by web developers (18%). These results bode well for Microsoft's strategy for Windows 10-based development, which spans a variety of devices. That stated, Windows 10 shipped in July 2015; a favourable consensus may take time to develop.



If You Were Running the Show, What Would You Do with Windows 10 Mobile?

No, Windows Phones aren't as rare as unicorns; but sightings of Windows–powered phones out in the wild are becoming elusive. Even the most hardcore fans acknowledge Microsoft's struggle in the mobile ecosystem space. Windows 10 Mobile may be the last hope or perceived as dead on arrival, based on who you talk to. A dramatic turnaround would require making up the app gap and some killer hardware, to compete with iPhone and Android. Results indicate some respondents wouldn't mind cross-pollination of apps from other platforms.



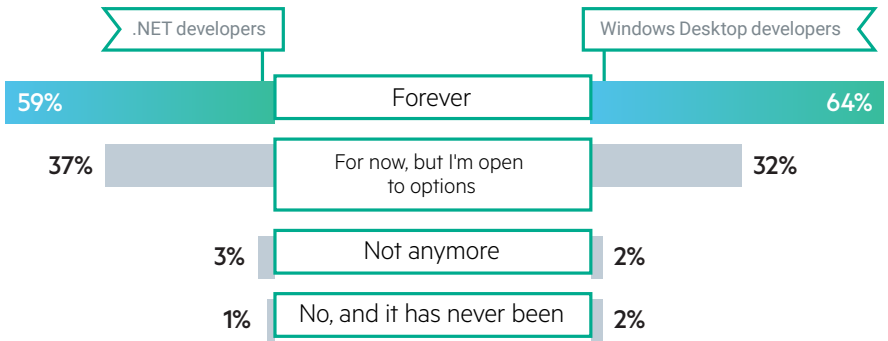
Tools of the Trade

Is Visual Studio Your Primary IDE?

Visual Studio is an excellent IDE, and the survey results reflect this. Its features benefit desktop developers in particular, because of its design-time experience for building apps with Windows Forms, WPF or UWP.

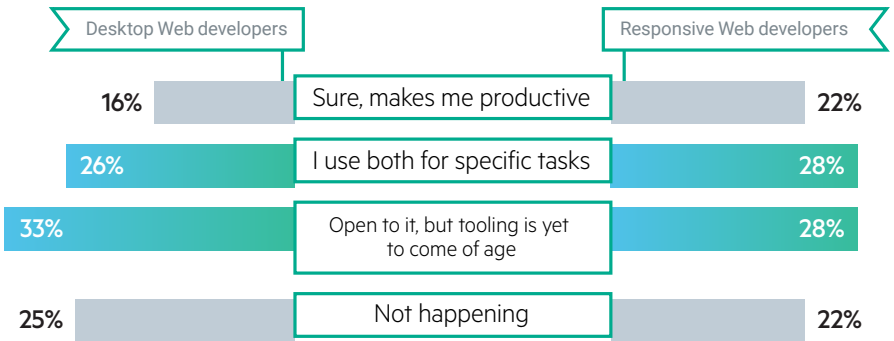
Web Developers

Web editing in Visual Studio has improved over the years. The HTML editor in Visual Studio is very robust with IntelliSense on CSS classes, auto-tag completion and productivity tools through Web Essentials. Visual Studio Code could become a favourable alternative for web in the near future.



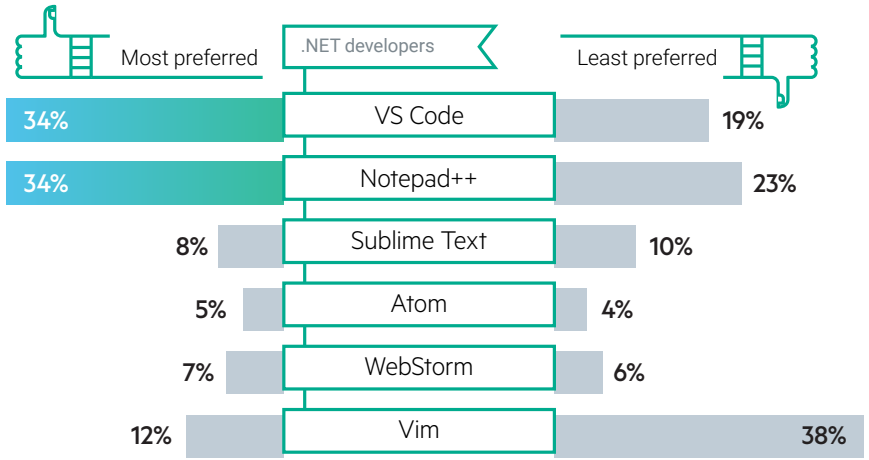
How Do You Feel about Using a Lightweight Code Editor Instead of a Full IDE?

Code editors such as Atom, Sublime Text and Visual Studio Code provide an alternative to IDEs like Visual Studio and Eclipse for building apps for the desktop, mobile and web. A sizeable segment of developers gravitate toward these code editors, because they are lightweight and fast. **Limitations including the lack of an interface designer tend to sway desktop developers toward IDEs such as Visual Studio.** That stated, code editors should still be considered as part of a developer's arsenal.



Lightweight Code Editors

Lightweight code editors provide a productive environment for situations in which developers make small changes or work with text files. According to respondents, winners in this category are Notepad++ and Visual Studio Code.

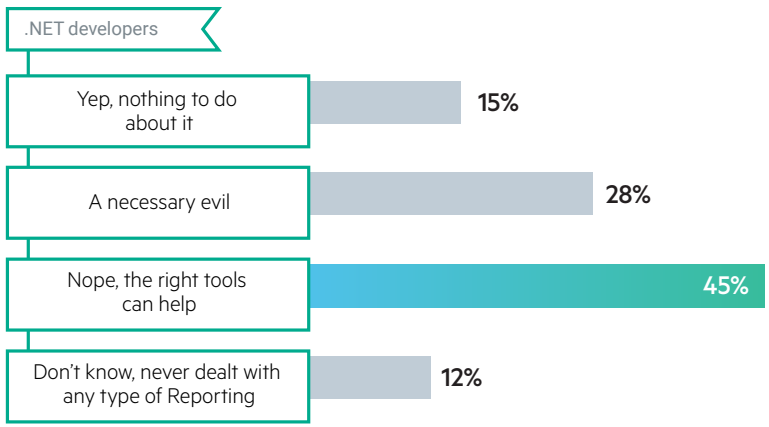


Visual Studio Code

Visual Studio Code is a free cross-platform editor that supports writing modern web and cloud applications. It provides a subset of features that have made its "older brother," Visual Studio, popular, such as Intellisense and strong debugging support. It helps developers get started with ASP.NET Core 1 development on Linux and OS X.

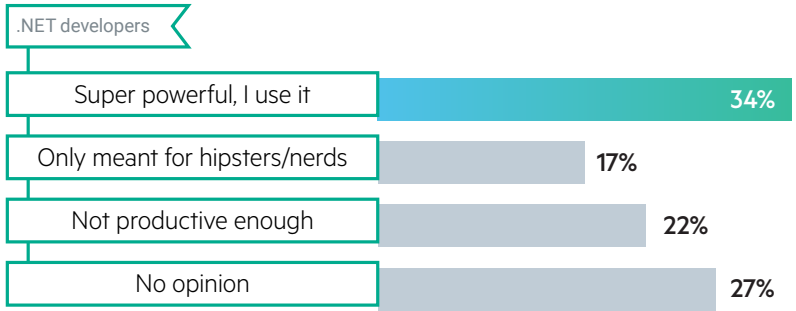
"Any Type of Reporting Sucks"

It seems as if everyone loves reports, except for the people tasked with creating them. A significant portion of respondents agreed that the right tools can help with this task. Many reporting solutions aren't very developer-friendly; often, just generating a report requires a new environment or language. A good reporting solution is one that fits within a developer's workflow and enables them to leverage their existing skills.



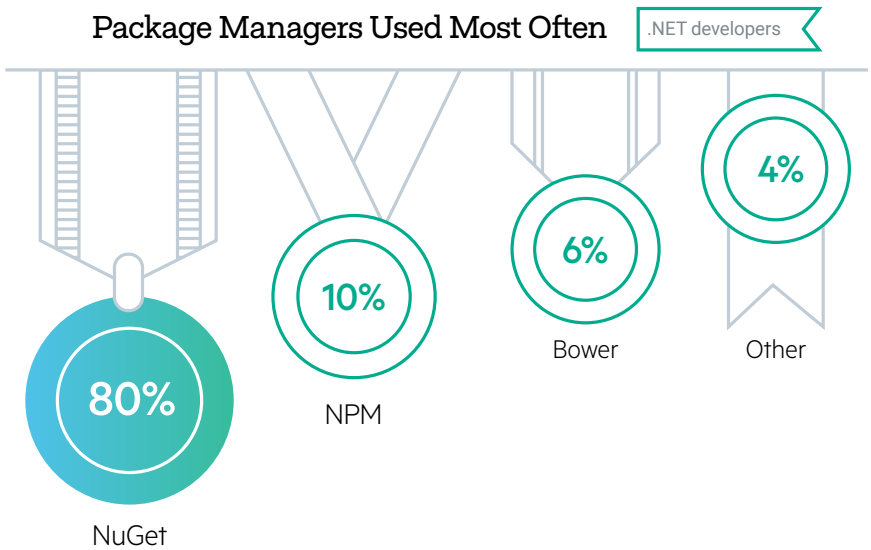
Command Line Tooling is...

The command line is a powerful utility for developers, because it's fast and can be scripted easily. However, it can require a steep learning curve for developers new to a platform. For a long time, IDEs such as Visual Studio have shielded developers from seeing the command line operate in the backend. This has changed over the past few years. Currently, many IDEs expose and/or leverage underlying command line utilities. For developers using lightweight editors such as Visual Studio Code, the command line is an extension of their workflow.



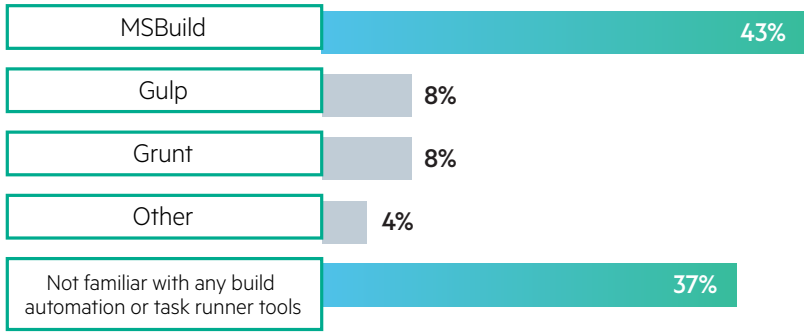
Package Managers

NuGet leads as the package manager of choice for .NET developers by a huge margin. Because of NuGet's close integration with Visual Studio, it's hard to imagine the results showing otherwise. The release of ASP.NET Core and its support through Visual Studio tooling, as well as its inclusion of Bower and NPM in project templates, may change adoption rates over the next few years.



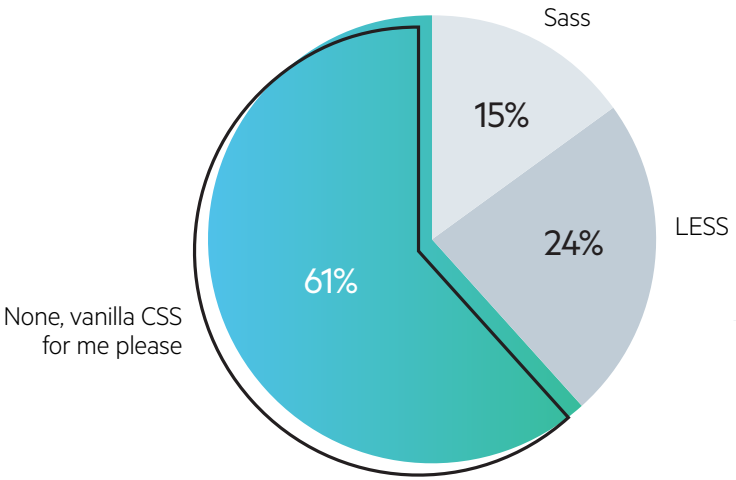
Task Runner/Build Automation

Many respondents were split between no task runner and MSBuild. Other choices including Gulp and Grunt are few and far between. While Gulp and Grunt are popular outside the .NET ecosystem, they have yet to gain market share with survey respondents. As with many other third-party tools covered in the survey, the tooling in Visual Studio is directly related to adoption. Again, ASP.NET Core may change developers' opinions over time.



CSS Preprocessors

CSS preprocessors aren't very popular among the .NET developers surveyed. Lack of adoption in this space could be due to the difficulty of adding tooling for pre-compilation. ASP.NET Core and its integration with Node Package Manager (NPM) and the Gulp task runner might influence future adoption. Of respondents who use preprocessors, LESS is slightly ahead; however, this may also change with the introduction of Sass in Bootstrap 4.



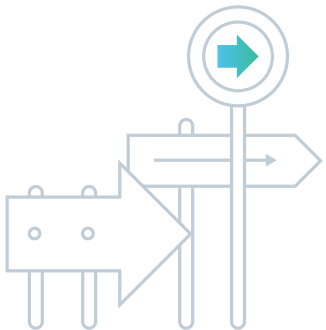
Your Thoughts on XAML

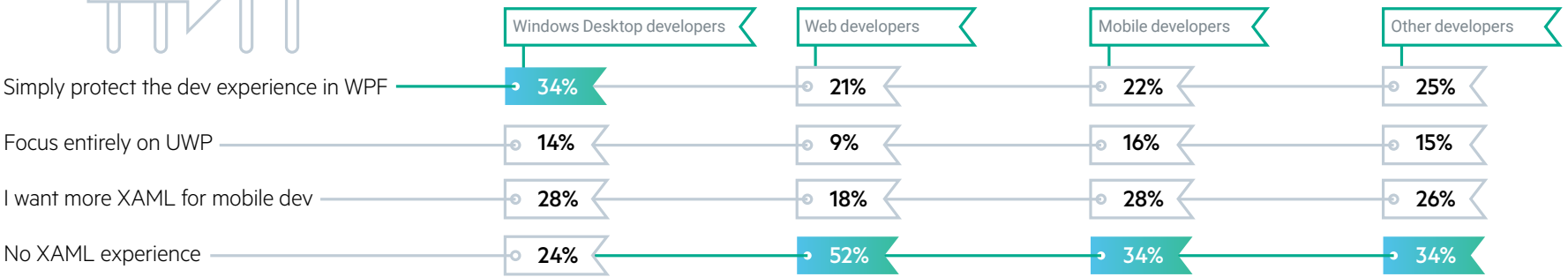
Mobile Developers

XAML as a markup language, began with humble roots in WPF and Silverlight. However, its strengths—the ease of describing complex nested UI, data binding and commanding patterns—were evident over time. In fact, XAML has grown up to be one of the beloved markup languages for .NET developers, and the rich ecosystem around XAML tooling justifies the need. As app platforms on top have evolved, XAML has seen improvements in tooling and language features. Many .NET developers would like the XAML experience to be best-in-class with WPF, but Microsoft is also pushing for similar experiences with UWP. With Microsoft's recent acquisition of Xamarin, expect even more from XAML—one unified markup language for cross-platform UI.

Desktop Developers

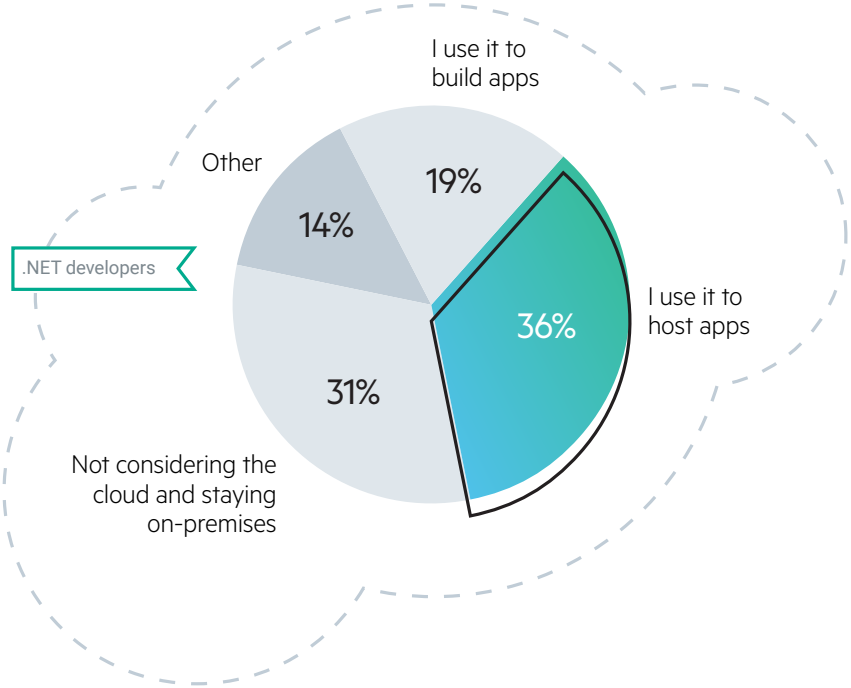
Generally, XAML is viewed as a suitable approach for building .NET apps, if and where it's supported. This bodes well for desktop developers who are invested in WPF and UWP. XAML is likely to remain supported for a long time. The alternative to XAML is to target HTML for the user interface, which is also supported when building UWP applications. Regardless of your preferred language and environment, the three primary ways to target UWP are XAML/C#, HTML/JavaScript or C++. Each approach has its benefits and challenges, which are addressed by Chris Anderson, who discussed these approaches at the Microsoft Build conference in 2013.





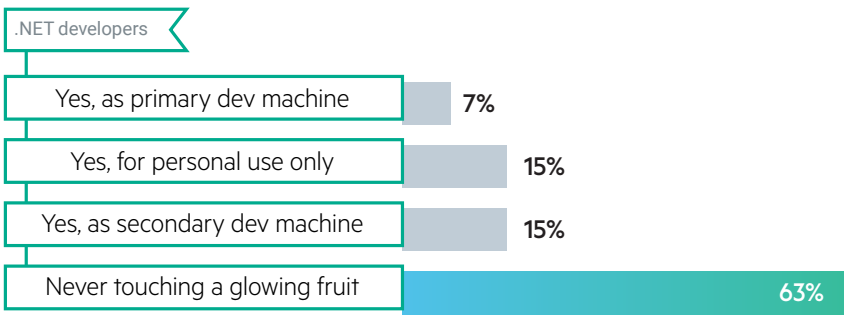
Your Take on the Cloud...

We are living the "mobile first, cloud first" era. Microsoft Azure and other cloud infrastructure providers are breaking down barriers, reducing costs and contributing to dev/test scalability. A significant portion of respondents appear to favor the PAAS model–build apps locally and host apps/data in the cloud. Of course, there are some valid security concerns, and on-premise deployments continue to be successful. Perhaps a hybrid model with a mixture of on-premise and cloud infrastructure is best. Developers have the right tools in hand today: services such as Data Connectors in Telerik Backend Services or Azure Hybrid Connections.



Do You Use a Mac?

Windows and Visual Studio have a rich ecosystem and vibrant community. That stated, change is in the air with a minority segment of respondents switching to the Mac. With Microsoft's open source friendliness with .NET and tools such as the cross-platform editor of Visual Studio Code, perhaps more .NET developers will be open to picking up a Mac. In the meantime, Microsoft champions the Windows-based hardware front with products such as the Surface Book and Surface Pro 4.



Do You Like Touch Capability on Your Main Development Machine?

Mobile Developers

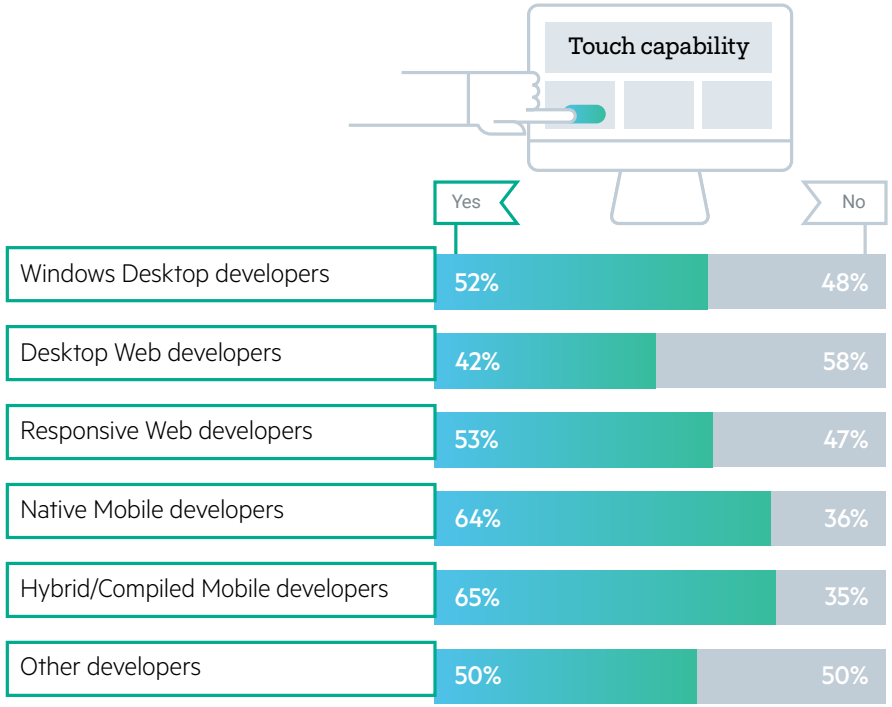
With Windows, touch is the new rage. You get beautiful convertible devices with touchscreens on high-DPI displays. But do you freak out any time someone touches your computer screen? No one likes smudges. When it comes to touch capability on the main development machine, .NET developers can disagree.

Desktop Developers

Support for touch and touch-gesture is important for desktop applications, as devices like the Microsoft Surface 4 feature rich built-in touch capabilities. Integrating touch into applications built with Windows Forms, WPF and/or UWP is facilitated through SDK libraries and frameworks from third parties such as Telerik, a Progress company.

Web Developers

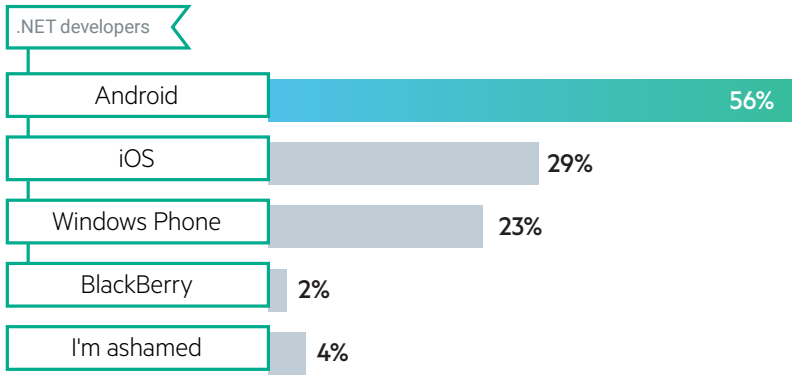
If you're developing responsive web applications, touch could be helpful for ad-hoc testing. However, using touch for testing on the desktop, even with good emulation, is no match for actual mobile hardware. Other than testing, there's no real advantage to using touch in a development scenario.



What Do You Carry in Your Pocket?

Maybe rumors of Windows Phone's slow demise are exaggerated; a good portion of .NET developers still carry it around proudly. Still, Android wins in overall market share, by far, followed by iPhone.

But remember: market fragmentation is why cross-platform mobile technologies are of such importance, enabling developers to target as big an audience as possible.

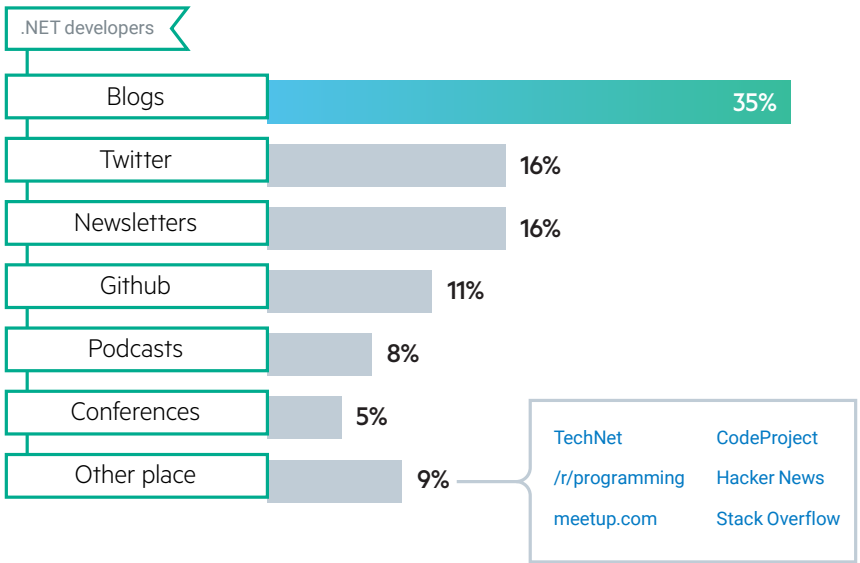


Passion for Developer Craft

Where Do You Hear About New/Exciting Stuff about Coding?

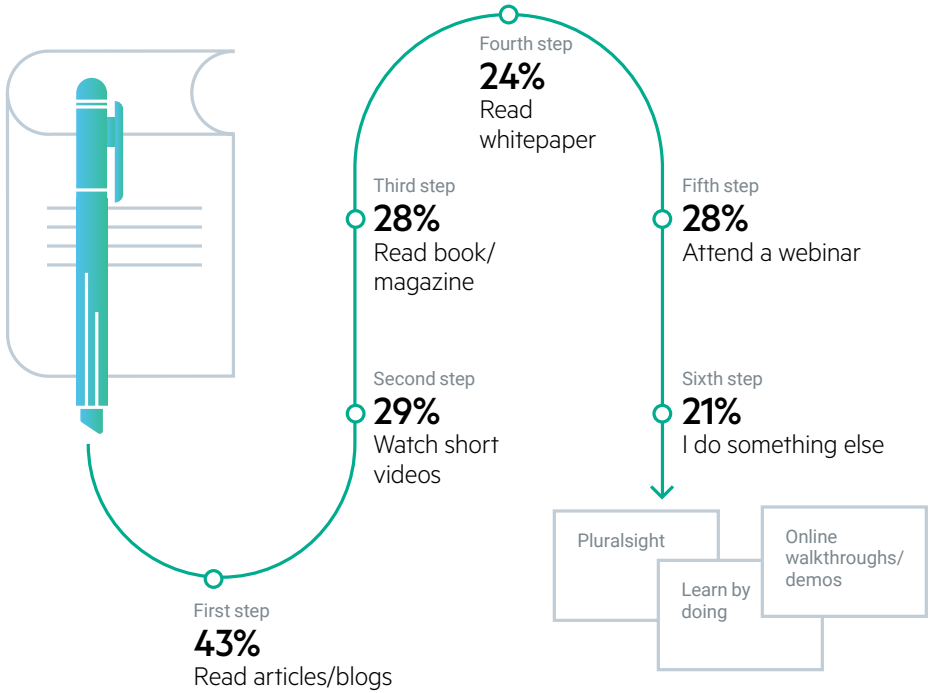
A significant number of respondents cited Reddit and Stack Overflow as online sources to hear about exciting coding news. Respondents also cited networks or friends as sources of such information.

Channel Used Most Often



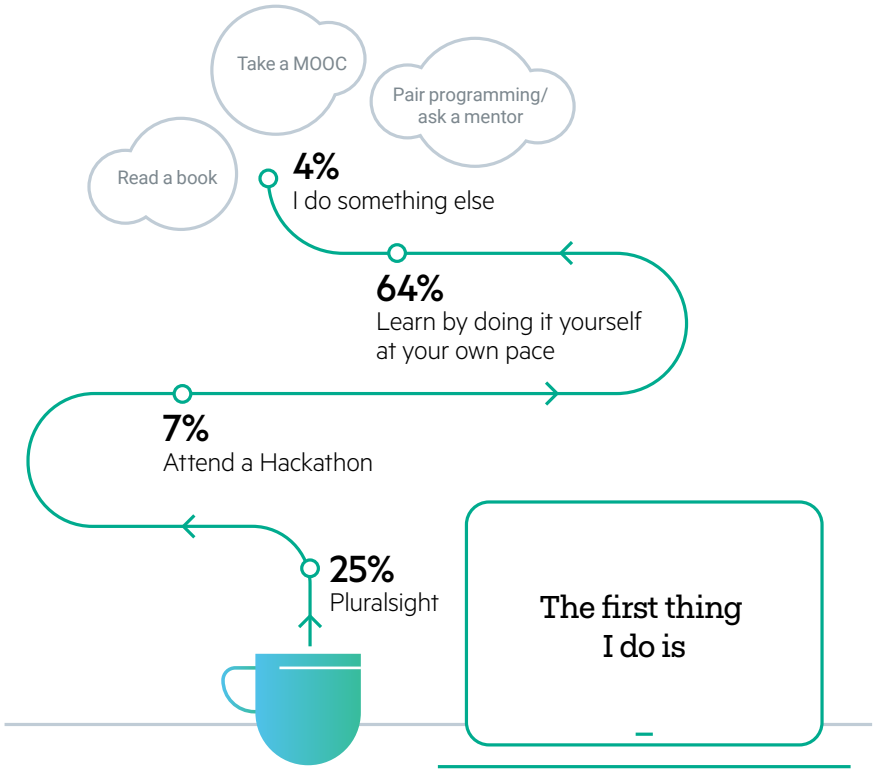
How Would You Go about Starting to Learn about a New Technology?

Many respondents cited Pluralsight as a top online resource to learn about new technologies. Others cited in-person courses and experimentation through prototyping as good places to start learning.



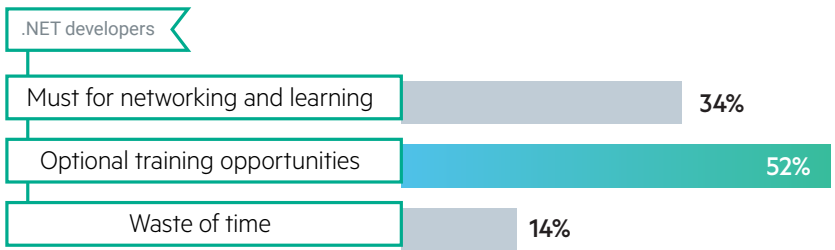
How Would You Deep-Dive to Learn Something New?

An important takeaway from the survey, self-teaching, very popular among developers. Developers have a tendency to "learn by doing," and resources that enable them to "get their hands dirty" are popular. At Telerik, a Progress company, we've used this approach on our products; a prime example is our NativeScript Getting Started Guide.



What Are Developer Conferences?

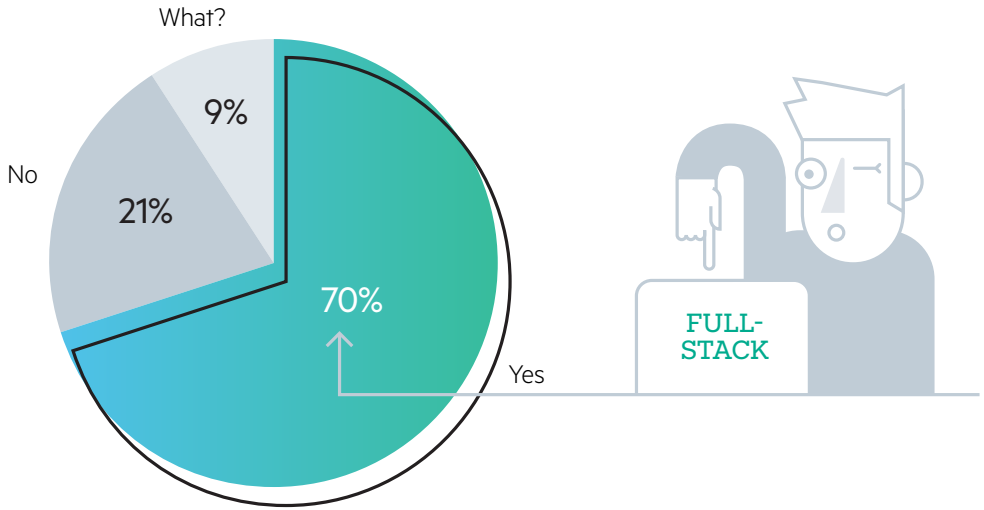
The majority of respondents favor developer conferences. Events such as Build provide developers with the opportunity to learn about the latest technologies from Microsoft. Community activities, including user group meetings, encourage networking.



Do You Consider Yourself a Full-Stack Developer?

Be it web, desktop or mobile development, most developers are exposed to the vast gamut of technologies. Many .NET developers are well-versed in technologies that span the entire application stack, from UI to the database. Even within a single stack, a developer might see several architectures. In effect, it is great to see most .NET developers starting to consider themselves full-stack and often polyglot developers.

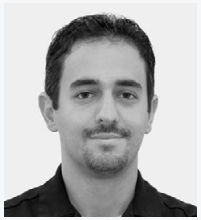
.NET developers



Summary

This year looks promising for .NET developers, as Microsoft continues to innovate across desktop, mobile and web platforms. UWP development is viewed favorably for building desktop applications. The recent acquisition of Xamarin appears to have reinvigorated mobile development through C#. And, there's momentum behind .NET Core and the next version of ASP.NET MVC. Overall, it's an exciting time to be a .NET developer.

About the Authors



Ed Charbeneau

Ed Charbeneau is a web enthusiast, speaker and writer. Ed enjoys geeking out to cool new tech, brainstorming about future technology and admiring great design.

John Bristowe

John Bristowe's passion lies in modern web standards and mobile development. A Canadian living in Australia, he's equally split between ice hockey and the rugby union.

Sam Basu

Sam Basu is a technologist, Apress/Pluralsight author, speaker, Microsoft MVP, believer in software craftsmanship and gadget-lover.

Ed, John and Sam are Developer Advocates for Telerik.

About Telerik

Telerik empowers its customers to create compelling app experiences across any screen. Our end-to-end platform uniquely combines industry-leading UI tools with cloud services to simplify the entire app development lifecycle. Telerik tools and services can be adopted individually or as a platform and seamlessly integrated with other popular developer solutions. More than 130,000 customers from 60,000 organizations in 94 countries depend on Telerik products, including more than 450 of the Fortune 500®, academic institutions, governments and non-profits. For additional information about Telerik, please visit telerik.com or follow @telerik on Twitter.

Rev 16/03 | 160323-0037

Telerik® A PROGRESS COMPANY