**Desktop Applications**

**Introduce the Topic**

* A desktop application is, according to PC Mag, “an application that runs stand alone in a desktop or laptop computer,” which is contrasted with a "web-based application, which requires the Web browser to run.”
* As the name suggests desktop applications run on desktop platforms whereas mobile applications that run in smartphones and tablets
  + <http://www.pcmag.com/encyclopedia/term/41158/desktop-application>

**History**

* August 12th 1981, when IBM introduced a PC hardware platform. IBM PCs used a text mode, command-line style operating system known as MS-DOS (which stands for Microsoft Disk Operating System), which eventually was replaced with the graphical Microsoft Windows OS in the 1990s. (http://www.seguetech.com/blog/2013/06/07/desktop-vs-web-applications-deeper-comparison)
* Then, the World Wide Web (WWW) took off in 1991 and the Mosaic web browser application was announced in 1993. These changes affected our world a great deal. (http://www.seguetech.com/blog/2013/06/07/desktop-vs-web-applications-deeper-comparison)
* Although early applications were developed to be run from mainframe computers and accessed via low-tech terminal devices, the increased power and availability of (relatively) powerful desktop computers ushered in an era of standalone desktop applications that were run locally on the PC. (http://www.seguetech.com/blog/2013/06/07/desktop-vs-web-applications-deeper-comparison)

**Desktop vs. Web Applications**

* “Software application development began with [desktop applications](javascript:void(0)), which could be used on standalone machines only.” (<http://www.streetdirectory.com/travel_guide/114448/programming/desktop_applications_vs_web_applications.html)>
* Web applications began replacing desktop applications for reasons of portability and better functions from usability point of view (http://www.streetdirectory.com/travel\_guide/114448/programming/desktop\_applications\_vs\_web\_applications.html)
* Web applications development “is usually made on client-server architecture and use a web-browser as the client interface.”( <http://www.streetdirectory.com/travel_guide/114448/programming/desktop_applications_vs_web_applications.html)>
* Desktop applications have traditionally been limited by the hardware on which they are run.
  + This hardware dependence, as well as the legacy of mainframe terminal applications, has typically limited the level of complexity in user interfaces for desktop applications.

|  |  |  |
| --- | --- | --- |
| Features | Desktop Applications | Web Application |
| Rapid Development | Designed from the beginning to be a quick and easy solution to building graphical user interfaces (GUIs), especially when using Windows Forms in Visual Studio for Windows | Never was designed for rapid development. Using MVC (model, view, control) style is typically seen as the “correct” way to create websites. The closest thing to “rapid development” in web application is ASP.NET WebForms |
| Security | Since the user keeps his or her data on their own computer systems (the program is running off the user’s desktop), this makes it harder for hackers to gain access to people’s data | Since all the user’s data is stored online, theoretically it would take just “one” hack to compromise “all” of the user(s) data. However, this is just a possibility and not a guaranteed situation |
| Available Controls | Desktop application developers have a whole buffet of user interactive controls to choose from. This goes for the out of the box controls (i.e. Visual Studio for Windows) as well as 3rd party controls | No real controls “out of the box” to speak of, except for the exception of HTML. While you can add controls via jQuery or something else, these are not as mature as the desktop equivalents |
| Flexibility | It is very easy to write desktop applications that take advantage of the user’s hardware | Web applications do not even compare to the flexibility of desktop applications. If you want to write a web application that basically interacts with the user’s hardware, you are doing it wrong. Just stick with a desktop application for your program |
| Portability | Desktop applications can be portable, but most are “not” portable and require manual installation from the user | Web applications have desktop apps beat here. Web applications are “very” portable and will work with just about any computer with a decent web browser |
| Maintenance | Desktop applications usually need to be manually updated to install updates | Web applications have desktop apps beat here again. Web applications are “very” maintainable. End users do not have to install “any” updates. All the updates are already taken care of by the web application administrator(s) who updates the web application on the server(s) that they are running on |

* (<http://blog.computerlagoon.com/2015/08/04/pros-and-cons-to-web-and-desktop-applications/>)

**Mac Tools/Framework**

* XCode
  + An integrated development environment (IDE) containing a suite of software development tools developed by Apple for developing software for OS X and iOS
  + (<https://en.wikipedia.org/wiki/Xcode>)
* Coca Framework
  + An Apple’s native object oriented application programming interface (API) for the OS X operating system
  + Consists of the Foundation Kit, Application Kit, and Core Data frameworks, as well as the libraries and frameworks included by those, such as the C standard library and the Objective C runtime
  + Are typically developed using the development tools provided by Apple using the Objective C or Swift language
  + (<https://en.wikipedia.org/wiki/Cocoa_(API>))

**Windows Tools/Framework**

* VisualStudio
  + An integrated development environment (IDE) from Microsoft
  + Used to develop computer programs for Microsoft Windows, as well as web sites, web applications and web services
  + Uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight
  + (<https://en.wikipedia.org/wiki/Microsoft_Visual_Studio>)
* .NET
  + .NET framework is a runtime execution environment that manages applications that target the .NET Framework
  + it consists of the common language runtime, which provides memory management and other system services and an extensive class library, which enables programmers to take advantages of robust, reliable code for all major areas of application development
  + Consists of two major components:
    - The common language runtime (CLR): the execution engine that handles running applications
    - .NET Framework Class Library: provides a library of tested, reusable code that developers can call from their own applications
  + Services that .NET Framework provides to running applications
    - Memory Management: the CLR are responsible for allocating and releasing memory and for handling object lifetimes
    - Common Type System: basic types are defined by the .NET Framework type system and are common to all languages that target the .NET Framework Class Library
    - Development frameworks and technologies: the .NET Framework incudes libraries for specific areas of application development, such as ASP.NET for web application, ADO.NET for data access, and Windows Communication Foundation for service oriented applications
    - Language Interoperability: language compilers that target the .NET Framework emit an intermediate code named Common Intermediate Language (CIL), which is complied at run time by the common language runtime
      * With this feature, routines written in one language are accessible to other languages and programmers can focus on creating applications in their preferred language or languages
  + Version Compatibility: applications that are developed by using a particular versions of the .NET Framework can run without modification on a later version
  + Side by side execution: the .NET Framework helps resolve version conflicts by allowing multiple versions of the common language runtime to exist on the same computer
    - This means that the multiple versions of applications can also coexists and that an application can run on the version on the .NET Framework with which it was built
  + Multitargeting: by targeting the .NET Framework Portable Class Library, developers can create assemblies that work on multiple .NET Framework platforms, such as Windows 7, 8, 8.1,10, Windows Phone, and Xbox 360
  + (<https://msdn.microsoft.com/library/hh425099(v=vs.110).aspx>)
* Is a software framework developed by Microsoft that runs primarily on Microsoft Windows
* It includes a large class library known as Framework Class Library (FCL)
* Programs written for .NET Framework execute in a software environment known as Common Language Runtime (CLR)
  + An application virtual machine that provides services such as security, memory management, and exception handling
* FCL and CLR together constitute .NET Framework
* (<https://en.wikipedia.org/wiki/.NET_Framework>)

**Present details about it (including code and non code based examples)**

**Provide Pointers to additional material on the topic for interested readers**

**Example of Desktop Application**

**Why should software developers care about this topic?**

**Other information**

**References**

<http://www.pcmag.com/encyclopedia/term/41158/desktop-application>

<http://www.streetdirectory.com/travel_guide/114448/programming/desktop_applications_vs_web_applications.html>

<http://www.investintech.com/resources/blog/archives/78-popular-desktop-apps-and-their-online-alternatives.html>

<http://successfulsoftware.net/2013/10/28/is-desktop-software-dead/>

<http://arstechnica.com/information-technology/2012/05/no-cost-desktop-software-development-is-dead-on-windows-8/>

<http://www.kalzumeus.com/2009/09/05/desktop-aps-versus-web-apps/>

<https://medium.com/@collinmathilde/why-desktop-apps-are-making-a-comeback-5b4eb0427647#.vkprytwgs>