Apple Worldwide Developers Conference

Company Participants

- Ali Ozer, Director Cocoa Frameworks
- Andreas Wendker, Vice President, Tools & Frameworks Engineering
- Darin Adler, Senior Director Internet Technologies
- Jeremy Sandmel, Senior Director, GPU Software
- Jon Andrews, Vice President Core OS
- Josh Saffer, Director Swift Frameworks
- Kristin Forster, Manager AppKit Team
- Lori Hylan-Cho, Manager Watch Frameworks
- Matthew Firlik, Senior Director, Developer Tools
- Sebastien Marineau-Mes, Vice President Intelligent System Experience
- Sri Santhanam, Vice President Silicon Engineering Group

Presentation

Andreas Wendker

Welcome to the State of the Union. If you go behind the scenes of the new technologies, we just announced earlier in the Keynote. We give the inside story, the developers need to know. We give you details on the new APIs you should become familiar with and the motivation for the deep changes we are introducing. And of course we show you lots of demos of all of this in-action.

As you heard during the keynote, this is a huge year for our software platforms, as they take massive leaps forward. One of the most exciting announcements of this year's WWDC, is of course the transition of the Mac product line to our own Apple Silicon, which we propel the Mac into an amazing new future. The new macOS Big Sur, in addition to supporting these low level changes in hardware, as in all new look and feel to go along with them, making it an entirely new operating system inside and out.

Similarly iPad OS 14 has a refined look and feel that makes the iPad experience even more distinct and powerful with improvements to many apps and support for Scribble with Apple Pencil. In iOS 14, users can now interact with your apps and exciting new ways by creating widgets that run right on the home screen. And they can discover apps directly in the moment of need with lightweight app clips without having to download the entire app first.

And in the new watchOS 7, you can build a whole set of a flexible complications to surface up-to-date information to your users. From the big screen of the Mac to the

iPhone, you're holding in your hand, every product is carefully designed to have amazing unique experiences that users love. But for developers, we are making our frameworks and APIs and even the design language in our user interface is more-and-more consistent, making it easier than ever for you to create a great user experience across all types of Apple devices.

All the system features are supported by a huge number of new APIs for you developers. All with first class support in the Xcode tools and many of them based on SwiftUI, which is one of the most important foundations for use across all our platforms. We are also constantly working on making our platforms open to you, allowing you to plug into our OSs in more-and-more ways. You might recall that a few years ago, we introduced Extensions, a way for your apps to customize our operating system functionality. Since then, we've been adding plugins for share sheets, VoIP calling, rich notifications and more.

Starting this year, you just can make your app the default app used for email in iOS and iPad OS. And you're just continues to default browser too. And developers can bring the existing web browser extensions to Safari from Mac using the WebExtension's API. With a simple command-line tool that is shipping with our Xcode tools, you cannot bring extensions over from Chrome, Firefox and Edge, as privacy preserving Safari extensions and ship them in the Mac App Store.

We are also happy to announce that we're launching a new program that will allow third-party accessory makers and device manufacturers access to the Find My network. Find My is a huge end-to-end encrypted and anonymous network of over 0.5 billion Apple devices. They can be used to locate a missing Mac, iPhone, iPad or Apple Watch, even if it can't be connected to the Internet.

And show the location to the owner, right in the Find My app, it works great for your Apple devices. And it's perfectly suited to locate all kinds of other objects as well. To learn more about this program and to see the draft specification that is available today, go to developer.apple.com. There are so much to cover this year, you better dive right in.

Starting with the next transition to our own Apple Silicon. Using Apple Silicon inside Macs, is an amazing new direction for the future. And we'll open the door for huge improvements to speed graphics performance, power consumption, security and more. And unifying architectures across our product lines will allow us to bring unmodified iPhone and iPad apps to the Mac for the first time. We've been working on it for several years and we are so happy with the result. It's going to be awesome.

Let me give you a sneak peek and take you on a quick tour of macOS Big Sur, running on a Mac with Apple Silicon. Here is Finder with the applications folder. Mail and Messages are fully functioning of course. For example, I can drag and drop attachments between them. And Calendar looks beautiful, it's incredibly smooth. Browsing the web is fast in Big Sur's huge update of Safari. So is reading News.

And moving around in Maps, is responsive and quick. Photos looks great on Macs with Apple Silicon as well. Same for my music and podcasts. Productivity apps like iWork's Pages, Numbers and Keynote run grate and this is one of the most demanding pro-apps, Final Cut Pro is real time rendering of several movie streams and Logic, our music creation powerhouse, easily keeping up with large projects.

We already worked with a number of our partners to bring the apps to Apple Silicon as well. Here is Adobe Photoshop, already taking advantage of the new architecture and running fast on this new Mac. And Microsoft has brought up their Office apps, such as Word and Excel. Of course, our developer tools also were great in Apple Silicon and can be used to create native apps. And as you can see here an activity monitor, when we say Big Sur and all the software is running great on Apple Silicon, we really mean it. Every single bit of code you just saw is a 100% native.

The entire suite of system apps, Apple productivity and pro-apps, developer tools and even the first versions of third-party software packages like the Adobe Creative Cloud and Microsoft Office. So we are already far along and our move to Apple Silicon, but you might be wondering, why we are making this transition? The reason is simple. If you want to build the best mix possible for our users and we can build them without own processes.

The hardware innovations that have taken place in the mobile device space and the extremely tight integration of the iOS software and the Apple Silicon have led to amazing user features and behaviors that other platforms cannot offer to this day. Now on all the energy we put into refining iOS for Apple Silicon for years-and-years, we come back to macOS in the form of share operating system components and frameworks and this will take the Mac to the next level.

Because, just like code written for the Mac originally served as the foundation for iOS more than 10 years ago, we can now take everything we learned about optimizing iOS and make the Mac amazing been running on Apple Silicon. Macs will be more secured, they'll be fast and more responsive. They have longer battery life and they would seem to be much better overall for graphics. Making the Mac an amazing platform for games for example.

One of the hallmarks of macOS on Apple Silicon is of course how the hardware and software works so well together. To give us some insight into how this happens in practice, I'm going to have a short chat with my colleague Sri from our Silicon team. And just like you for the last few months, I've been having a lot more face time calls. So I'm going to call him at the lab.

Hey, Shri..

Sri Santhanam

Hi, Andreas. How are you?

Andreas Wendker

Hey, I'm good. Thank you. And thanks for taking the time to talk with us. During the keynote, Johny gave us the motivation behind the families of chips we've been building over the last decade. Could you quickly give our developers the high points?

Sri Santhanam

Sure. We've been building many different chips for our products for the past decade. And one of the key ideas when we build chips is that we really focus on what's truly important for each product. For example; in the iPhone, we needed to deliver a level of performance that had never been seen in a battery powered device that small. For the iPad, it started with driving an incredibly high resolution display without sacrificing battery life.

And for the Apple Watch, we needed a combination of performance, efficiency and features that had never been put together in such a tightly integrated package. So we built a scalable architecture that powered each of these chips and help make each of these products best in class and we haven't stopped. We continue to improve the architecture and drive performance and efficiency, while building industry leading features. We also executed to scale when you look across all of our products, we've shipped over 2 billion SoCs over the past decade. And not only that, billings of additional chips to enable our products. You put that altogether and it makes for an exciting transition for the Mac to Apple Silicon.

Andreas Wendker

That's for sure. So what exactly are you building for the Mac?

Sri Santhanam

Well, we are building a family of SOCs, designed specifically for the Mac. Just like we did for the iPhone, iPad and Apple Watch, we're making sure the chips we build are tailored to the unique needs of the Mac. And practically speaking, that's going to mean, we're going to bring to the Mac a whole new level of performance. Our scalable architecture that I've already mentioned, translates really well to running the types of application we see in the Mac. We've never really constrained ourselves to only looking at iOS workloads, when designing our chips. We've always looked at a wide variety of application and application traces, because we always want to build the best chips possible. So this really sets the stage well for the Mac.

Andreas Wendker

It should us. Now you mentioned Performance. How do power and energy efficiency playing to all of this?

Sri Santhanam

Yes, that's a good question. The iPhone really taught us that energy efficiency and performance were tightly intertwined. In the most early and power constrained devices, you need to improve energy efficiency. And that means to add performance without increasing power and that's not as easy as it sounds. All of the technologies we've built over the past decade have focused on this one paradigm.

And it turns out that same discipline and focus that paradigm translates really quite well to the Mac. All the systems built today are constrained by thermals, power consumption or both. So that's true for the Mac, just as it is for the iPhone. So obviously we're talking about a whole different range of power consumption. For any given Mac system, improving efficiency, means improving performance. And as you know, all our Macs have different enclosures. So our goal is to deliver more performance within each of these enclosures and that's made possible, thanks that relentless focus on efficiency or performance per Watt over the years.

Andreas Wendker

That's great to hear. Can you tell us a little more about what else will come along with this transition?

Sri Santhanam

Sure. The feature story for the Mac is also a great one. Remember, we've built highly integrated SoCs, which include a number of features and technologies that are critical in enabling our products. We build these technologies for the iPhone and iPad Pro and they're coming along in this transition to Apple Silicon for the Mac. So in addition to our CPU, our display and media processing engines, our ISP, our Secure Enclave, Advanced Power Management, are all examples of industry-leading technologies that we're bringing to the Mac.

And for the first time, the Mac will have a dedicated influence accelerator for machine learning applications with a Neural Engine coming over to the Mac. Of course, one of the things I'm most excited about, is that we're bringing our high performance GPU architecture to the Mac. We've already seen our GPU architecture do really well in the iPhone and iPad Pro. And in the Mac, this architecture is going to be great for Pro application and games.

Andreas Wendker

Well. I'm sure a lot of macOS developers are eager to get their hands on these new Macs. What does the transition mean for developers to build for our other platforms?

Sri Santhanam

Yes, I think this point is a very important one for developers. This transition will give our developers a common architecture behind all of our products. So whether

you're building software for the Apple Watch, iPhone, iPad Pro and now the Mac. There is a unified scalable architecture behind all of them, which should make it easier to write and optimize their application across all of our platforms. It really is a really exciting time.

Andreas Wendker

It really sounds great. Thank you, Sri for taking the time to talk with me.

Sri Santhanam

Thanks Andreas. Hope you and our developers have a great WWDC.

Andreas Wendker

Yes. See later. So as you heard, there are countless ways Apple Silicon creates a better computing platform for the Mac. Going forward software can take advantage of the amazing Silicon features we built. Such as the unified memory architecture, which lets the GPU Access Memory without delay. The neural engine for blazingly fast machine learning applications and the Secure Enclave that safely stores users keys. And all that we provide users with more powerful everyday experiences than ever before.

Let me give you some examples of this. The advanced power efficiency of Apple processors will let us keep cached cloud content fresh and up to date for days, even if you're Mac goes to sleep. Or we will be able to run a higher quality hardware supported 4:4:4 encoder for even better image quality and connecting a Mac to an iPad with Sidecar. Our users will be able to experience the best in class platform security pioneered by Apple. Comprehensive runtime integrity and data protection technologies out of the box with the flexibility and configurability of macOS.

And performance will be off the charts in many areas. In particular, the Apple Silicon unified memory architecture, powerful GPUs in our highly optimized Metal APIs, are going to allow for responsive complex image editing workflows, such as the one you see here in Affinity photo or cinema 4D, an app optimized for Metal for real-time interactive 3D editing and modeling. You can see how smoothly it runs on Apple Silicon. Every Mac for the Apple Silicon is going to have that powerful GPU and Metal implementation. Making Mac as a fantastic platform for running games like DiRT Rally, at high frame rates. And every Mac with Apple Silicon is also going to have a Neural Engine making machine learning applications so much faster.

Here you can see a demonstration using advanced machine learning to identify objects in the video clip, evaluating each frames image content. On Apple Silicon, the video runs super fast on the GPU and the machine learning model gets evaluated separately by the Neural Engine. As you can easily see, the Neural Engine allows for a whole new experience and the Mac with an Apple SoC finish the task much quicker. And yet, with all the new capabilities in advantages that Apple Silicon offers, there is one thing that certainly stays the same.

Macs will stay Macs, the way you know and love them. They will run the same powerful Pro apps. They will offer the same developer APIs Macs have today. They will let users create multiple volumes on disk with different operating system versions and they will let users boot from external drives. They will support drivers for peripherals. And they will be amazing Unix machines for developers and the scientific community that can run any software, they like.

So how are we going to approach this transition? Well, with this WWDC, we're launching an entire Quick Start Program that will enable you to bring a Mac apps to the new architecture and take full advantage of the capabilities Apple Silicon will offer. We will provide you with all the tools, documentation and even prototype hardware you will need and several of the videos from this year's WWDC, we provide you with a guide for how to get started.

Starting today, we even make a Developer Transition Kit or DTK available to you. The DTK looks like an existing like many, but it's completely different inside. It runs on an A12Z process, the same chip that powers our current generation iPad Pros. The current premium version of macOS Big Sur will be pre-installed on it, as well as a version of our latest Xcode tools with the capability to create apps that run on both Apple Silicon and existing Intel Macs. The DTK is a prototype though and not fully representative of the hardware we were actually shipped to customers.

There are some software features that will not be supported on it, but it does have everything you need to get started on supporting Apple Silicon. To apply for access to DTK, simply go to developer.apple.com later today and register for the Quick Start Program. We'll be shipping our DTKs to you as early as this week. And so the transition actually starts today, and it starts with you, the software developers. We expect to ship our first Macs with Apple Silicon by the end of the year and that means that you have to get your software ready for it quickly over the next several months. And the best way to do this is to create a universal app.

Matthew Firlik

We've made it really easy to bring your app to any user on any Mac by making your app universal. A universal app contains code compiled both for Apple Silicon and Intel processors. The two binaries becomes slices in a single executable. And the operating system chooses the right one for the hardware architecture of the Mac, the user is running on. Everything else in the App is the same. All the resources, such as images and machine learning models are identical for both hardware architectures. This means you can distribute a single universal app to your users and it runs on any Mac, whether they use Apple Silicon or Intel processors.

We've used this approach before. For example, a few years ago, when we ask you to include both the 64-bit and 32-bit version of your app binaries. For most apps, all it takes to create a universal version is opening a project in the latest version of Xcode and building. Let's take a look. During development, Xcode builds my aap just for the native architecture from my machine to save time. So getting started on a DTK is as simple as clicking run to launch by app natively on Apple Silicon.

To create a universal version, Xcode offers a new Any Mac option. This app is using the standard architectures built setting, which automatically includes Apple Silicon and Intel. So I can select this option and I'm ready to build universal. For the rare case, where I'm not using standard architectures, the menu lets me know. Here, I can see I previously set my frameworks to build just for Intel. When I build my framework for any Mac, Xcode will recommend that I update to use the standard architectures build setting. I can just quick update and build and now I have a universal version.

Andreas Wendker

You can turn most Mac apps and frameworks into Universal binaries with very little effort. This is true for low level libraries in Mac apps that we created with our Mac system frameworks such as AppKit and SwiftUI. But equally so for apps created with Mac Catalyst. While catalyst uses a different set of frameworks that utilizes UIKit APIs and sort of AppKit APIs, they are native apps compiled for the Mac platform just like any other Mac app and so they too will easily turn into universal apps when building them in Xcode.

For many apps, this work will only take a few engineering days. Even for some of the most complex software packages, if generally seen that it only takes a few weeks to get a full version up and running. In fact, as you've seen earlier, we've already reported all of Apple system apps, iWork, enterprise management tools, and pro apps and worked with a number of important partners to get them started and to prove out our tools and reporting process leading up to this WWDC.

And the results have been amazing. Even pro apps such as Final Cut Pro or Adobe Photoshop run beautifully on Apple Silicon. These apps represent some of the most advanced mature software packages, running on the Mac platform supporting the most demanding needs of pro users. And we found consistently that adding support for Apple Silicon then quickly and problem free.

We also passion about making sure that open source projects thrive on the Mac. Especially, if they benefit the largest software developer community and can help accelerating the port of other software packages. So we have already done the initial work for some of the more widely used open source projects to hit the community to get started. And we be publishing patches to them in the next days.

Another important partner we've been working this is Unity.

Jeremy Sandmel

Unity is an industry-leading platform for creating interactive real-time 3D content. It is used by millions of developers to make games and apps that run across Apple's iPhones, iPads, Apple TVs and Macs. Now, we've been working with them and within just a few weeks, they were able to update their powerful 3D editing and game development tools to take full advantage of the power, performance and features of the new Macs build with Apple Silicon. As a developer artist, you can use the Unity Editor to create, edit and build your games. So let's see it in action.

Here we have loaded Unity's Spaceship Demo project into the 3D Editor and it's all running on a Mac with Apple Silicon. We can edit our content, our geometry, our textures, shading, even our lighting and much more. The Spaceship Demo app, users Unity's high definition render pipeline, which uses Metal to drive the Apple designed GPU in these systems and delivers the advanced rendering features required to run modern high-end games. Now these effects we're seeing here, can be created using the VFX graph, an artist friendly toolset that makes it really easy to create complex effects that can be rendered right on the GPU.

Now that we're done editing our project, we can make a build of our spaceship app. The Unity Editor has also been updated to build universal apps. So now with just one click, you can build your game to run great on both existing Intel based Macs and the new Macs with Apple Silicon. Now that we built our Spaceship Demo app, let's see it running on a Mac with Apple Silicon.

Here you can see how Unity's runtime engine uses Metal's modern graphics and compute pipeline and unified memory architecture, to deliver advanced rendering techniques, such as volumetric lighting, ambient occlusion and real-time reflections. Because Unity's runtime has already been optimized for Metal and the Apple designed GPU. Your games will automatically run with a full native graphics performance on the Mac with Apple Silicon.

Next month, Unity plans to release a preview version of their software with support for Macs with Apple SoCs. So you can get started right away, building your games as universal apps. And later this year, Unity plans to release our production version of the editor runtime and tools, to enable you to easily create, build and ship your games optimized for both existing Intel based Macs, and those with Apple Silicon. So with powerful tools like these, we can't wait to see all of the amazing new games, you're going to be able to create.

Jon Andrews {BIO 17564490 <GO>}

Games built native for Macs with Apple Silicon are going to be great. But if your software needs a little more time or you're an end user who relies on software that's not been ported natively in Apple Silicon. macOS Big Sur will include new version of our emulation software, Rosetta 2. This provides compatibility for existing Mac software. Now Rosetta 2 is just an amazing technology written from the ground up for Apple SoCs, allowing existing Intel apps to run seamlessly alongside your native apps.

It's able to take Intel executables and translate them into native ARM instructions. Rosetta use the number of advanced techniques for binaries run very fast, even during emulation. This is especially true for Metal or Rosetta produces natives goals to the GPUs built into Apple SOCs. So even high-end games execute with incredible performance.

Another major technology is ahead of time compilation. So that nearly all applications are translated before you even launched them. And if you use macOS

packages or the App Store to distribute your software. Rosetta will translate the application in the background during the install. All others have translated on first launch. And if you dynamically load code yourself, of a JITs like Javascripts, then the code will be translated on the fly.

All code generated is optimized for Apple Silicon and tuned for our high-performance CPUs. Now we have completely integrated Rosetta into macOS Big Sur, so there is no need for any special setup. When system calls were executed, that translated into native calls to the kernal. Rosetta is also secure, with all its caches integrated with system code signing bound to your specific machine and OS version. And we're including Rosetta in macOS Big Sur as a transitional technology. To give you time to complete support from Macs with Apple Silicon at your own pace.

Let's dig into the technology with some quick demos. Starting with Xcode, running on a Mac with Apple Silicon. I've selected to run under Rosetta and Xcode builds my app at the same way it would for an Intel Mac, but then launches it under Rosetta. You don't need to configure special debug mode to work in this environment, which is a real enhancement for Rosetta 2.

So here we are using the App, able to interact with it as you'd expect. And I've hit a break point. The familiar tools for development, including debugging, static analysis, testing and more are available. I can even use the memory graph debugger to browse objects. Let's load the memory graph, it takes just a moment and it works like you would expect. So Xcode can build and run my app for both Rosetta and Apple Silicon.

Now let's look at Rosetta running third-party apps. Starting with the App Transmit running under Rosetta. I'm going to connect to OneDrive and download some drone footage that I've taken. We have an emulated version of all the macOS frameworks running alongside the emulation of the apps in the same process. So I'm going to simply download the file, because Rosetta is running in a special container file system and network access and just like any other app.

Okay. Let's take a closer look at apps you saw earlier in the State of the Union. Starting with DiRT RALLY, which really shows us some great features of Rosetta. The translation cache was built during the install from the Mac App Store, completely transparently, which means there's no glitches due to a need to translate new blocks of code. As it's really important for games with motion and real-time audio requirements. And things just work even this game controller that I really need to be able to stay on the road here. And CPU instructions will be translated very accurately, whereas the graphics commands are sent through to the native GPU. There's simply no frame drops or lags on the controls, it's amazing.

Okay. Now let's look at a Pro App, Affinity Photo, which really helps show off the emulation has very little memory overhead. I've got this huge 82-mega pixel image loaded here. There is no additional memory overhead for having such a large file

open under Rosetta and because the translation cache is filed backed, just like the original executable, this little additional overhead here too.

And we didn't just focus on the speed of translation. The translation cache is also secure. Thanks to the codes signatures that are checked before execution, just like the original binary. Saving a file is an all-native operation through our optimized storage stack. The emulation doesn't add any copies or buffering, coupled with native Metal, I'm able to apply these effects quickly and save them just as fast. Rosetta 2 is wickedly fast, transparent to users and completely integrated into macOS Big Sur. It's a great transitional technology that will give you time to migrate your apps to support Macs with Apple Silicon.

What about code other than apps? For example, app extensions can be built as universal binaries too. And since the extensions run out of process from the host app, Rosetta can even emulate extensions when they're being used with a native host app. So end users can rely on extensions to continue working on Macs with Apple Silicon, while you work on porting them to run natively. And other standard plug in architecture such as audio units run out of process under Rosetta 2.

Drivers utilizing the DriverKit API, introduced in macOS Catalina, are in the same situation. If you ship a driver for example, for a USB peripheral and have not yet adopted DriverKit, now its a great time to make the change. That way you can create a universal driver that will support all Macs. We will continue to shrink the surface area for Kernal extensions, replacing them with safer APIs going forward. And while we will continue to support running kernal extensions on macOS, you can't run them in emulation and the increased security we offer on Apple SoC's will require all KEXTs to be notarized.

Driver kit offers a much better alternative, with less friction for the end user and we highly recommend you adopt it in all situations possible. One area that we have worked on removing Kernal extension usage is for virtualization products. The Mac is the world's best developer platform. Many developer workflows require using tools from different platforms are performing server deployments. To meet these needs, many of you spin up virtual machines running different operating systems or lightweight containers like Docker.

We know this is critical to your work and important we support these needs. That's why we've been working hard with our partners to support virtual machines on Macs with Apple Silicon. As you saw in the Keynote, we started working with Parallels and I want to walk you through an early port of Parallels Desktop for Mac, running on the Apple Hypervisor that's been updated for Apple Silicon.

This is running on a prototype Mac, with Apple Silicon that supports virtualization. Here we have a fully featured ARM version of Debian Linux running on Parallels Desktop. Remember, this doesn't require any third-party kernel extensions to be installed now. I've installed Swift for Linux and started a new project with Vapor. A

Swift web framework, you may be familiar with. Let's go ahead and build our new project. And then we'll start the server.

I can go ahead and load the page from this server in Firefox for Linux. I'd like to be able to test this in Safari in macOS. And we have Webmin installed to help manage this machine and Apache configured to proxy port 80 of the VM to the local Vapor server. So let's start it. Now we can go into Safari and macOS and see how it looks there.

It looks great. These environments can interact due to features built into macOS Big Sur. And thanks to the Hypervisor framework with complete support for networking, storage, input devices and much more. Virtualization technology is used by container solutions such as Docker. And we know it's incredibly important for many of you, as it's a very common for service site development and testing. So we'll be working with Docker to enable this in the coming months.

We've mentioned before, that this transition puts Apple platforms on a common architecture. And one of the coolest benefits of this is the ability to run software originally built for iPhone and iPad. Completely unmodified. Let's take a look at how that works.

Ali Ozer

As you may have guessed, the technology that enables running iPhone and iPad apps on Macs with Apple Silicon is the same technology that enables Mac Catalyst. In effect, Catalyst is an implementation of the iOS frameworks for macOS, the same APIs made from Mac. And these frameworks when running on the Mac with Apple Silicon are even binary compatible. This makes it possible to run many iPhone and iPad apps without any kind of change. Unmodified iOS apps, however, will not get all the customizations that Catalyst apps can offer.

In particular, it's worth pointing out that running unmodified apps will only be possible on Macs with Apple Silicon, not on Intel based Macs. So you may want to check that Catalyst box and Xcode and start making your app even better for your Mac users. We are planning to make unmodified iPhone and iPad apps available in the Mac App Store, once we launch our Macs with Apple Silicon.

In fact, all the apps that users purchased on iPhone and iPad that are eligible to run on the Mac will simply show up as purchased in the Mac App Store as well. You, the developer will have total control or whether you wish to participate. But we believe this will be a great addition for Mac users for apps that don't already offer a native Mac version.

Now let's look at an iPad game. Monument Valley. This app is the exact same on available in the iOS App Store. When we launch it, the app appears in the dock like any other Mac app, I can minimize and re-open the window. Menu bar is an important part of the Mac experience. And iOS apps get a menu bar generated

automatically. For instance, the Edit menu here with the usual entries and app menu as well.

You will notice preferences here, which puts up and in-app preferences panel automatically created from your apps iOS settings bundle. Also high and even quit, which is a thing on the Mac. Let's hide our other apps and get back to the game. We have our beautiful graphics. Mouse clicks are mapped to taps. I can zoom in as expected on the track-pad, I can click and drag to rotate and create a path for our character. The game is fully playable.

The application on the Mac is in the Applications folder by default. But there is a user, I can rename it or move it out to desktop, say if I wish. When iOS apps are installed and launched from system managed-app containers and are not used to having their pads arbitrarily modified. To achieve our users expectations, while also remaining compatible with the apps, we play a couple of tricks. First, we have a new app bundle format, which wraps your application as is. This allows the user visible app to be freely renamed or moved.

Second, to ensure that apps are launched as compatibly as possible, we use a future we introduced in macOS Sierra called app translocation. This allows the iOS app to be efficiently launched from a sanitized path that is not affected by the user's actions. Let's look at a different type of app, Documents by Readdle. While Monument Valley is an example of a nice pure gaming experience, Documents is a productivity tool that uses a number of advanced iOS features and it's a good example of how these come across on the Mac.

For instance, this application supports iPad multitasking, which means the window on the Mac is automatically resizable. I can even go into full screen. As you see here, we have macOS scrollers and I can scroll with two fingers on the trackpad. The app automatically adjust to macOS appearance. Dark and back to light. And since, this app supports multiple windows on the iPad, it also does so on the Mac.

From within the app, I can access resources on the Mac, such as my photos library. Accessing other files brings up a native macOS open panel. Here for example is a PDF file. iOS apps can also automatically make use of macOS shared services, such as the ability to bring up a Mac mail composed share sheet for emailing content.

I want to show you one more thing that extensions in your iOS apps work, where appropriate on the Mac. Let's open photos, choose a photo and edit it. I have an iOS app installed and it provides a photos editing extension. This extension automatically appears in photos extension list. I can choose it and enhance my photo. That's amazing. Using an extension provided by an iOS app, as is in the Mac Photos app. That's a look at iPhone and iPad apps on Macs with Apple SoCs. Users will love the flexibility to run these apps at their desktops. And remember that for you the developer, it takes just a few extra steps to unlock more customization and refinement of your app for the Mac through Catalyst.

Andreas Wendker

Macs with Apple silicon represented incredibly powerful computing platform. They take advantage of the most advanced and power-efficient chip technology offering unprecedented power per watt performance and incredibly graphic speed. And they run a wealth of software, ranging from native universal apps fully optimized for Apple Silicon to Rosetta emulated apps not yet ported from Intel Macs, to apps from other environments running in virtualization, all the way to unmodified iPhone and iPad apps.

We are looking forward to opening the next chapter for the Mac and seeing how you take it to the next level, by leveraging the benefits of Apple Silicon in your own apps. And now let's talk about macOS Big Sur. Beyond support for Apple Silicon, Big Sur is a huge software release. To go along with the transformative hardware changes we are announcing. This release is packed with new capabilities, is faster than ever and has a whole new look.

Kristin Forster

The new look and feel of MacOS Big Sur eliminates visual clutter in the user interface and gives the Mac an air freshness. And as you'll see in a moment, we've brought the best designed from each of our platforms together. We've created a family resemblance between iPad OS and macOS, while retaining the power and flexibility of the Mac. This similarity makes it easy for users to transition between our devices, while making Mac Catalysts and iPhone and iPad apps naturally feel at home on the Mac.

Starting with the dock, we see a main theme of nacOS Big sur. The Doc is rounded and floats above the bottom of the screen slightly. The dock corners echo a curvature of a new design for app icons, which all are conformed to a standard shape. We will provide templates for you to design your icons to match. Control Center has come to the Mac, providing easy access to system controls. And Notification Center has a whole new look, built completely with SwiftUI.

Notifications in widgets are now shown simultaneously rather than being sequestered in the separate tabs. Notifications from each application are grouped together. You can provide a content extension to show custom content using notification API, which we brought to macOS from iOS and made available to both AppKit and Mac Catalyst apps.

The menu bar floats at top of your desktop picture and the selection floats within it. Menus have a refined layout and menu selection echoes the rounded theme. In the Finder, we see a new design for document icons. You can customize your own document icons by simply providing an image and text. macOS will automatically position that elements mask them and apply the page curve.

Sheets come up in a new way. They no longer rollout of the toolbar, but instead float above the window in a rounded platter. The parent window is dimmed, reinforcing

the modality of the sheet. The toolbar in this preview window sits at top of the content area, while the sidebar is a visually distinct vertical element. Your windows will automatically adopt this new structure, if you are using a toolbar, a full size content view and a split-View-Controller with sidebar styling. Mac Catalyst and SwiftUI windows receive a full sized content view by default.

There are a number of major styling changes in the toolbar. It's taller and the window title is vertically in line with the toolbar controls. The in-line title hide its document icon until you roll over it, contributing to the streamline visuals. Most toolbar items are position to the right, except the navigation items which remain leftmost in the toolbar. You can designate a toolbar item to be navigational.

Toolbar controls have a new large size and show they're backing, when you roll over them. The search field clocked a centered lift when the toolbar is space constrained and expands our tough full field when clicked. You can get this great new behavior in windows of your app by embedding your search field in a new search toolbar item. There is additional toolbar structure, if you have the three pane layout like mail.

The mail toolbar is split into two sections with controls corresponding to the message list positioned over that pain and controls corresponding to the conversation view positioned there. We've added toolbar API to AppKit in Mac Catalyst to express this separation. Swift UI toolbars attached to navigation view pains get automatic separators. As you can tell the toolbar has gained a lot of new functionality. If you are already using and its toolbar now is a great time to start.

Images in both toolbar and sidebar are provided via SS symbols. There are thousands of SS symbols now available on the Mac and we are using them widely. SS symbols can be configured to perfectly match the size and weight of text. In the sidebar, the symbol seamlessly adapt to provide correctly sized and styled images when I change my sidebar icon size.

Controls have a brand new look as well. Shown here in the system preferences pop up and radio buttons. Sliders, progress indicators and other controls, have also been updated with a fresh new look. We are really embracing color in this release. We've given your applications a mechanism to define an app specific accent color by providing a color name in your P-List. This effects not just sidebar icons, but also selection color and the color of standard controls.

Mail uses a blue accent color. We've got even further with the color theme, giving apps a way to assign colors to individual sidebar items to delineate functionality. In the mail side bar, the VIP items are coated yellow, smart mailboxes our monochrome and local mailboxes are till. The combination of color through tinting and shape through SS symbols contributes to great sidebar design.

Notes is another example of app specific accent color. Here the yellow branding of notes comes alive in the list selection and text highlighting. We hope you enjoy choosing an accident color for each of your apps that expresses its unique

personality. As you can tell every little detail of the Mac has been made fresh and modern and there are many opportunities for your own applications to participate.

Andreas Wendker

Big Sur feels really fantastic and Mac Catalyst apps also look better than ever before. We continue to improve the technology with every release and you continue to prove out the implementation in our own apps, before making it available to you. Catalyst is basically a Macs specific implementation of the iOS APIs, most notably UIKit. And why we of course recommend SwiftUI in AppKit, if you want to take full advantage of the Macs capabilities, this technology makes it very easy to bring iPad OS apps to the Mac with a single shared code base to support all iPhones, iPads and Macs.

This year, you're making a number of improvements that will make the experience of Catalyst apps even better. Some features require a small amount of adoption work on your part, but you get even better results for the Mac. Most importantly, the new Mac Idiom allows you to run your apps at the native resolution of your Mac, instead of using the default scaled user interface that matches iPad sizes. The Mac Idiom puts you in total control of the user interface and controls to draw to better match the rest of the apps on the Mac even more closely.

We also bring in more APIs into the set of supported Catalyst APIs that will provide even better compatibility with iOS and will allow you to make very lightweight customizations in your code that they adopt your apps to running inside macOS on all Macs. We're using it ourselves in critical system apps like messages, which is now written with catalyst to give Mac messages parity with its iOS counterpart. Let's take a look.

The new version of messages has everything you want from a Mac app. Rich menus, key equivalents and multiple windows to take advantage of the large Mac display. Popovers give quick access to details and integrated find helps navigate and communicate efficiently. With Catalyst, you get a familiar experience on both the Mac and iOS, so you can use new additions like pinned conversations, inline replies and mentions, along with existing favorites such as the Memoji and Messages Effects.

The next version of Swift Playgrounds, which we are planning to ship later this year, also shine in the new Mac idiom. It fully takes advantage of the new Mac look and feel with a simplified toolbar and full height sidebar. Adjustable tech sizes and streamlined editing with pop overs and the Mac system pickers.

Like Messages, Swift Playgrounds will support multiple windows, make it feel right at home on the large screen of the Mac. If you are an iOS developer and have not yet considered creating a Mac Catalyst app, now is a fantastic time to do so. So Big Sur starting the transition to Apple's Silicon is packed with new features includes a much improved catalyst and on top of all of that has a great new look. It's incredibly how much has changed in the release, Big Sur is a huge step forward for the Mac. In fact,

so much of the general architectures are improved and we are giving macOS a new number.

Big Sur Next is macOS Version 11. Because it really is a new Mac operating system all around. This is such an exciting year for the Mac, there has never been a better time to be a Mac developer and we hope you are as excited as we are about the deep changes we are making and the future it opens up for the platform.

Next, let's talk about the great improvements we are bringing to iPad OS.

Josh Saffer

This has already been a huge year for iPad. Just a few months ago, we added trackpad support alongside the new Magic keyboard, continuing iPad's multi-year evolution into a powerful platform for the future of computing. And only a few short years ago, iPad OS gained support for dragon drop, a familiar interaction, completely re-thought for a multi-touch experience.

Next came enhanced multitasking capabilities, including the ability to open multiple windows from your favorite apps. And this year, we're taking iPads capabilities even further, with new ways to navigate and interact with the apps themselves all designed for iPad. We've brought these new ideas to apps across iPad OS. And of course, we added new APIs so that you can bring them to your apps as well.

Let's look at a few examples including the brand new side bars, new pickers that streamline data entry and some new opportunities to refine the interactions in your apps. So let's get started with side bars.

The new sidebar support a standard 2-column layout like you have today, but with just a swipe, they can expand to a 3-column layout on every size iPad. This 3-column view is even available in portrait orientation, making it easier than ever to browse and organized content even on the smaller screen. Adding a sidebar to your app is easy with all the new functionality available and existing components that you already use. A sidebar can also simplify navigation in your app, like in photos where it includes rose for library and albums, which were previously found in the tab bar along the bottom, but also rose like favorites in places, which used to be nested further down.

By bringing all the key parts of your app to one place, sidebars can make your apps deeper functionality, much easier to access. They also help with organization, with support for displaying hierarchical content and an outline, including collapsible items. These new outlines build on API you're already familiar with in both SwiftUI and UIKit, which have been enhanced to support these new appearances and behaviors.

It's really easy to add a sidebar that perfectly matches other apps on the system and it can help your app take even better advantage of iPad's large screen. You'll also

find some great opportunities to improve data entry in your apps. We've added a number of new pickers like the brand new date picker that lets you choose a date by just finding it on a calendar. It supports this inline style that makes great use of iPad screen realistic and if you need to fit it in a smaller footprint, it also has a more compact style. There's also a new in line emoji picker that can be invoked with the same keyboard shortcut as on macOS and all your iPad apps will get it for free.

You'll also find a new color picker API, which brings a standard way to pick colors across all apps. It's really easy to adopt and it provides your app standard behaviors like a place to save your favorite colors for later use. And an eyedropper to pick a color right off the screen.

As you use the new iPad OS apps, you'll notice even more refinements that make interacting with them more fluid than ever. But there are some great opportunities to enhance your apps in similar ways. For example, new lightweight context menus are a great way to organize common actions. We've converted most interactions that previously used in action sheet to these new menus. And I think you're going to love them as much as I do.

Everything we've discussed was designed for iPad. But if you choose Mac Catalysts to bring your to the Mac, these new APIs are also designed to look great on macOS. And as you look deeper, you'll find many refined interactions in individual apps like the new in line rename and files. Each of these changes is small, but they add up to a much more powerful iPad experience and finding these opportunities in your apps can be easy too. We look for ways to make deeper functionality, more discoverable to streamline navigation and to remove modal states. And I'm sure that you can find opportunities to do so in your apps as well.

These are just some of the enhancements that we've made to apps across iPad OS. And hopefully, it gives you some ideas for how you can make your apps more powerful too. It's always exciting what you can do with software alone, but let's also explore how you can incorporate iPad's hardware capabilities to further enhance your apps.

Sebastien Marineau-Mes

IPad's capabilities are extended even further through its powerful hardware and accessories. And earlier this year, we announced the new iPad Pro that comes equipped with a LiDAR Scanner. Now the LiDAR Scanner gives iPad Pro an incredibly precise we have measuring distance and it paves the way for developers to build entirely new experiences. The LiDAR Scanner uses direct time of flight to project light onto objects up to five meters away. And that light is then reflected back on to our custom designed sensor and by measuring how long that process takes, we obtain precise distance measurements of your surroundings.

Now you can access this information through APIs in ARKit. The first is seen geometry, which provides you with a 3D mesh of the scene. We also use a neural network to identify the planes and surfaces in the scene like the floors, the walls, the

doors and the windows. The new depth API gives you access to the precise distance information captured by the LiDAR Scanner. And what we see here is a 3D depth map that was created using the output from the depth API.

Combined, these APIs bring a new level of realism to all AR experiences on iPad Pro. Now all virtual objects are placed instantly, motion capture and people occlusion is dramatically improved and with measurements captured by the LiDAR Scanner, virtual objects can be accurately placed in front or behind items in the world.

It brings a whole new level of realism to all AR experiences. The LiDAR Scanner also unlocks new capabilities and professional apps like shaper 3D to quickly scan a user's room and generate a 3D map of their surroundings. And it can also be used for entirely new apps that leverage iPad Pro precise understanding of the physical world, like those built for industrial maintenance, design and manufacturing. Making iPad Pro a powerful tool for manufacturers, architects and creative professionals.

Now, another creative way to extend the capabilities of iPad is with Apple pencil. Apple pencil is an amazing tool that turns iPad into a powerful drawing and note taking device. This year, we've added great new capabilities to make handwriting on iPad just as easy and powerful as type text. Many of these improvements will automatically work in your app without having to do any additional work.

So let's take a look at Pencil's awesome new capabilities in action. Here's a sample app that we've built that help students practice their handwriting skills. And what's great about Apple Pencil is that it lets you work in a free form way. You can just start writing anywhere. Writing with pencil works in any standard UI text field. We've added UI text fields here, so that students can write their name at the top of the assignment. And with the new iPad OS these text fields automatically support scribble. And that means that any hand written word is automatically converted into tight text.

Now, let's get started on the assignment. Notice that as soon as the pencil is picked up, this beautiful drawing tools appear and the drawing canvas. And because our app users pencil kit, these beautiful tools are provided automatically. The new PencilKit makes handwriting as powerful as type text. So you can select hand written text using the exact same gestures that you would for type text.

We've also added a new gesture called drag to select, which allows you to select by sweeping the pencil over entire paragraphs, align or even a single letter. This makes copying handwritten notes a snap and it's a sneaky way to get through this assignment quickly. Now under the hood, drawing with pencil uses a new canvas API. These new capabilities have been added directly to PencilKit, so if you're already using it in your iPad app, you'll inherit all of these great new features for free. And this includes the color picker, which makes your apps and this homework, a little more fun.

Now one of your biggest request for PencilKit is the ability to get stroke data as pencil is moving across the screen. And I'm very happy to say that this capability is now available in PencilKit. Using the new stroke API, you get access to the important attributes of the drawing like angle, pressure, drawing ink and exact location, grouped into strokes and updated as the user draws. This capability is perfect for our handwriting app. So let's practice.

The student improves their handwriting by tracing these letters. And it was easy to write this app. We simply use PencilKit to show how the pencil is moving across the intended lines. We also added comparison logic, so that our App can choose to let the student know that they're not quite tracing the letters as closely as they should, did help them as they practice. The app is built for pencil and thanks for the new drawing policy API scrolling with your finger is still consistent with the rest of the system.

An iPad OS let's you identify separate areas, some for free form drawing and others for text entry, which specifies where to apply scribble. Now as you can see, these new updates to iPad OS and PencilKit make handwriting as powerful as type text. Scribble offer as a seamless writing experience in any of your app's text fields and the new stroke API delivers real time drawing information, enabling pencil kit apps to offer new experiences like annotation, markup and recognition.

Now these are just some of the amazing new features that continue to push what's possible on iPad. We can't wait to show you more of the exciting new iPad features this week and we'll see how all of you customize your apps for iPad. And of course iPad apps also benefit from the great features coming in iOS 14. So let's dive in iOS 14, is an awesome update that is jam packed with new features for you to transform your apps. Two of the most exciting developer features in iOS 14 are widgets and app clips, both of which have been designed to extend your apps experience and delight your users.

So first let's talk about widgets. Widgets just have always been a great way to convey information at a glance. And in iOS 14, we've completely re-imagine this experience. Creating all new widgets that take designed to the next level and use on device intelligence to show users the right information at the right time. There are more beautiful, data rich and now they come in three sizes. The new widgets are available on the today view. And for the first time, we're making them available on the home screen.

Now users can get at a glance information from your apps, whenever they go home. A space that users visit many times a day. The new widgets were written in SwiftUI, making it easy to share code across iPhone, iPad and Mac. New widgets that works seamlessly across all three platforms. Now displaying widgets on the home screen, means that they're always visible and always running. So performance and memory efficiency really matter. The previous approach of using a live running extension, just could not deliver the efficiency that the home screen requires.

And to solve this for new widgets, we leverage SwiftUIs ability to describe your widgets appearance, separate from when and how it is rendered. And we've added a new method to serialize a SwiftUI view into an archive. Now this archive is very light weight and can be rendered asynchronous on the GPU to ensure great performance of the home screen. The new widget API widget kit is structured around timeline entries, each of which contains one of these new efficient Swift UI archives, as well as time and date and a relevant score. Each time line entry is rendered on demand when it is both timely and visible to the user.

So when your widgets becomes visible, we can efficiently construct and render a full view higher key from this archive, without needing to run your apps code. Your typically provides multiple timeline entries, so the process is repeated throughout the day without requiring your app to be constantly running. Now each widget the user has configured, provides its own set of timeline entries and the system renders each of them at the right time. And to help optimize the use of space on the home screen, users can stack widgets. Stacks come in all sizes and the user can easily flip through each widget that is part of the stack.

Now taking this idea even further, is the smart stack, using on device intelligence, iOS surfaces, the right widget to the top of the stack, adjust the right time. And this is where the relevant score that you provide in your widgets timeline entries comes into play. It is your signal to the system that's something of high interest is happening. Such as this news alert and this weather warning.

Now, thanks to SwiftUI, we've created this efficient and intelligent widget experience that is unique to every user. And now I'd like to show you a demo. Before we dive into Xcode, let me show you how easy it is to lay out widgets on your home screen. I've already customized my home screen with some of my favorite widgets. You'll notice weather, everyone's favorite topic of conversation and my Emoji Rangers widget, which shows my Panda character and an online multiplayer game.

I can figure these widgets in the widget gallery. Each of these views in the widget gallery are entries that you've created in code. The system knows how to display the timeline entry by writing it to disc to just in time render this UI. This technology allows us to create this vis-e-vis gallery for users. This technology also allows us to create a smart stack, which our users can flip through. In addition, the system will intelligently analyze all time line entries from widgets in a stack and no went to service the most relevant information for each user.

For example, the system knows when there is a relevant calendar event coming up and the stacking intelligently rotate to that widget. There, now I have my widgets beautifully organized on my home screen. Now this widget was actually created by my team. We built this widget so that we can see our character status in the game.

Now lets dive into Xcode, so they can show you how we created our Emoji Rangers widgets. In our iOS game, I define my widget, in my WidgetKit extension. To define a widget, all you need to do is conform to the widget protocol. Conforming to the

widget protocol returns of widget configuration to the system, which consists of our kind of widget in our timeline provider. Here, I've created a single entry and I can return that using the snapshot function.

Snapshot is used when the system wants to display a single entry and I've also created a series of entries in the timeline function, which tell the system when to display each individual entry based on the date that I have defined. This is really the engine of my widget. All I need to do is to find a widget and return to timeline. To tell the system when to surface my widget, I defined a relevant score. If I have multiple timeline entries, all relevant scores I provide will be weighed against all other relevant scores that I've provided and the system will dynamically display the entry that is most relevant to my users.

This widget is currently showing my own character in the game, but what I really want to see is how other players are doing and how I'm competing. WidgetKit provides a lightweight configuration ability, leveraging the intense framework. Developers simply pass in their parameters to be configured and the framework provides the configuration UI. And my team has already started to build this out, which you can see here.

They are using the intense timeline provider to generate a new timeline based on the intense values, which are really the answer to the question we're asking. Just lets me choose which character to see in my widget. Now I can enter edit mode and edit my widget, which brings up this configuration UI. This UI is created for me by the system. I didn't have to lift a finger to create this view. And just by using the power of intense, the system can dynamically create this UI on my behalf.

Now I can't even select my teammates characters to view their status in my widget. This is how you define a widget that is easily customizable and can be read by the system to intelligently surface the most relevant information from your app. Now these are just a few of the amazing possibilities of WidgetKit. Next, let's look at some of the other exciting features in iOS 14.

Darin Adler

We're also introducing App Clips. An App Clip is a small part of your app that is light and fast and easy to discover. So users can quickly get within need, right when they need it. Everything about App Clips is designed for speed. Let's start with this card, which quickly pops up. It's auto generated by Apple for metadata you submitted to the App Store. So users will find a familiar and safe. And as soon as you see this card, behind the scenes the App Clip is already downloading. And with just a tap, you could launch the App Clip.

You don't need to enter any credit card numbers, because App Clips can use Apple Pay for payments. You don't have to manually log into an account, because it could take advantage of Sign In with Apple. App Clips won't clutter your home screen and will only stay around as long as you need them. But you can easily launch recently used App Clips from the new App Library. And discovery is key, App Clips are all

about getting to a part of the app at the moment you need it. So it's critical that we made them really easy to discover.

You'll be able to open them from apps you use every day and places in the real world. And what a user sees are new Apple designed app clip code they will know there is an app clip waiting for them. And the great thing about App Clips for you the developer, is that they're made from a part of your app. You use the full power of the SDK. Make your clip as small as possible, so users are in the app instantly leave out analytics libraries, you don't really need do that and you won't be even close to the 10 megabit limit.

App Clips can present notifications within eight hours of opening one. Like reminding you that your part and spot is about to expire. There's even an App Click specific API that lets you verify the App Clip launched exactly where you expect it to be. An App Clip is the best way to introduce users to what your full app offers. So we made it really easy for them to upgrade to it. You can prompt the user to get the full version of your app at the time that makes the most sense for your App Clip. What's great? Is that we seamlessly migrate any choices, they've made over to the full App as part of that upgrade.

Here I have the Fruta app. Fruta lets you see a menu of delicious smoothies, place orders and explore smoothie recipes. Placing orders works by invoking Fruta with the right url. Let's make an App Clip for this ordering experience. First in Xcode, I'll choose new target and choose App Clip to create a new App Clip target. Next I'll select the classes and asset catalog that are needed for ordering. At add our App Clip to their target membership.

Now let's build and run. Great. Here's my App Clip. It preserves my apps ordering experience. You'll notice that my App Clip ask permission to send notifications to customers, when their order is complete. We can streamline this by using 8-hour notification permission, exclusive to App clips and designed for light and fast experiences.

To do this, I'll add a check. If my App Clip has that 8-hour permission, I won't request authorization. So the user won't be prompted. I also want to configure a store kit overlay, to let the user get my app. When the order is ready, it's the perfect time to present it, along with an explanation of what they will get in the full app.

And there you have it. Now my customers have a streamlined experience of my App Clip and when they're done ordering, they can get my app and take the recipe home. And that's App clips, immediately discoverable in a variety of ways. Built from a small part of your apps, so they launch fast. And with features like signing with Apple and Apple Pay, user privacy is built right in. In fact, privacy is incredibly important to our users. With our newest products, we're making it easier to give users transparency and control with respect to their data.

You now declare privacy information about your apps, right on the product page in the App Store. To do this, simply go to App Store connect and answer a few questions about how you collect share and protect your user's data. The highlights will be presented to potential user right on the App Store across all platforms. These are just some of the new technologies for you to take advantage of an iOS 14.

Lori Hylan-Cho

Last year was a very big year for Apple Watch developers. In Watch OS 6, we introduced independent apps, the App Store and Apple Watch and of course the first ever native UI framework for watchOS, SwiftUI. In Watch OS 7, we're giving you a new place to user SwiftUI skills and your Swift UI views with SwiftUI complications. And we're giving you more space on the face, by letting you create multiple complications for each family.

You now have incredible flexibility to provide beautiful and timely information when the user raises their wrist. We're also making it easier than ever to get your app and all its information which complications into users' hands with watch face sharing, which integrates App Store purchase and download right into the sharing flow.

Let me show you some of these cool new features with an app I've been working on to help my friends in Hawaii to track the hump back whales that visit in the spring and me to live vicariously through them from here, since travel is pretty restricted right now. In my Whale Watch app, you can see recent Whale sightings from several viewing stations around Maui and you can also submit a sighting, if you happen to spot a pod.

Now I'd like to make it easier for me to see what's happening with the Whales from here. And for my friends to long sightings themselves, right from the watch face. It's super easy to describe the complications I'm creating and the families they support. First, I'll offer complications that display data about the most recent sightings from the various viewing stations around the island, similar to how world clock offers a complication for each of the cities you've selected in app.

I can do that by looping to the list of stations and creating a complication descriptor for each one. As you can see, each complication uses the station name as a unique identifier. This makes it possible to have each complication navigate to a different part of my app. For example, I can have each station specific complication, go to the detail screen for that viewing station. Next I'll create a complication for quickly logging a Whale sighting.

Again, I can use the unique identifier to navigate directly to the screen for logging a sighting, when the user taps on it. And I can also use it to provide a different template for this complication, then for this station specific ones, while supporting the same families. Finally, I'll create a third descriptor from my season data complication. With the other complications, I'm supporting all families, but this one is more complex. So it only makes sense in the large rectangular space.

I'll finish up here by calling the Handler with my descriptors array sorter in the order I want the complications to appear when the user edits their watch face. I mentioned that Watch OS 7 now support SwiftUI complications and I have some great options here, from the new native APIs for gauges and progress fuse to repurposing SwiftUI views from my app. I've already got a chart of recent whale sightings across the island in the app, so I'll use that.

As you can see that as easy as wrapping the SwiftUI view and one of the new view templates. I'm using a version of my chart view with the font size adjusted down a bit and populating it with data from the past week, which is a nice adjustable amount of data for our watch face. To see how everything will look, I'll use my favorite new feature of Xcode 12 complication previous.

You may have noticed that was handling all the different complication families in that giant switched statement. Now I can see what those complications look like on different watch sizes and in full-color or tinted by writing a bit of SwiftUI. And of course, previous update live. So if I don't like how my complications look, I can adjust the code and see what impact that has.

Moving to My Watch. You can see how the complications I've built would appear to a user and where I can customize the complication selections to create the canonical Whale watch face for sharing with friends. I'll share this phone with myself to bundle with my app or post on social media. While I'm here, I think an island friend who has been testing my apps for me, would appreciate a face that includes our favorite water sports along with my Whale Watch complications. So, I'll make a quick face with surfing and paddle boarding complications, sighting data from the Whale watch station closest to her favorite beach and the log a sighting complication, in case she spots a part of Whales well out on her Board.

She will get a text for me with a preview of the watch face and when she taps on it, she will be offered the opportunity to get any app she doesn't already have from the App Store. When she add the face to My Faces, it will automatically be selected as the default face. So that's a sneak preview of multiple complications, SwiftUI complications and face sharing. We're so excited about these and other new tools and capabilities on Watch OS 7 that will help your customers connect to the information that matters most to them. And for you and your customers to share your awesome faces and the apps.

Matthew Firlik

You've seen an overview of the most important changes to our operating systems. And the best way to take advantage of the new functionality is to get the new version of our Xcode developer tools. The first thing you'll notice about Xcode 12, it's how really shows off the new-look and feel of MacOS Big Sur with a cleaner and more expressive UI.

The modernized toolbar puts controls right where I need them and the icons now have a fresh new appearance. The font size of the navigator now matches the system

sidebar setting. And I can adjust the size separately too, to get the look I like. Documents now show through under the toolbar and inspectors have an updated with new spacing and layouts. So Xcode 12 it's really gorgeous. I can get a simple and clean layout and more quickly. For example, I can maximize my workspace for code going into full screen and then use the slide out sidebar to quickly access files, where even perform a search to move throughout my project.

Now, I have something I'm really excited to show you. Document tabs. Document tabs appear in the familiar space across the top of the editor and are a great way to organize my work. I can use the dynamic tab shown in italics for most of my navigation. Any file I select in the navigator opens in this tab. When I find a file and want to keep open, I can double click on it or hold on option when I click and have it open in a new tab. And this works great too with actions like jumped a definition or with open quickly. I can also open any kind of content in a tab, such as the commit view from the branch I'm working on or a built lock I might want to reference later.

So tabs let me easily manage the set of content, I want to work on. I can also use many tabs too. For example, adding to the test classes I already have. I'll use the file menu to split my editor and then drag a group of tests into the tab bar. And now I can work through all the files one-by-one. And when I'm done working on them, I can close them all at once, by closing the split.

Document tabs build in the Xcode are fast and frontiers. Xcode 12 supports the new features and APIs in our latest STKs, providing a great way to develop for all Apple platforms. Xcodes knew SwiftUI app templates, include everything I need to create common code to share across platforms. And then tailor for each UI idiom. These groups and their targets are created automatically in new projects. And help me keep good layering, as my project grows. Keeping an app working great on our platforms, requires testing and Xcode 12 makes achieving broad coverage even easier.

For test, which have been factored into reusable functions, I can now navigate through the nested calls. Just as I would when debugging, to find the source of a problem. My test can now also evaluate the responsiveness of my UI, by working with animations. UI tests can specify the velocity to use for scrolls and presses. And performance tests can capture diagnostic information. I can review details such as the animation duration, frame rate, even the number of UI hedges that occurred.

Our biggest addition to testing is with Storekit. The new StoreKit test framework, let's me specify in app purchase and subscription information. Right in my project, so I can test without needing to stage or deploy my app. And working with StoreKit, is now interactive. Xcode has a new StoreKit transaction manager that shows details for app some debugging. Here's my current session running in the simulator. So now I can select an item, buy it and confirm the purchase and then review the transaction in Xcode. I can even refund it and my app updates live. So now developing apps with StoreKit is almost as much fun as purchasing items with it.

Of course, creating an app is way more fun than testing. And we've made working with Swift and SwiftUI, even more awesome. In the source editor with previews and especially using both. SwiftUI previews have a new toolbar across the top that speed up common actions. Here I can duplicate the preview and then use the inspector to change the appearance to be dark. This makes working with my previews, just like designing an UI control. I can now quickly edit my views by double clicking in the preview, which scrolls the matching code into view and smartly selects the content to change.

Adding modifiers is faster too. The inspector now recommends modifiers based on my selection. Join them here at the top of the list. And I can search for modifiers too, to find the ones I need. Code completion with Swift will feel totally new. The results come back up to 12 times faster and a simplified presentation, makes it easy to find what I'm looking for, particularly amongst similar results. And code completion it goes way beyond just helping me type, helps me get what I want.

The same place whereas I get from the library, they compile with no additional input, now appear for completion results too. This means I can make quick additions and customize later. Keeping my code and previews compiling the whole time. Along with the tools, helping enhance my UI, my UIs can now enhance the tools. Views and modifiers in my project can now appear in the library. And with their own default content by just implementing a protocol and they can do so from Swift packages too.

Swift packages now support all the necessary components, including building asset catalogs and resources and localizing content for use with Swift UI. So now I can Creates Swift UI controls in packages, design them with previews and add them to the library, all with just a little code. So I can switch back to my view, open the library and find my control and with the others. And then just add it to my UI and double click to customers. It is now easier than ever for me to create my own controls. We use them in all my projects and share them with others. There is a lot to discover in Xcode 12 when you try it out.

A great user experience with document apps. New testing functionality with Storekit, tonnes of workflow additions for designing great apps and so much more. And these are build on top of some dramatic improvements we've made this year with Swift and Swift UI.

Josh Saffer

With last year's introduction of Swift UI, we took a huge step toward a new way of building apps. We started with a strong foundation of core infrastructure and components and deep integration with our existing frameworks. The full surface of mature frameworks like AppKit and UIKit can't be covered all at once, so we're taking our time to get it right.

We're prioritizing source stability, while making the API more powerful over time. As a result, this year's Swift UI API improvements are all completely additive. Code

you've already written will continue to compile and run, but this year's releases with no migration required. That's as important to us, as it is for you, because Swift and Swift UI are playing a central role across the Apple ecosystem. We're evolving them together and creating a set of APIs powerful enough to build everything from the system features like the new control center and notifications center on macOS to entire multi-platform features, like the new widgets.

We've even gone as far as implementing the new color picker API and you UIkit using SwiftUI. This extensive usage requires great performance and the language and frameworks teams have worked together to achieve some dramatic improvements this year. You'll notice faster launch and layout, smaller code size and memory usage and more.

One of the most important pieces of SwiftUI's performance is the layout system. Many of you have been asking for a collection view to enable high performance display of large data sets. We didn't include one last year, because we had a bigger vision that we wanted to pursue. We saw an opportunity to support high performance collections without an entirely different set of APIs. You can already build complex layouts like this list of photos with SwiftUI today.

But large collections can be expensive. This sample contains 20,000 photos and consumes over 500-megabites of memory. With this year's releases, you can simply add the word lazy to your existing stack view to enable faster load times and dramatically reduced memory usage for large collections. This same 20,000 photos sample still enable fast scrolling to any point in the list. But now it loads instantly and consumes far less memory and for more complex needs, we've added new lazy grids as well. This is a great example of the flexibility that comes from SwiftUI's declarative syntax.

You can learn a single set of layout primitives and then use them to compose 10 views or 10,000, combined with new support for switch and if let's statements in the expanded Swift function builders syntax, you can now efficiently display large complex collections of views. Our focus was SwiftUI last year was to enable you to add new views to your existing apps, while sharing more of that code across all our platforms, all supported by deep integration with our existing UI frameworks. This year, we've of course enhanced and expanded on that foundation with new API covering things like outlined views and paging views. And we even began adding SwiftUI interfaces to other frameworks like MapKit and AVKit. But we have some bigger enhancements as well.

Today, we've talked about many technologies to help you bring your existing apps to new places. But with Swift and SwiftUI, we know that you will also want to build brand new experiences and deploy them everywhere. So we're also taking the next big step, expanding SwiftUI's learn once, write anywhere story with API that lets you deliver fully native experiences across Apple's platforms starting from your first line of code.

SwiftUI now includes an incredibly concise way to describe your app's structure in just a few lines of code. It's supports common elements, like windows and toolbars, but also platform specific features like menus and preference windows. It's a common set of app structure APIs for all Apple platforms that also supports the features that are unique to each one.

Mathew showed you the new shared App template. But let me show you what makes it work. To give you a sense of how powerful this will be, let's build a simple document based text editor. You'll notice the API is designed around Swift's new add main feature, which identifies the main entry point of your application. Apps use the same declarative structure that you're familiar with from SwiftUI views, enabling your text editors entire interface to be expressed in just four lines of code.

To start, will add a document group, which manages creating and opening files. Of course, a document needs a view to display and edit its content. So well I add a simple plain text editor and connected to our model. With just these few lines, we can build and run and we already have a fully functional, multi-document app with all the behaviors that you expect.

If we start typing, you'll see it automatically marks the document is edited and needing to be saved. And we even get automatic support for undo. The app is fully native to each platform, providing standard behaviors like open panels on macOS and on iPhone and iPad, you get a standard file browser.

Of course it's also easy to add platform specific functionality. For example, we can add a preference window front on MacOS with just a few more lines of code. Let's had a setting seen and fill it with a view that I created earlier. Composing the different parts of an app is just as easy as composing SwiftUI views. If we run it again, we'll find the preference window we expect. Brought up with the preferences menu item that was automatically added in the right place and configured with the appropriate key command.

SwiftUI's declarative app structure can provide appropriate platform native behaviors like these automatically. And just like with views, you can learn these new application APIs once and then apply them everywhere. We're following the same development philosophy too, starting with a strong foundation and deep integration with our other UI frameworks and then expanding support over time. This is the next big step in bringing you a language and API to build fully native experiences across Apple's platforms without compromising on what makes each platform unique all using Swift and Swift UI.

Andreas Wendker

And that concludes our overview of the most important announcements for Apple platform developers this year. The new releases are full of new features. There are so many new capabilities in APIs that we didn't even have time to talk about here in the State of the Union. Such as the new ways Core ML allow you to encrypt machine learning models and to deploy them via iCloud.

Or tvOS, which continues to be the best platform for the big screen, now with the Airplay support for 4K HDR streams, multiplayer support for games and many more new and enhanced functionalities. As usual, we are also making develop a previous of all our operating system and Xcode releases available to you for download at developer.apple.com. If you're a Mac developer, don't forget to sign up for the Quick Start program and apply for DTK with Apple Silicon there as well.

To learn even more about all the new technologies we've unveiled today, please take advantage of the more than 100 sessions available throughout the week. We can't wait to see how you are going to use the new API's and technologies and enjoy your great apps. Finally, here at Apple we've always drawn strength from the creativity of our global developer community. To celebrate this, we've made a film comprised of images you all shared with us from your homes, as you continue to create the work that inspires us all.

Thank you. Be well and I hope you enjoy the week ahead.