Xcode 10 beta 6 Release Notes

Developer

About Xcode 10 beta 6

Supported Configurations

Xcode 10 beta 6 requires a Mac running macOS 10.13.4 or later.

SDK Versions

Xcode 10 beta 6 includes SDKs for iOS 12, watchOS 5, macOS 10.14, and tvOS 12.

Installation

Xcode 10 beta 6 can coexist with previous versions of Xcode.

Prerelease versions of Xcode are made available from developer.apple.com, packaged in a compressed XIP file. To install Xcode during the beta period, download the XIP file, double-click the file to expand it in place, then drag Xcode-beta.app to the Applications folder. To use command line tools from the beta Xcode, run `xcode-select -s <path to Xcode>`.

The final release of Xcode 10 will be available in the Mac App Store. Previous versions of Xcode are available from developer.apple.com/downloads/more.

Accessing Additional Developer Tools

To launch additional developer tools, such as Instruments and FileMerge, launch Xcode-beta and select Xcode > Open Developer Tool. You can keep these additional tools in your Dock for direct access when Xcode isn't running.

Technical Support and Learning Resources

Apple provides the following resources to support your development with Xcode:

- Apple Developer Forums. Participate in discussions about developing for Apple platforms and using developer tools.
- Bug Reporter. Report issues, enhancement requests, and feedback to Apple.
- Apple Developer website. Get the latest development information as well as technical documentation for Xcode.
- Xcode homepage. Get high-level information about the latest release of Xcode. Download current and beta Xcode releases.
- For help with using Xcode, use the built-in help by choosing Help > Xcode Help.

Deprecation Notices

- The macOS 10.14 SDK no longer contains support for compiling 32-bit applications. If developers need to compile for i386, Xcode 9.4 or earlier is required.
- Support for Subversion has been removed. (33361671)
- Custom dtrace-based instrumentation that was available for macOS and Simulator only is no longer supported in Instruments. The new Custom Instruments packages based on os_signpost allow for greater flexibility and control over presentation of data and support all platforms. (38812912)
- Building with libstdc++ was deprecated with Xcode 8 and is not supported in Xcode 10 when targeting iOS. C++ projects must now migrate to libc++ and are recommended to set a deployment target of macOS 10.9 or later, or iOS 7 or later. Besides changing the C++ Standard Library build setting, developers should audit hard-coded linker flags and target dependencies to remove references to libstdc++ (including -lstdc++, -lstdc++.6.0.9, libstdc++.6.0.9.tbd, and libstdc++.6.0.9.dylib). Project dependencies such as static archives that were built against libstdc++ will also need to be rebuilt against libc++. (40885260)
- Libgcc is obsoleted. Xcode 10 can no longer build apps with deployment targets of macOS 10.4 and 10.5. (42818150, 38035243)

Command Line Tools

The Command Line Tools package installs the macOS system headers inside the macOS SDK. Software that compiles with the installed tools will search for headers within the macOS SDK provided by either Xcode at:

```
/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.14.sdk
```

or the Command Line Tools at:

```
/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk
```

depending on which is selected via xcode-select.

The command line tools will search the SDK for system headers by default. However, some software may fail to build correctly against the SDK and require macOS headers to be installed in the base system under `/usr/include`. If you are the maintainer of such software, we encourage you to update your project to work with the SDK or file a bug report for issues that are preventing you from doing so. As a workaround, an extra package is provided which will install the headers to the base system. In a future release, this package will no longer be provided. You can find this package at:

`/Library/Developer/CommandLineTools/Packages/macOS_SDK_headers_for_macOS_10.14.pkg`

To make sure that you are using the intended version of the command line tools, run `xcode-select -s <path to Xcode>` or `xcode select -s /Library/Developer/CommandLineTools` after installing. (38476941)

Release Notes Updates

Release notes are occasionally updated after a beta is distributed. The latest version can be found on the developer.apple.com/download.

Revision: XC100B6 - RNV1

New in Xcode 10 beta 6 – IDE

Interface Builder

- The exception “Impossible to set up layout with view hierarchy unprepared for constraint.” has improved diagnostics. (42514606)
- Scrolling performance, especially in very large storyboards, is improved. (34904204)
- Control-dragging from Interface Builder to a Swift source file defaults to creating `@IBOutlet` when the insertion point is above the first method in a class, and `@IBAction` otherwise. (36891045)
- Discouraged uses of Vibrant Light and Vibrant Dark appearance are now reported. `NSPopover` and `NSVisualEffectView` are exempted. (42036478)

Source Editor

- Xcode no longer allows creating discontiguous selections using the command modifier key. To create discontiguous selections, hold control+shift during the selection. (42564654)

Testing

- When running tests from xcodebuild or Xcode Server with iOS Simulator destinations, test failures will result in the collection of log archives embedded in the test action result bundle. (42172805)

Resolved in Xcode 10 beta 6 – IDE

Build System

- Dependencies for Swift applications are properly tracked including `CopySwiftLibs` for test targets. (42285144)

Interface Builder

- Fixed a crash when viewing the Attributes inspector for an `NSLevelIndicatorCell` in a cell-based `NSTableView`. (39231884)
- Embedding views in macOS documents now supports `NSVisualEffectView`. (39924232)
- Interface Builder can properly embed `NSGridView` when views span multiple columns or rows. (40436057)
- Search results can be viewed in a macOS document containing an `NSTableView`. (42787652)

Testing

- macOS UI tests run properly on macOS Mojave without requiring that Xcode is listed in the list of Accessibility-approved applications in the Security & Privacy preferences pane. (Security & Privacy > Privacy > Accessibility.) (42188063)
- Resolved an issue that could cause `xcodebuild` to crash after running tests in parallel on the simulator. (42070331)

Instruments

- When using the Profile action, Xcode brings up Instruments' template chooser with the target application pre-selected. (42566328)

Devices

- The Open Console button in the Devices and Simulators window properly opens the Console app. (39955550)

Localization

- Xcode will no longer crash when importing a localization for a sticker pack. (39251257)
- Fixed a crash that sometimes occurred when importing a localization that would add a new file to an existing variant group. (39364780)

Known Issues in Xcode 10 beta 6 – IDE

General

- Opening Xcode projects and workspaces stored in iCloud Drive, or changing source control branches for an open workspace or project stored in iCloud Drive, might cause Xcode to hang. (34086758)

Build System

- xcodebuild will export an invalid `AppStoreInfo.plist` when using an `ExportOptions.plist` with `<key>generateAppStoreInformation</key><true/>`.(42745138)

  Workaround: To invoke iTMSTransporter with the -assetDescription flag, you may generate a valid `AppStoreInfo.plist` via the `swinfo` command line tool located in `Xcode.app/Contents/Applications/Application Loader.app/Contents/Frameworks/ITunesSoftwareService.framework/Support/swinf`

- The new build system doesn't yet support On Demand Resources (ODR). If your project uses ODR, the legacy build system can be reenabled in File > Workspace/Project Settings. (31508570)

- When performing `xcodebuild clean` on a project that uses a customized build location outside the derived data directory, and that has older build products produced prior to Xcode 10, Xcode might report an error indicating that it won't delete directories not created by the new build system. (40427159)

  Workaround: Delete the build folder manually.

- When build settings are defined in an `xcconfig` file that inherit values using `$(inherited)` or `$(<setting_name>)`, the Build Settings editor shows the last assigned value for the setting instead of the properly composed value. (28572636)

- After installing a Swift toolchain, Xcode won't immediately detect the toolchain. (31360319)

  Workaround: Close and reopen the workspace.

- Custom build system plugins are not yet supported by the new build system. (32795438)

  Workaround: If required, switch to using the legacy build system in the File > Project/Workspace Settings sheet.

- The Intel ICC compiler isn't supported by the new build system. (33286594)

Workaround: If required, switch to the legacy build system in the File > Project/Workspace Settings sheet.

- The new "Build with Timing Summary" command doesn't show timings for some tasks run within the build system during the build. (39801746)
- Targets which have multiple asset catalogs that aren't in the same build phase may produce an error regarding a duplicate output file. (39810274)

  Workaround: Ensure that all asset catalogs are processed by the same build phase in the target.
- Targets with Copy Headers build phases ordered after Compile Sources build phases may fail to build and emit a diagnostic regarding build cycles. (39880168)

  Workaround: Arrange any Copy Headers build phases before Compile Sources build phases.
- In the new build system, shell scripts can't rely on the state of build artifacts not listed in other build phases. (For example, the `Info.plist` file or `.dSYM` files.) (40852184)

  Workaround: Add the file the script depends on as an explicit input dependency to the shell script build phase.
- When using the new build system, only build settings with a nonempty value are exported to shell script build phases and build rules. A build setting set to an empty value is treated the same as one which isn't set at all by both build systems. (40843445)

Source Editor

- Code completion for Swift image literals was removed in Xcode 10. (38087260)

Source Control

- Xcode doesn't support ed25519 encrypted SSH keypairs. (40912136)

  Workaround: Use an SSH key pair that uses a different form of encryption.
- Under some conditions, the Xcode Clone window might fail to show projects from within groups, with GitLab.com or GitLab - self hosted accounts.

  This may occur when you have a small number of personal projects. (40787204)

Debugging

- In the sidebar of the Shader Debugger, the button for accessing the detail view may be positioned so that it can't be pressed. (40665537)

  Workaround: To display the same information, increase the width of the sidebar or hover over the variable in source code.

Create ML

- Create ML isn't supported for use in Command Line Tools. (40637430)

Devices

- Running a WatchKit app with Xcode 10 that was built with a previous version of Xcode can give an installation error "The WatchKit app has an invalid stub executable." (40567857)

  Workaround: Clean the build folder and run the app again.

Instruments

- Instruments may fail to launch if Xcode has not finished preparing any attached devices for development.(43066159)

  Workaround: Wait for Xcode's device setup phase to complete and then open Instruments.

- In macOS 10.14 beta, a wirelessly paired Apple TV might not appear in Instruments. (41662978)

Simulator

- The OS can take several minutes to boot for the first time in a simulator. (40535421)
- Weak linking against frameworks that are not present in simulator runtimes older than iOS 12.0, tvOS 12.0, and watchOS 5.0 yet are present in macOS Mojave can cause an app to crash on launch when running in those older simulator runtimes. (41224059)

  Workaround: Use older simulator runtimes on macOS High Sierra.
- Video output may stop playing on external displays if using keyboard commands for fast forward and rewind. (41917187)

  Workaround: Pull down the control panel from the top-right corner of the screen and then hide it again.
- Simulator will prompt for Microphone access on startup. If you decline permissions Simulator audio sessions will not be able to use audio input of any kind, regardless of the permissions granted inside the Simulator. Use the macOS System Preferences, Security & Privacy pref pane to change this setting.

  Your application must still be granted Microphone permission inside the simulator as well. There are two levels of permissions: macOS applies its permission policy to the Simulator as a whole, across all simulator runtime versions and all applications inside Simulator. The Simulator applies permission policies to individual applications just like devices.

  For CI and unattended systems you can pre-authorize Simulator by running `/Library/Developer/PrivateFrameworks/CoreSimulator.framework/Resources/simul trampoline -preflight`.

  This dialog will present as requesting access for “simulator-trampoline”.

New in Xcode 10 beta 6 – Swift Compiler

Swift Compiler

- To help prevent inconsistent hashing for Cocoa objects, overriding the `NSObject.hashValue` property now produces a compiler warning, and `NSObject.hash(into:)` is now marked non-overridable. To customize hashing in `NSObject` subclasses, you need to override the `NSObject.hash` property, not `hashValue`. For example, here is a sample implementation for an `NSObject` subclass with a custom definition for hashing. Note that hashing and equality go hand in hand: if you override `hash`, you also need to override `isEqual(_:)`, and vice versa (42780635):

```
class Person: NSObject {
    let name: String

    init(name: String) {
        self.name = name
        super.init()
    }

    override func isEqual(_ other: Any?) -> Bool {
        guard let other = other as? Person else { return false }
       return other.name == self.name
    }

    override var hash: Int {
        var hasher = Hasher()
        hasher.combine(name)
        return hasher.finalize()
    }
}
```

Known Issues in Xcode 10 beta 6 – Swift Compiler

Swift Compiler

- In the presence of certain errors in the code being compiled, an incremental (non-whole-module) compilation can terminate in a fashion that fails to report errors. For example, the build log might only state "Command CompileSwiftSources failed with a nonzero exit code"(43033749)

  Workaround: Switch to another compilation mode by changing the build settings in one of two ways: - Use "Add User-Defined Setting" to add a setting named `SWIFT_ENABLE_BATCH_MODE` with value `NO`. This change will restore the legacy single-file compilation mode (like Xcode 9.4), which can still run incrementally. - Change the "Compilation Mode" build setting from "Incremental" to "Whole Module". However, this change will generally result in slower incremental compiles.

- Passing a `let` property of a generic class that has function type as an argument to another function or method may cause a compiler crash.(41056468)

  Workaround: Assign the property to a local variable, and pass the local variable as an argument instead.

```
class A<B> {
    let function: (B) -> B
}

func takeFunction(_: (Int) -> Int) {}

func passFunction(from a: A<Int>) {
    // reference to a.function as an argument here may crash
    // the compiler
    takeFunction(a.function)

    // workaround: assign to a local variable, and pass the
    // local variable instead
    let function = a.function
    takeFunction(function)
}
```

- The compiler may crash or miscompile when forming an array of heterogeneous class objects as an `[AnyObject]` array (42666956):

```
func f(_: [AnyObject])

f([NSObject.self, NSString.self]) // may crash or miscompile
```

Workaround: Individually assign the class objects to variables of type `AnyObject`, then form an array of those variables:

```
let myNSObject: AnyObject = NSObject.self
let myNSString: AnyObject = NSString.self

f([myNSObject, myNSString])
```

New in Xcode 10 beta 5 – IDE

Source Editor

- Xcode now supports syntax highlighting for Swift compile-time diagnostic statements and line-control statements. (40887538)

Testing

- The General pane of Xcode's preferences now includes UI for controlling the degree of parallelism that Xcode should attempt while running tests in parallel. The "Mac Test Parallelization" slider can be used to adjust the number of runners to spawn when running macOS unit tests in parallel. The "Simulator Test Parallelization" slider can be used to adjust the number of simulator clones to spawn when running iOS or tvOS app/UI tests in parallel. Setting either of these sliders to "Auto" (the default) instructs Xcode to pick a default number. Note that when testing from `xcodebuild`, the `-parallel-testing-worker-count` and `-maximum-parallel-testing-workers` command line options take precedence over these values. (41779908)

- XCTest has added new API on `XCUIElement` for capturing the entire state of the UI, either for exporting to external systems or for inspecting locally. It includes a type representing a snapshot of the UI and API for exporting that snapshot to a dictionary of standard value types (strings and numbers). (35168151)

- XCTest now enforces that `XCTestExpectations` must not be waited on more than once. This prevents accidental misuse and is especially important for `inverted` expectations because they cause waiting to take the full timeout duration which might cause tests to execute more slowly. (41641176)

  One way that an expectation might be accidentally waited on multiple times is if it's created using `XCTestCase.expectation(description:)`, is waited on using `XCTWaiter.wait(for:timeout:)`, and then `XCTestCase.waitForExpectations(timeout:handler:)` is called. XCTest will now raise an exception when this occurs, listing the problematic expectation(s) which should be adjusted.

  For example, the following code suffers from this problem:

```
func testFoo() {
    let expectation1 = self.expectation(description: "expectation1")
    doSomething { expectation1.fulfill() }
    XCTWaiter().wait(for: [expectation1], timeout: 1)

    let expectation2 = self.expectation(description: "expectation2")
    doSomething { expectation2.fulfill() }

    self.waitForExpectations(timeout: 1) // API Violation exception
                                         // since it waits on both
                                         // expectations, but
                                         // expectation1 has already
                                         // been waited on
}
```

Here are two possible fixes:

```
func testFoo_fix1() {
    let expectation1 = self.expectation(description: "expectation1")
    doSomething { expectation1.fulfill() }
    XCTWaiter().wait(for: [expectation1], timeout: 1)

    let expectation2 = self.expectation(description: "expectation2")
    doSomething { expectation2.fulfill() }

    self.wait(for: [expectation2], timeout: 1) // Only wait on
                                               // expectation2, which
                                               // has not yet been
                                               // waited on
}

func testFoo_fix2() {
    // Create an XCTestExpectation directly, not using XCTestCase
    // convenience API
    let expectation1 = XCTestExpectation(description: "expectation1")
    doSomething { expectation1.fulfill() }
    XCTWaiter().wait(for: [expectation1], timeout: 1)

    let expectation2 = self.expectation(description: "expectation2")
    doSomething { expectation2.fulfill() }

    self.waitForExpectations(timeout: 1) // Waits only for expectation2
}
```

Interface Builder

- Named colors can now symbolically reference system colors. (39196638)
- The macOS library now contains an “Import From Device” menu item for Continuity Camera. Its submenu will be auto-populated at runtime on macOS 10.14 and higher; on previous versions, it will be disabled and have no submenu. (41474755)

Create ML

- Augmentation now supports the ability to crop images. (42077294)
- `MLDataTable` now supports math operations, logical conjunctions, and additional methods. `MLDataTable` now also supports slicing, column type conversion, and multiple column subselection. (38199042)
- Validation data can be specified using `MLImageClassifier.DataSource` and `MLTextClassifier.DataSource`. (40203967)
- When importing CSV or JSON files in an `MLDataTable`, `MLDataTable.ParsingOptions` are supported along with automatic type inference. (42074416)

Resolved in Xcode 10 beta 5 – IDE

General

- Fixed issue that could cause Xcode to crash when working with some Windows-formatted Markdown files. (35725637)
- Removed duplication in the "Superclasses" submenu of the jump bar's Related Items menu. (32954204)
- The Code Snippet pane accepts hitting enter to insert the selected code snippet at the cursor locations. (40781401)

Build System

- Open, idle Xcode workspaces do not prevent building with `xcodebuild`. (42247751)
- Localized `.strings` files which are encoded using UTF–16 and don't have an explicit encoding selected in the Xcode project file will now build. (41902162)
- Xcode's new build system supports build setting expansions in the advanced build location options found in Location Preferences or Project Settings. (40236969)
- The "Clean Build Folder" command will clean if you are using legacy build locations. (41059641)
- Copying a read-only file won't cause the build to fail with an error. (41813879)

Source Editor

- Fixed an issue that caused semantic highlighting to disappear in files that are open in multiple tabs. (39675025)

Source Control

- The performance of the Source Control Change Bar in the Source Editor has been significantly improved for large files. (38916653)

Interface Builder

- When adding constraints with the control-drag canvas menu, shift-clicking on multiple items now adds them to the canvas instantly, rather than after the menu is closed. (18991966)
- Fixed an issue where storyboard view controllers would change their relative placement unexpectedly while navigating between code and the Interface Builder canvas. (40925441)
- Fix a crash for macOS documents that were using "Can Draw Concurrently" with `@IBDesignable` instances. (41593368)
- Fixed a crash when opening or compiling a document that uses system colors that were introduced in a later macOS version. Now, the document XML tracks such dependencies and prevents opening on an earlier Xcode. (41012499)
- When adding constraints with the control-drag canvas menu, shift-clicking on multiple items no longer requires clicking an "Add Constraints" menu item. Clicking outside the menu now works. (22970158)
- Fixed missing proxy icons when a `.xib` file's document outline is collapsed. (41449749)
- Fixed a crash when clicking an Interface Builder warning in the Issue navigator. (41904920)
- Fixed an issue where `@IBDesignable` views wouldn't build when the project included a WatchKit target. (36281806)

Asset Catalog

- Fixed a hang that could happen when opening an asset catalog that's part of an external project reference. (14235592)
- The background in the Asset Catalog is now lighter in Dark Mode, which improves visibility of template images. (40408386)

Testing

- Resolves an issue that could cause testing in a simulator to fail with the error message: "Could not launch simulator for iphonesimulator". (41862568)
- Fixed an issue where testing on an iOS device sometimes failed with an error message: "Too many instances of this service are already running". (41645466)
- Running UI tests in parallel in a simulator won't cause tests to become slow or hang. (42271166)

Instruments

- Fixed an issue preventing profiling of unit tests with no host application in iOS simulators. (39334812)
- Custom Instruments based on os_signpost and os_log instrumentation work in Simulator. (40661069)

Server

- Selecting SSH credentials for Xcode server bots won't cause Xcode to crash. (41848927)

Create ML

- Changing the name of a model saves correctly when clicking off of the field. (40437531)

Simulator

- Siri works in simulated tvOS devices running tvOS 12 beta 5 or later. (37195435)
- iOS 12 Simulator now prompts for permission to use audio input. You must provide a `NSMicrophoneUsageDescription` in your `Info.plist` file and gracefully handle rejection of permissions. Use the Settings app inside the simulator to change this permission. (41750344)
- Recording videos with Simulator won't cause the recording process to quit unexpectedly. (41723474)

New in Xcode 10 beta 5 – Swift Compiler

Swift Compiler

- `Random.default` is renamed `SystemRandomNumberGenerator` to avoid misuse and to conform to naming guidelines. For more information, see the amendment in SE–202. (42298827)

Swift Package Manager

- Swift targets are now compiled using batch mode. (39262812)
- The package manager emits deprecation notices for manifests that use the version 3 API in a package dependency graph. (41792011)

Resolved in Xcode 10 beta 5 – Swift and Apple Clang Compilers

Apple Clang Compiler

- LLDB works correctly when stopping in Swift code while running tests after printing out an error about the minimum deployment target of the Swift module. (41652444)

Swift Compiler

- The `Bundle` initializer `init(for:)` from Foundation now returns the correct bundle when used on a class that inherits from a generic class, even if that class isn't itself generic. (40367300)

- An improved error message is displayed when you pass an optional value where an unwrapped optional value is required by context. The error message now presents a clear choice between different strategies for fixing the problem, describes what each strategy means, and provides Fix-it dialogs to repair your code. (42081852)

  For example:

  ```
  error: value of optional type 'X?' must be unwrapped to a value
  of type 'X'
    f(x)
      ^
  note: coalesce using '??' to provide a default when the optional value
  contains 'nil'
    f(x)
      ^
       ?? <#default value#>
  note: force-unwrap using '!' to abort execution if the optional value
  contains 'nil'
    f(x)
      ^
       !
  ```

  When accessing a member of an optional value, the Fix-it dialog includes optional chaining as a potential solution:

```
error: value of optional type 'X?' must be unwrapped to refer to
member 'f' of wrapped base type 'X'
  let _: Int = x.f()
               ^
note: chain the optional using '?' to access member 'f'
only for non-'nil' base values
  let _: Int = x.f()
               ^
                 ?
note: force-unwrap using '!' to abort execution if the optional
value contains 'nil'
  let _: Int = x.f()
               ^
                !
```

- The runtime stability of availability conditions (`if #available`) is improved. (41849700)
- The Swift 4.2 migrator is enhanced to eliminate build errors associated with overrides of `imagePickerController(_:didFinishPickingMediaWithInfo:)`. (41828411)
- The runtime stability of generic conversions to `AnyHashable` is improved. For example, the stability of creating a dictionary using the `init(uniqueKeysWithValues:)` initializer is improved because the keys you pass are implicitly converted to `AnyHashable` using the improved conversion behavior. (40583597)

New in Xcode 10 beta 4 – IDE

General

- Holding the Option key when opening the Library will cause it to remain visible until it is manually dismissed, rather than automatically closing after each use. (40880961)
- Added export option to Quick Look popover for data types such as NSData. (41370369)

Debugging

- NSVisualEffectView properties are now presented in the view debugger inspector. (18124270)

Resolved in Xcode 10 beta 4 – IDE

General

- Fixed a crash when pressing the tab key in the code snippet editor. (41296805)
- Fixed an issue where the text range of a find result wouldn't be highlighted when selecting from the Find Navigator. (37820835)
- Fixed a crash when reopening a project that has a pre-Xcode 5 format .xib file open. The upgrade prompt is now correctly displayed in this case. (39462313)
- Fixed a problem where issues for different objects were incorrectly coalesced. (13162286)

Build System

- Further improved the performance of incremental builds with the new build system. (28580112)
- When a target that contains Swift code adds a new dependency on a Swift system library, Xcode will copy that runtime library into the app. (40456595)
- When using the new build system, output from shell script build phases displays in the build log. (40492041)

Source Control

- A pull or merge will pause file operations for the current project or workspace, not all of them. (39994122)

Source Editor

- Highlighting of a matching brace can now be disabled with the following user default:

  ```
  defaults write com.apple.dt.Xcode DVTTextShowMatchingBrace –bool NO
  ```

  (33876309)

- Fixed an issue that caused tabs before a closing brace to still be shown when folding code. (40041202)
- Fixed an issue where expanding a closure placeholder would move the cursor after the closure instead of to the first placeholder. (41046443)
- Fixed an issue that caused the source editor to skip a line when pressing up arrow on a wrapped line. (41504239)
- Fixed an issue that prohibited clicking in the code folding ribbon immediately after folding a region. (39361198)
- The source editor for AppleScript on macOS Mojave no longer covers the first few characters of each line. (41085793)
- Improved formatting for JSON files in the source editor. (31758362)
- Fixed an issue where code completion in Swift may produce incorrect results when completing within an expression. (40384814)
- Fixed an issue where clicking to the immediate left of the first column of text would place a breakpoint when line numbers were disabled. (41352438)

Interface Builder

- Color attributes in the user-defined runtime attributes table can be edited. (40035743)
- Custom fonts are supported in watchOS dynamic notifications. (35843402)
- Fixed an issue with rendering table view cells that are scrolled out of view at design time. (41213048)
- When designing Today widgets for Notification Center on macOS 10.14, the appearance now matches the canvas correctly, instead of always being dark. (41282414)
- Fixed an issue where clicking a build issue wouldn't select the referenced object in the canvas. (14779509)
- Opening the object library for macOS is faster. (40559699)

Asset Catalog

- Auto scaling is no longer turned on by default in asset catalogs for watchOS PDF assets, regardless of selection. (35788204)

Debugging

- When view debugging a SceneKit scene the inspectors will display the properties for the selected scene node. (39801376)
- Fixed a crash that occurred while stopping a Pixel Shader Debugger session, or when reloading a shader while in a Shader Debugger session. (41508020)

Signing and Distribution

- Enabling the Hardened Runtime capability in macOS will not cause Ad-Hoc signed apps containing Swift code to crash on launch. (40571112)

Instruments

- The event-type field of os-signpost point events in Instruments appears as "Event". (40591556)
- Fixed an issue where breakpoints could not be added or modified by command clicking when Xcode was in the background. (37835585)

New in Xcode 10 beta 4 – Swift and Apple Clang Compilers

Swift Compiler

- The behavior of casts from `Optional` types to generic types has changed in some circumstances in Swift 4.2. For example in:

```
struct ConditionalCast<T> {
    func cast(value: Any?) -> T? {
        return value as? T
    }
}
```

  If value is `nil` and the type `T` is an `Optional` type at runtime, the return value will now be `.some(nil)` rather than `nil`. This behavior is consistent with the casting behavior when concrete types are used rather than generic types. (40916953)

Swift Package Manager

- The scheme generation logic is improved and generates schemes as follows:
  - One scheme containing all regular and test targets of the root package.
  - One scheme per executable target containing the test targets whose dependencies intersect with the dependencies of the exectuable target. (30955712)

Resolved in Xcode 10 beta 4 – Swift and Apple Clang Compilers

Apple Clang Compiler

- The debugger correctly displays global variables that are Foundation value types. (39722386).
- The performance of the first Swift expression evaluation has been optimized when a large number of static swift modules are present (eg. Static CocoaPods). (40905971)
- Clang now correctly handles invalid Objective-C properties that could affect code completion support. (33761186)

Swift Compiler

- An Objective-C method that takes a block argument annotated with `__attribute__((noescape))` invoked by Swift, no longer raises an incorrect "closure argument passed as @noescape to Objective-C has escaped" runtime error. (40857699)

Swift Migrator

- The Swift 4.2 migrator correctly updates member variable references to global variable references. (41658300)

New in Xcode 10 beta 3 – IDE

Source Editor

- You can add selections for the next and previous find results using the “Find and Select Next” and “Find and Select Previous” menu commands.

Additionally, you can quickly add selections for the next and previous occurrences of the current selected text using the “Select Next Occurrence” and “Select Previous Occurrence” menu commands. (39283721)

- Moved some menu items into a new Editor > Selection submenu. Added new “Select Column Up” and “Select Column Down” items to allow creating columnar selections with the keyboard. (39321297)

Interface Builder

- The new Volume control for watchOS 5 now allows setting tint color. (40830006)
- NSVisualEffectView will backwards deploy new standard semantic 10.14 materials to appropriate materials for prior OSes. (40511407)
- NSVisualEffectView’s inherited appearance inspector property has a backwards deployment option of Vibrant Light or Vibrant Dark for prior OSes. (40536518)

Signing and Distribution

- When developing on macOS 10.13.6 beta 3 or later, the Hardened Runtime capability is available, and codesign supports passing the runtime flag. (39886423, 39886414)

Create ML

- Training can be performed on a subset of features in a given `MLDataTable`. (41255835)

Resolved in Xcode 10 beta 3 – IDE

General

- Fixed an issue where insertion point cursor was not visible in code snippet editor. (40347074)

Build System

- The command timings listed in Xcode's build log will list accurate times for each Swift subtask when using Multi-File Mode compilation. (40424339)

Testing

- In the destination configuration options for an Xcode Server bot, the "Run tests in parallel" option has been renamed "Test on the selected destinations concurrently". This option causes the selected destinations to run the scheme's tests simultaneously. It does not cause an individual destination to run tests in parallel. To allow each destination to run tests in parallel, deselect this checkbox and either enable parallel testing in the scheme or add `-parallel-testing-enabled YES` as an xcodebuild argument. (39985444)

- Fixed an issue causing baseline measurement changes to be lost and future measurements to report as %inf when editing those measurements on a system with a locale that does not use dots as decimal separators. (31813795)

- Xcode no longer crashes during UI test recording when interacting with content inside of WKWebView and SFSafariViewController. (33593609)

- Tests using a third-party testing framework that integrates with XCTest now work properly when run in parallel. (40939090)

Source Editor

- Fixed an issue where the source editor wouldn't update when editing a line with folded code. (40314098)
- Improved the performance of Undo and Redo after doing a Replace All. (40565102)
- Fixed an issue that caused Xcode to crash after deleting code. (40817677)
- Fixed a problem where code completion would fail for Swift source files included in multiple targets. (40909436)
- Fixed an issue that caused Xcode to crash when doing cmd + right arrow and cmd + shift + right arrow on a line with folded code. (41143589, 40431777)
- Fixed a problem with the Kill and Yank commands where some deleted text would not get added back with Yank. (40464103)
- Fixed a problem where semantic coloring would be lost after multi-cursor editing commands. (40710839)
- Fixed an issue where using the "noexcept" keyword in C++ source files would cause functions to not show up in the Jump Bar. (13266374)
- Resolved a crash that could occur when editing markup files. (36566474)
- Resolved an issue where attempting to show the Action Menu would fail immediately after displaying the Find and Replace control. (40355913)
- Fixed an issue where code completion for code snippets in CLIPS source files did not work. (40260325)
- Fixed a problem where double-clicking UI Test recording tokens deleted the text instead of confirming the selection. It now properly replaces the token with the selected item. (33573785)
- Resolved an issue where the scroll position may change slightly when switching files. (39100209)
- Resolved an issue where the scroll position may jump when scrolling through a file with very long lines. (40252210)
- Resolved an issue where diagnostics may re-appear after being resolved. (40295665)
- Resolved an issue that would cause Xcode to hang when unlocking a file. (40072075)

Refactoring

- Fixed a problem where the editing field of Refactor > Rename would not get keyboard focus if the "Reduce motion" Accessibility option was turned on. (40719848)

Interface Builder

- NSGridView has improved resizing when multiple columns are set to Automatic width. (40903162)
- Fixed a problem with scenes drawing blank when table view cells are scrolled beyond the UITableView bounds. (39338152)

Asset Catalog

- When importing an `ARReferenceObject` into the asset catalog, the preview image is shown. (39959348)
- Asset catalog compilation no longer produces an error when both Universal and Mac dark slots are defined for a color. (39506690)
- Dragging an `ARReferenceObject` to the empty region in the Asset catalog imports the asset. (40379198)
- Fixed an issue that prevented compiling an asset catalog with an empty `ARReferenceObject` slot. (40382246)

Debugging

- The items in the view hierarchy will appear at the correct z position in the view debugger. (39483664)
- Resolved an issue that would cause the view hierarchy to not render if the canvas area was scrolled before presenting the view hierarchy. (27635197)
- Resolved an issue where clicking on a frame in the Debug Navigator would not scroll to the corresponding line. (40559599)

Playgrounds

- macOS Single View playgrounds execute. (40533054)
- Resolved an issue that would cause playgrounds to fail to run, presenting a "couldn't lookup symbols" error message. (38505726)

Create ML

- Resolved an issue where renaming an image classifier in the UI and then dragging out an mlmodel might not use the specified name. (40535700)
- Fixes an issue where trying to train an MLImageClassifier on macOS Mojave Beta 2 would fail with the error: "Detected inf/nan values in feature(s) '__image_features__', '__image_features__', '__image_features__', ..." (41065289)

Server

- Certain Xcode Server logs, such as trigger logs, are now visible in the integration Logs report. (40462372)

Devices

- Fixed an issue causing the Connect via Network checkbox to become unchecked when first enabling network development for a device. (36797900)
- Unpairing a network device from Xcode will disable network development between the device and your Mac. (39226136)

Localization

- Xcode now extracts the appropriate values of the following Info.plist keys for localization: UTExportedTypeDeclarations, UTImportedTypeDeclarations, UIApplicationShortcutItems, INAlternativeAppNames. (40853982)

Signing and Distribution

- Xcode will wait for a simulator to be available before attempting to run an app on that simulator. (39116182)

Instruments

- os_signpost and os_log recording includes data immediately if recording from a device that has rebooted recently. (40654502)
- Instruments' Leaks tool and Xcode's memory debugger now more reliably runs on Swift projects. (40826018)

Simulator

- Executing `/path/to/Xcode.app/Contents/Developer/usr/bin/simctl list` will show devices for the Xcode from which `simctl` is being run. (41008319)
- Performing a scroll gesture while the mouse pointer is inside a watchOS simulator will scroll the digital crown. This makes bezel and non-bezel behavior identical. (40691638)

New in Xcode 10 beta 3 – Swift and Apple Clang Compilers

Apple Clang Compiler

- Xcode 10 adds support for the C++17 headers `<any>`, `<optional>`, and `<variant>`. (39271859)

Swift Package Manager

- The generate-xcodeproj has a new `--watch option` to watch the file system and automatically regenerate the Xcode project if needed. (32390792)
- SwiftPM now supports declaring dependency on a package using its path on disk instead of the git URL. This requires updating the package's tools version to 4.2. (39418745)
- The PackageDescription API in tools version 4.2 supports a new type of target "system library target", which moves the current system-module packages feature from package to target level. (39418939)

Resolved in Xcode 10 beta 3 – Swift and Apple Clang Compilers

Apple Clang Compiler

- The Swift REPL now successfully imports frameworks. (40458863)

Swift Compiler

- `type(of: object)` now provides the expected result when `object` is being observed by KVO. (37319860)
- The compiler now correctly emits code that references the type metadata for a private or internal type that should not be accessible, e.g., because it's from another module or (in non-WMO mode) source file. The `-emit-public-type-metadata-accessors` compiler flag that was used as a workaround will no longer be available at the end of the beta period. (39763787/40229755)
- Swift 4.2 migrator no longer unnecessarily inserts base names where dot-member access is sufficient. Base type names are still inserted if the member names have been changed in a newer SDK, because of uncertainty whether the contextual types still stay the same. (40373279)
- Swift 4.2 migrator no longer unnecessarily inserts a pair of conversion functions to an expression so that the type of the expression under migration stays the same. (40979943)
- Assigning between properties of a value of protocol type no longer raises inout exclusivity errors in some circumstances. (39524104)
- The Swift REPL shipped in the Command Line Tools package now successfully imports Objective-C frameworks or Swift frameworks depending on Objective-C frameworks. (40537961)
- `os_signpost` and `os_log` have been updated to not have a named `type:` parameter. (40528229)
- The `zero` property on `UIEdgeInsets` can now successfully be used. (40735990)
- Functions and methods that have exactly two overloads, where both overloads are generic, now selects the same overload during type checking as in prior releases. (40819547)

Resolved in Xcode 10 beta 2 – IDE

Build System

- The “Clean Build Folder” command will clear errors and warnings. (39659924)
- The `Clean Build Folder` action is available for projects which use Custom or Legacy build locations. (40108679)
- Previously broken input references will not cause new build system to generate spurious “Build input file cannot be found” errors on subsequent builds. (40667135)
- `xcodebuild clean` deletes all build products. (40460666)

Testing

- Resolved an issue on macOS 10.14, where running macOS UI tests from Xcode would result in a password prompt being shown for allowing the debugger to attach to the test runner. (40530732)
- Resolved an issue that would cause Simulator to hang after the OS had finished booting when tests were run on an iOS or tvOS simulator. (40147579)

Source Control

- Fixed a possible “Use of uninitialized value” error when using `git-svn`. (40651114)
- `git` has been updated to version 2.17.1 to address CVE–2018–11233 and CVE–2018–11235. (40651217)

Interface Builder

- Resolved an issue macOS 10.13 where attempting to drag an `NSTextView` object from the library to the canvas would crash Xcode. (40529925)
- Resolved a crash that occurred when hovering over a watchOS notification category in the document outline. (40561230)
- Resolved a runtime crash on iOS 10 or earlier for apps with a constraint between a `UIScrollView`’s layout margin and one of its descendant views. (40540860)

Asset Catalog

- Resolved an issue causing certain image assets to not be included in compiled car file when deploying apps to iOS 11 or earlier. (40507731)

Debugging

- Resolved an issue requiring the user to explicitly use `xcode-select` in Terminal to select the right installed Xcode when using the GPU Frame Debugger to debug shaders of offline-compiled libraries with included sources. (40340429)
- The GPU Frame Debugger's Geometry Viewer does supports issue detection. (40224469)

Create ML

- Resolved an issue preventing evaluation from starting when dragging in a folder with fewer than half of the number of trained classes. (40558219)
- True labels properly appear while scrolling through evaluation results in the `MLImageClassifier` UI. (40534700)

SiriKit Intent Editor

- You can set a key image for an intent with no parameters. (40395951)

Signing and Distribution

- Resolved an issue causing `xcodebuild -exportNotarizedApp` to crash when checking the status of an archive. (40625436)
- When running Xcode on macOS 10.14 beta 2 or later, resolved an issue causing signing with a Developer ID certificate to fail with the error "signature size too large to embed". This can occur if you've enabled the Hardened Runtime capability in Xcode. (40665475)

Simulator

- Resolved an issue preventing iOS 9.2 and earlier simulator runtimes from functioning correctly on macOS 10.14 Beta. (40335891)
- Shortcuts are available on Simulator. (40380495, 40380729)
- Improved the diagnostic messages when attempting to installing an improperly constructed app to Simulator. (40505189)
- iOS 12's user interface no longer disappears when activating an external display with an iOS simulator including a CarPlay window, nor when switching external displays. (39728482)
- iPhone X simulators will no longer render all black on some 2012 Macs. (40741248)

Documentation Viewer

- Resolved an issue causing light font colors to be used when printing documentation viewer content on macOS 10.14 beta. (40274975)

New in Xcode 10 beta – IDE

General

- Library content has moved from the bottom of the Inspector area to an overlay window, which can be moved and resized like Spotlight search. It dismisses once items are dragged, but holding the Option key before dragging will keep the library open for an additional drag.

  The library can be opened via a new toolbar button, the View > Libraries menu, or the ⇧⌘L keyboard shortcut. Content dynamically matches the active editor, so the same UI provides access to code snippets, Interface Builder, SpriteKit, or SceneKit items. The media library is available via a long press on the toolbar button, the View > Libraries menu, or the ⇧⌘M keyboard shortcut. (37318979, 39885726)

- Custom code snippets can now be added to the library via the Editor > Create Code Snippet menu item. (37946810)

- Newly-created schemes are now shared by all users of an Xcode project. To create a personal scheme, uncheck the “Shared” checkbox in the “Manage Schemes” sheet. (40223696)

- The Capabilities tab in the Project Editor provides a new Hardened Runtime capability for macOS apps and app extensions. Enabling this capability will opt your app into new security protections provided by macOS 10.14 and will be required for your app to be notarized. (39674498)

- Select Schemes and Run Destinations from the keyboard. Press “Ctrl+0” to open the Scheme popup and “Ctrl+Shift+0” to open the Run Destination popup. Once the popup appears, type enough characters to highlight the appropriate entry, use the arrow keys to highlight it, and press return to select it. (8999215)

Source Editor

- The Xcode Source Editor now supports multi-cursor editing allowing you to quickly edit multiple ranges of code at once.

  You can place additional cursors with the mouse via ^+⇧+Click or with column select (⌥+Click+Drag), or with the keyboard using ^+⇧+Up to column select up or ^+⇧+Down to column select down. (12564506)

- Xcode has increased support for code folding, including:

  1. A new code folding ribbon showing all of the multi-line foldable blocks of code in the editor
  2. A new style for folded code in the editor that allows you to edit lines with folded code
  3. Support for folding any block of code enclosed in curly braces
  4. Support for folding blocks of code from the folding ribbon, from structured selection, or from the Editor > Code Folding > Fold menu item

  To turn on the code folding ribbon, open preferences under Text Editing > Editing and select "Code folding ribbon". (33518606, 34203382, 35391767, 35932817, 36554006)

- All instances of the current symbol can be selected in the editor with the Editor > Structure > Select All Symbols command. (35064345)

- With a source control-enabled project the source editor displays changes made by a developer in the gutter and shows changes made by other developers that haven't yet been pulled into the project. (9794871)

- Find results in the source editor can now be selected using the Find > Select All Find Matches menu item. The results to select can be constrained to those in the current selection using the menu item Find > Select Find Matches in Selection. (35064581, 39283557)

- The source editor will now automatically detect and use the predominant line ending style in a file. (35343242)

- Selected text ranges in the source editor can now be split into multiple selected text ranges by line using the Editor > Structure > Split Selection By Lines menu item. (39300660)

- The source editor now supports configurable overscroll at the end of the file. The overscroll amount can be configured in Preferences > Text Editing > Editor Overscroll. (9075043)

- Updated default source editor themes for light and dark, with improved colors optimized for contrast, and taking advantage of bold and italic font traits. (40036785)

Build System

- Xcode 10 uses a new build system. The new build system provides improved reliability and build performance, and it catches project configuration problems that the legacy build system does not.

  Although the new build system is highly compatible with existing projects, some projects may require changes due to the following:

  - The new build system has stricter checks for cycles between elements in the build in order to prevent unnecessary rebuilds.
  - It is an error for any individual file in the build to be produced by more than one build command. For example, if two targets each declare the same output file from a shell script phase, factor out the declaration of the output file into a single target.
  - If an output file which is generated by a shell script is used as an input elsewhere in the build (for example, to another shell script), then that output must be declared as an explicit output by the script that generates it; otherwise the build system may attempt to search for the file before it has been generated, causing the build to fail.
  - The legacy header map that was generated when the Always Search User Paths (`ALWAYS_SEARCH_USER_PATHS`) setting was `YES` is not supported by the new build system. Instead, set `ALWAYS_SEARCH_USER_PATHS` to `NO` and migrate to using modern header include syntax. Add any needed header files that are in the project repository to the Xcode project to ensure they are available for use in #include (via the project wide header map). Use quote-style include (`"foo.h"`) for project headers, and reserve angle-bracket include (`<foo.h>`) for system headers.
  - The new build system passes `undefined_arch` as the value for the ARCH environment variable when running shell script build phases. The value was previously not well defined. Any shell scripts depending on this value must behave correctly for all defined architectures being built, available via the ARCHS environment variable.
  - The new build system uses the “clean build folder” behavior. The legacy “clean” behavior is not supported.

  More details on resolving issues which the new build system detects can be found in Xcode Help.

  If required, the legacy build system is still available in Xcode 10. To use the legacy build system, select it in the File > Project/Workspace Settings sheet. Projects configured to use the legacy build system will display an orange hammer icon in the Activity View. (35267155)

- Xcode provides improved warnings and errors for dependency cycles and other build issues detected by Xcode’s new build system. (37370377)

- While a target is performing post-compilation build steps, such as linking and code signing, Xcode's new build system will begin compiling sources from targets that depend on that target, to improve build speed by using more of your available processor cores. This compilation will not start until any "Run Script" phases from the target have completed. (38772619)

- Run Script build phases now support declaring input and output files in a `.xcfilelist` file. This file should contain a newline-seperated list of the file paths for the inputs or outputs. Build setting references may be used in these paths. The path to a copy of the `xcfilelist` file with resolved build setting references will be provided in the `SCRIPT_INPUT_FILE_LIST_(#)` and `SCRIPT_OUTPUT_FILE_LIST_(#)` environment variables of the Run Script phase. Projects which use an `.xcfilelist` require Xcode 10. (39241255)

- A new Product > Perform Action > Build With Timing Summary command produces a build log with statistics about the aggregate time taken by each command in the build. These timings aid in analyzing the time taken by the build. (40069871)

- When an `.xcconfig` file contains multiple assignments of the same build setting, later assignments using using `$(inherited)` or `$(<setting_name>)` will inherit from earlier assignments in the `.xcconfig`. The legacy build system caused every use of `$(inherited)` or `$(<setting_name>)` skip any other values defined within the `.xcconfig`. To detect whether your `.xcconfig` is affected by this improvement, running `defaults write com.apple.dt.XCBuild EnableCompatibilityWarningsForXCBuildTransition -bool YES` in Terminal will cause Xcode to generate a warning about this situation. (40283621)

Testing

- Test bundles can now be configured to execute their contents in a random order each time the tests are run. This can be enabled in the Test pane of the scheme editor as an option for each test bundle. (11719679)

- Xcode 10 supports running tests in parallel, which reduces the time it takes to run tests. Test parallelization is supported for macOS unit tests, as well as unit and UI tests on iOS and tvOS simulators. To enable parallelization, navigate to the scheme editor (Product > Scheme > Edit Scheme), select the Test action followed by the Info tab, and then next to your test target, click Options. Finally, select "Execute in parallel" (for macOS tests) or "Execute in parallel on Simulator" (for iOS and tvOS tests).

  Test parallelization occurs by distributing the test classes in a target across multiple runner processes. Use the test log to see how your test classes were parallelized. You will see an entry in the log for each runner process that was launched, and below each runner you will see the list of classes that it executed.

  When testing in parallel on Simulator, each runner process executes on a separate clone of the selected simulator. For a simulator named "iPhone X", these clones will appear in Simulator as "Clone 1 of iPhone X", "Clone 2 of iPhone X", etc. (35224733)

- `xcodebuild` has new command line options to control the behavior of parallel testing. Use `-parallel-testing-enabled` to override the per-target setting in the scheme for whether parallelization is enabled. If you want to control the number of runners that are launched, use `-parallel-testing-worker-count` or `-maximum-parallel-testing-workers`. (39648990)

- XCTest UI tests now capture the standard output and error streams from launched target apps to a file in the result bundle. (30929875)

- Schemes can now be configured to only run a fixed subset of tests in a test bundles and not automatically include new tests that are added to the bundle. This can be configured in the Test pane of the scheme editor as an option for each test bundle. (35050431)

Source Control

- Integration with Bitbucket Cloud and Bitbucket Server source control. (31156776)
- Integration with GitLab.com and GitLab self-hosted source control. (37501192)
- SSH keys can be easily managed using the new create, validate and upload workflows with GitHub, Bitbucket, and GitLab. (31798220)
- Improved source control authentication workflows provide more information and greater control to perform an authenticated operation. (33726987)
- Rebase support when pulling source control changes. (8937399)
- Tags can optionally be pushed from the push sheet. (40141815)

Interface Builder

- Canvas rendering now happens in parallel, and scales based on demand and hardware to improve the performance of edit operations, particularly for large scenes. (36999759)
- The “Stack” button in the canvas bar has been replaced with a pop-up menu containing all embedding options for the selection. (38201831)
- Some binaries used by Interface Builder have been renamed for better consistency. Instead of “Interface Builder Cocoa Touch Tool”, “Interface Builder WatchKit Tool”, and “IBAppleTVTool”, they are now called “IBAgent-[platform]”. (35400833)
- Opening a .nib or .xib file last saved in a format prior to Xcode 5 now prompts for confirmation, and automatically upgrades the document to a modern format. (37691996)
- The Editor > Embed In menu item allows embedding in a tightly-wrapped view that doesn’t add any margins to the selected content. (11515741)
- Resizing a `UIStackView` or `NSStackView` along its distribution axis in the canvas will adjust spacing between items. (21166308)
- The menu for choosing a font family in the Attributes inspector now renders a preview of each font. (37157540)
- Support for Large Title font text style (`UIFontTextStyleLargeTitle`) on iOS 11 or later, and on watchOS 5. (33150633)
- Image and color inspector properties that reference an asset catalog resource have a navigation button to jump to that resource. Option + click will show the resource in the Assistant Editor. (39661055)
- Controls using named colors from an asset catalog now update as the value of the color changes. (37613468)
- Inspector color pickers now use a standard pop-up button. The color swatch can be dragged by holding the Option key. (38582636)
- In the Identity inspector, the localization comment property has been changed to a plain string. It is still stored in the document as an attributed string to maintain compatibility with earlier versions of Xcode. (39856810)

macOS SDK

- Support for `NSGridView`, which manages Auto Layout constraints for its content in a two-dimensional table layout. `NSGridView` is backward deployable to macOS 10.12, or macOS 10.13.4 when using merged cells. (23707607)
- On macOS 10.14, the Device Bar and Preview assistant editor allow designing for and previewing in Light Appearance and Dark Appearance. (38673229, 38940117)
- Support for new `NSVisualEffectView` materials introduced in macOS 10.14. (37688179)
- Designable `NSView` subclasses are rendered with the appropriate `NSAppearance` for their use in the canvas. (39021773)
- `NSTextView` has multiple entries in the object library. This separates configurations appropriate for user interface elements that should adapt to the system appearance, and document content which has a fixed appearance that does not change with the system setting, including for print content. (39022169)
- The default colors for `NSTextField` and `NSTextView` have been changed. The new values are still appropriate for deployment on macOS 10.13 and earlier. (39577751)
- When a Mac XIB or storyboard is opened in Xcode 10, `NSTextView` that have default colors will be changed to use new recommended colors. The text color and insertion point color change from black to `NSColor.textColor`, and the background color changes from white to `NSColor.textBackgroundColor`. These should appear the same on earlier releases while also adapting to appearances on macOS 10.14. Editable NSTextField instances have their text color changed from `NSColor.textColor` to `NSColor.controlTextColor`. These changes only apply when opening and saving the document from within Xcode. (40280823)
- On macOS 10.14, `NSButton` and `NSImage` support content color tinting in the Attributes inspector. Tinting only applies to borderless buttons and template images. (39957802)
- On macOS 10.14, `NSToolbarItem` can center itself in an `NSToolbar` by checking “Is Centered Item” in the Attributes inspector. (38639941)
- An `NSToolbarItem` containing an image or custom view has an Automatic size option in the Size inspector. Use this option to automatically manage the minimum and maximum width and height of the toolbar item, for example when the item contains a segmented control. (38810182)
- Support for setting accessibility attributes on `NSWindow`. (9929792)

watchOS SDK

- Support for the new Dynamic Interactive notification interfaces. (36801793)
- Support for the new “Now Playing” and “Volume” interface objects. (36824666, 36824736)
- Notification categories now have a “Handles Grouping” property, which allows notifications to be coalesced. (38503942)
- Static notification interfaces now support connecting a subtitle label outlet. (39536566)

iOS SDK

- In iOS 12, the default value of the `UITableView` property `cellLayoutMarginsFollowReadableWidth` is now false. This means by default, table view cells will no longer have their layout margins increased automatically when the table view is wider than the maximum readable width.

  If you are using a table view to display text that spans multiple lines, in order to improve readability when the table view is very wide you should opt-in to these automatic margins by setting `cellLayoutMarginsFollowReadableWidth` to true, or consider adapting your layout to better take advantage of the additional horizontal space. (36828516)

tvOS SDK

- New `UILabel` inspector property “Marquee on Ancestor Focus” for controlling scrolling behavior. (31222028)

Asset Catalog

- Support for varying image and color assets by Light, Dark, and High Contrast appearances on macOS 10.14 and above. (38735965)
- Support for CarPlay assets. (35543584)
- Support for ARKit 3D `ARReferenceObject` assets. (38086640)
- The background of the asset catalog and view debugger can be set explicitly to light or dark so foreground elements display with sufficient contrast. (39073926)

Debugging

- The files table in the Disk Gauge Report includes files that were closed so that developers are aware of their size. (39359014)
- The Disk Gauge Report includes a new table that shows the total Read and Write sizes of the process across time. (39359147)
- The debugging Attach sheet allows developers to turn on Malloc Stack Logging (Live Allocations). (35447219)
- Named colors shown in the inspector while view debugging now indicate their names and whether they are system colors. (38193961)
- The view debugger adds `appearance` and `effectiveAppearance` properties to the `NSView` inspector. (38198002)
- Xcode's view debugger adds an option to choose between light and dark canvas background color. (39159102)
- You can change the appearance of your macOS app at runtime by using the Debug > View Debugging > Appearance menu, the Override Appearance menu in the debug bar, or the touch bar. (39448599)
- Metal Frame Debugger (33194071)

  Xcode 10's Metal Frame Debugger lets you debug your Metal shaders. Capture a Metal workload, select a draw / dispatch call and click the new debug button on the debug bar to start debugging your shaders.

  Selecting command encoder items in Debug Navigator will bring up a graph showing how encoders depend on each other through resource usage. Zoom in to get more information about attachments and encoders.

  Every draw call has new "Geometry" item where you can see 3D visualization of your post transform vertex data alongside with your vertex inputs and outputs.

Playgrounds

- Xcode playgrounds now execute code incrementally, compiling new lines of code when you hit shift-return or click the Run button next to the new line of code. This is especially useful for long tasks that you don't want to run repeatedly, like training a machine learning model or setting up your live view's state, and allows you to progressively iterate on your ideas without restarting the playground. (34313149)

Localization

- Xcode will offer to enable Base Internationalization for projects that do not yet use it but support multiple localizations. (13178091)

- Xcode 10 supports a new Xcode Localization Catalog (xcloc) export/import format for localization data that can contain both an XLIFF file and other localizable content such as image files. (28662326)

Metal

- When calling the `metal` compiler directly you must use the `-c` flag when building a `.metal` file. Not adding the option will result in an error when the resulting `.air` file is consumed by `metallib`.

  Calling the `metal` compiler via Xcode's build system will add the flag automatically. (40655432)

Signing and Distribution

- The Developer ID distribution option in Xcode's Organizer now provides support for uploading apps to Apple to be notarized. After building an archive, this option can be selected in the Organizer by clicking the Distribute App button and then selecting the Developer ID method and the Upload destination. In order to upload an app to be notarized, you must enter an Apple ID in Xcode's Accounts preferences pane with the necessary App Store Connect role and provider membership. In addition, apps uploaded to be notarized must be signed with a Developer ID certificate. The distribution workflow can create this certificate if necessary, but requires an Apple ID account with the Agent role in order to do so.

  After uploading an app to be notarized, you can view your app's status in the Organizer window by selecting your archive and clicking the Show Status Log button. When you receive notification that your app has been notarized, you can export it from the Organizer window by selecting your archive and clicking the Export App button. The exported app contains a stapled ticket and is ready for distribution. (36409604)

- Support for uploading apps to Apple via the command line. The `xcodebuild -exportArchive` command will perform an upload if the provided `ExportOptions.plist` contains a key named "destination" with value "upload". In addition, an Apple ID account with the necessary App Store Connect role and provider membership must be added in Xcode's Accounts preference pane. The "app-store", "developer-id", and "validation" distribution methods are supported for use from `xcodebuild`. (28555930)

Instruments

- You can now build and distribute your own custom instruments. (37987666)
- The support for system trace instrumentation prior to Instruments 8 has been removed. Some trace files containing data from these instruments will no longer open in Instruments. A version of Xcode prior to Xcode 10 can be downloaded from developer.apple.com in order to view these older files. (38506479)
- There is a new graph mode available for threads when using System Trace instrumentation in Instruments 10 – Thread States – which shows only the thread state lane. The previous OS Fundamentals graph style (system calls, VM faults, and thread state lanes) is still available as well. (37727022)

Documentation Viewer

- Quick Help has been updated to use a single column layout, making better use of available space that it is presented in. (39518057)
- Code listings in the documentation viewer and Quick Help match the user's currently selected theme colors. (39435799)

Resolved in Xcode 10 beta – IDE

General

- The New File template for Objective-C header (.h) files includes `NS_ASSUME_NONNULL_BEGIN/END` macros. (22753521)
- Project settings validation understands build setting values set by reference to other build settings. (30549576)
- Double Click Navigation now defaults to Same as Click. This can be changed in the Navigation pane of Preferences. (37294346)
- Schemes for which “Show” has been unchecked in the Manage Schemes sheet no longer appear in the Product > Scheme submenu. (24579413)

Source Editor

- Improved the performance of many editing operations, including Undo and Redo. (30874294)
- Fixed a problem where the “Automatically balance brackets in Objective-C method calls” preference was not respected. (34195824)
- Live issues will now be cleared when renaming a file. This resolves an issue where live issues may remain after a file has been renamed and edited. (36186038)
- Fixed a problem that would cause wide content in the “Jump to Definition” popover to be clipped. (39060697)
- Resolved a crash when editing a file using Classic Mac OS (CR) or Windows (CRLF) line endings. (39137414)
- Resolved an issue where attempting to present the Action menu immediately after selecting the next find result may fail. (39884637)
- Fixed a problem with code completion using code snippets starting with @ character, such as @autoreleasepool. (31768452)
- The source editor now provides more accurate code completion when typing initializers in Swift. (38265505)
- Enhanced reliability of cmd+click jump to definition, now showing results more reliably in situations with syntax errors or other errors in the code. (34905255)
- Resolved an issue where breakpoints may appear at the incorrect line in folded code. (37288192)
- Fixed crash of image literal quick editor. (38281672)
- Fixed problems where the “Insert Newline without Extra Action” and “Insert Newline and Leave Selection Before It” commands didn’t work correctly. (38477237)
- Improved the scrolling estimation for large files. (39344677)
- Fixed a problem with occasional hang/spin during code completion. (40088929)
- Fixed an issue where comments in XML files that contained MARK: or TODO: would cause syntax coloring of the rest of the file to fail. (12051726)

Testing

- Xcode and xcodebuild now build all targets in the active scheme when clicking a test gem in the Test navigator or source editor (Xcode), or when using `-only-testing: (xcodebuild)`. Previously, only the relevant test bundle target and its dependencies would be built. Xcode and xcodebuild will now also disallow you from running a test or test class whose test bundle is not part of the active (Xcode) or specified (xcodebuild) scheme. (38935442)

- Errors that may lead to missing or no coverage data are now displayed in the test log. If you notice any such errors, please file a bug at https://bugreport.apple.com, and attach a zipped archive of your project's folder in DerivedData. (39930570)

Source Control

- The source control authors editor submode performance is improved to load and scroll faster. (40179372)

Interface Builder

- Typing delete (⌫) when an edge is selected in the constraint filter of the Size inspector now removes the applicable constraints, rather than the view itself. (25373426)

- Corrected an issue where previews and drag images did not always render at the correct scale factor if multiple displays with different scale factors were connected. (38775753)

- Fixed an issue where connect-to-source mistakenly allowed adding code to Swift generated interfaces. (35263074)

- When a default color is selected in the inspector, the name of the color is shown in addition to the text "Default". (39663381)

- Improved performance compiling documents, particularly those containing multiple `NSComboBox` controls. (30468986)

Asset Catalog

- Editing fractional color values in an asset catalog works correctly for non-English locales. (39666878)

Debugging

- Options in the Debug > View Debugging menu, like “Show View Frames”, now continue the target application as expected. (26649378)
- The Memory Gauge Report now reports the physical footprint consistently across all the platforms. (20472364)

Playgrounds

- A crash that sometimes occurred when creating a new playground page, especially if the playground was locked, has been fixed. (38281509)
- Code completion is now supported in auxiliary source files in playgrounds. (34363732)
- A common crash that occurred in playgrounds with inline results has been fixed. (38281379)
- Resolved an issue that prevented the iOS Game playground template from working correctly. (38828600)

Localization

- In an XLIFF produced by the Export For Localization command, content from a file outside the project directory will be referenced via a path relative to the project directory rather than an absolute path. (38680116)

Instruments

- Resolved an issue that would cause most templates to fail when profiling on iOS versions prior to iOS 11.3. (39759266)

Simulator

- Simulator now supports a "Point Accurate" snap-to scale factor in addition to the existing "Physical Size" and "Pixel Accurate" options. (38232400)
- "Pixel Accurate" scaling is now correct for tvOS Simulator devices. (36341177)
- CarPlay is now supported in the iOS Simulator. (37416934)
- Copy/Paste in Simulator's Edit menu is no longer used for synchronization with a simulator devices's pasteboard. The Edit menu now has explicit menu items to handle these actions. (38225290)
- Rendering is now always done at full resolution. As a result, screenshots are always at screen size. Simulator's "Optimize Rendering for Window Scale" option has been removed. (38232185)

New in Xcode 10 beta – Apple Clang and Swift Compilers

Apple Clang Compiler

- The static analyzer is more efficient and will report additional issues on most programs. (36672459)

- The static analyzer checks for a common performance anti-pattern when using Grand Central Dispatch, which involves waiting on a callback using a semaphore:

```
+ (NSString *)requestCurrentTaskName {
    __block NSString *taskName = nil;
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    NSXPCConnection *connection =
        [[NSXPCConnection alloc] initWithServiceName:@"MyConnection"];
    id remoteObjectProxy = connection.remoteObjectProxy;
    [remoteObjectProxy requestCurrentTaskName:^(NSString *task) {
        taskName = task;
        dispatch_semaphore_signal(sema);
    }];
    dispatch_semaphore_wait(
        sema,
        dispatch_time(DISPATCH_TIME_NOW, 100)
    );
    return taskName;
}
```

  Such a pattern can degrade performance and cause hangs in your application. The check is currently disabled by default, but can be enabled using the build setting "Performance Anti-Patterns with Grand Central Dispatch". (37312818)

Swift Language

- The new `CaseIterable` protocol describes types which have a static `allCases` property that is used to describe all of the cases of the type. Swift will synthesize this `allCases` property for enums that have no associated values. (SE–0194) (17102392). For example:

```
enum Suit: CaseIterable {
    case heart
    case club
    case diamond
    case spade
}

print(Suit.allCases)
// prints [Suit.heart, Suit.club, Suit.diamond, Suit.spade]
```

- As part of fully implementing (SE–0054), `ImplicitlyUnwrappedOptional<T>` is now an unavailable typealias of `Optional<T>`. Declarations annotated with '`!`' have the type `Optional<T>`. If an expression involving one of these values will not compile successfully with the type `Optional<T>`, it is implicitly unwrapped, producing a value of type `T`. In some cases this will cause code that previously compiled to require updating. Please see this blog post for more information: (Reimplementation of Implicitly Unwrapped Optionals). (33272674)

- Public classes may now have internal `required` initializers. The rule for `required` initializers is that they must be available everywhere the class can be subclassed, but previously `required` initializers on public classes were forced to be public themselves. (This limitation was a holdover from before the introduction of the open/public distinction in Swift 3.) (22845087)

- The standard library now uses a high-quality, randomly seeded, universal hash function, represented by the new public `Hasher` struct. "Random seeding" varies the result of `hashValue` on each execution of a Swift program, improving the reliability of the standard library's hashed collections such as `Set` and `Dictionary`. In particular, random seeding enables better protection against (accidental or deliberate) hash-flooding attacks.

  This change fulfills a long-standing prophecy in `Hashable`'s documentation: *Hash values are not guaranteed to be equal across different executions of your program. Do not save hash values to use during a future execution.*

  As a consequence of random seeding, the elements in `Set` and `Dictionary` values may have a different order on each execution. This may expose some bugs in existing code that accidentally relies on repeatable ordering.

  Additionally, the `Hashable` protocol now includes an extra function requirement, `hash(into:)`. The new requirement is designed to be much easier to implement than the old `hashValue` property, and it generally provides better hashing. To implement `hash(into:)`, simply feed the exact same components of your type that you compare in `Equatable`'s `==` implementation to the supplied `Hasher`:

```
struct Foo: Hashable {
    var a: String?
    var b: [Int]
    var c: [String: Int]

    static func ==(lhs: Foo, rhs: Foo) -> Bool {
        return lhs.a == rhs.a && lhs.b == rhs.b && lhs.c == rhs.c
    }

    func hash(into hasher: inout Hasher) {
        hasher.combine(a)
        hasher.combine(b)
        hasher.combine(c)
    }
}
```

Automatic synthesis for `Hashable` (SE–0185) has been updated to generate `hash(into:)` implementations. For example, the `==` and `hash(into:)` implementations above are equivalent to the ones synthesized by the compiler, and can be removed without changing the meaning of the code.

Synthesis has also been extended to support deriving `hashValue` from `hash(into:)`, and vice versa. Therefore, code that only implements `hashValue` continues to work in Swift 4.2. This new compiler functionality works for all types that can implement `Hashable`, including classes.

Note that these changes don't affect Foundation's hashing interface. Classes that subclass `NSObject` should override the `hash` property, like before.

In certain controlled environments, such as while running particular tests, it may be helpful to selectively disable hash seed randomization, so that hash values and the order of elements in `Set`/`Dictionary` values remain consistent across executions. You can disable hash seed randomization by defining the environment variable `SWIFT_DETERMINISTIC_HASHING` with the value of `1`. The Swift runtime looks at this variable during process startup and, if it is defined, replaces the random seed with a constant value. (SE–0206) (35052153)

- The standard library now has a Random generator API. The Swift standard library now supports generating and using random numbers. Swift collections now support `shuffle`/`shuffled` operations as well as choosing a `randomElement()`. Numeric types can now generate a uniform value in a range e.g. `Int.random(in: 1...10)` or `Double.random(in: 0..<100)`, and `Bool` provides a random coin-flip: `Bool.random()`.

  A `RandomNumberGenerator` protocol has been introduced as a source of randomness to drive these methods, with a default implementation for each platform. On Apple platforms, this implementation wraps `arc4random_buf`. (40564129)

- Custom compile-time warnings or error messages can be emitted using the `#warning(_:)` and `#error(_:)` directives (SE–0196). (16015824)

```
#warning("this is incomplete")

#if BUILD_CONFIG && OTHER_BUILD_CONFIG
#error("BUILD_CONFIG and OTHER_BUILD_CONFIG can't both be set")
#endif
```

Swift Compiler

- The Swift compiler defaults to a new compilation strategy that can greatly speed up debug builds. This strategy allows each compilation job to process a batch of files instead of just a single file.

  Projects that had sped up debug builds by opting into building with Whole Module Optimization in Debug mode (at `-Onone`) should try the new compilation style. To do so, choose the Incremental option for the Compilation Mode build setting and ensure that your project is not using older whole-module settings for Debug builds such as `SWIFT_WHOLE_MODULE_OPTIMIZATION=YES` or `SWIFT_OPTIMIZATION_LEVEL=-Owholemodule`.

  The new compilation strategy should be transparent in terms of the build products. However, in case it causes a problem, you can revert to the previous single file strategy with a custom Build Setting `SWIFT_ENABLE_BATCH_MODE=NO`. (39253613)

- C macros containing casts are no longer imported to Swift if the type in the cast is unavailable or deprecated, or produces some other diagnostic when referenced. These macros were already only imported under very limited circumstances with very simple values, so this is unlikely to affect real-world code. (36528212)

- Swift 4.1 warnings about 'overlapping accesses' are now errors in Swift 4 mode. These new errors most commonly arise when a generic mutating method that modifies a variable is passed a non-escaping closure that reads from the same variable. Code that previously compiled with a warning will need to be updated. Please see this forum post for more information: (Upgrading exclusive access warning to be an error in Swift 4.2). (34669400)

- Various function-like declarations can now be marked as `@inlinable`, making their bodies available for optimizations from other modules. Inlinable function bodies must only reference public declarations, unless the referenced declaration is marked as `@usableFromInline` (SE–0193). (40566899)

  Note that the presence of the attribute itself does not force inlining or any other optimization to be performed, nor does it have any effect on optimizations performed within a single module.

- The C `long double` type is now imported as `Float80` on i386 and x86_64 macOS and Linux. The tgmath functions in the Darwin and glibc modules now support `Float80` as well as `Float` and `Double`. Several tgmath functions have been made generic over `[Binary]FloatingPoint` so that they will automatically be available for any conforming type. (SR–2046) (27196587)

Swift Standard Library

- `Range` is now conditionally a `RandomAccessCollection`. Through conditional conformance, the `Range` type is now conditionally a `RandomAccessCollection` when its `Bound` is `Strideable` with a `SignedInteger` stride. This means that you can now use use `Range` as a collection when these criteria are met. (40564272)

  `CountableRange` is now a constrained generic type alias for a `Range`, rather than an independent type. This means that you can still use `CountableRange` in the same ways as before: extend it, take it as an argument to a function, or declare a variable of that type. But you can also now use ranges and countable ranges interchangeably:

  ```
  let countableRange: CountableRange<Int> = 0..<10
  var range: Range = countableRange

  func f<T: SignedInteger>(_ countableRange: CountableRange<T>) { }
  f(range)
  ```

- The standard library types `Optional`, `Array`, `ArraySlice`, `ContiguousArray`, `Dictionary`, `DictionaryLiteral`, `Range`, and `ClosedRange` now conform to the `Hashable` protocol when their element or bound types (as the case may be) conform to `Hashable`. This makes synthesized `Hashable` implementations available for types that include stored properties of these types (SE0143). (40567748)

- The behavior of the `description` and `debugDescription` properties for floating-point numbers use a new algorithm that is both more accurate and significantly faster than the previous code. In particular, we now guarantee that `Double(String(d)) == d` for every Double value `d`. Previously these unconditionally printed a fixed number of decimal digits (e.g. 15 and 17 for Double, respectively). They now print exactly as many digits as are needed for the resulting string to convert back to the original source value, and no more (SR–106). (40565639)

Swift Package Manager

- As per SE–0209, `swiftLanguageVersions` property in `PackageDescription` manifest API for tools version 4.2 is changed from an array of Integers to an array of `SwiftVersion` enum. (38721967)

Example usage:

```
// swift-tools-version:4.2
import PackageDescription

let package = Package(
    ...
    swiftLanguageVersions: [.v3, .v4]
)
```

Resolved in Xcode 10 beta – Swift Compiler

Swift Compiler

- When a property satisfies a requirement in an `@objc` protocol, the selectors for its getter and setter are in all circumstances now inferred from the protocol requirement. (37363245)
- When switching over an `@objc` enum, if the value is not one of the recognized cases and the switch does not have a default case, the program will trap at run time. (20420436)
- The compiler now correctly handles when a nested type in a `Codable` type has the same name as a property of the `Codable` type (SR–6569). (37570349)
- Protocol conformances are now able to be synthesized in extensions in the same file as the type definition, allowing automatic synthesis of conditional conformances to `Hashable`, `Equatable` and `Codable` (both `Encodable` and `Decodable`). For instance, if there is a generic wrapper type that can only be `Equatable` when its wrapped type is also `Equatable`, the `==` method can be automatically constructed by the compiler:

```
struct Generic<Param> {
    var property: Param
}

extension Generic: Equatable where Param: Equatable {}
// Automatically synthesized inside the extension:
// static func ==(lhs: Generic, rhs: Generic) -> Bool {
//   return lhs.property == rhs.property
// }
```

  Code that wants to be as precise as possible should generally not conditionally conform to `Codable` directly, but rather its two constituent protocols `Encodable` and `Decodable`, or else one can only (for instance) decode a `Generic<Param>` if `Param` is `Encodable` in addition to `Decodable`, even though `Encodable` is likely not required:

```
// Unnecessarily restrictive:
extension Generic: Codable where Param: Codable {}

// More precise:
extension Generic: Encodable where Param: Encodable {}
extension Generic: Decodable where Param: Decodable {}
```

  Finally, due to `Decodable` having an `init` requirement, it is not possible to conform to `Decodable` in an extension of a non-final class: such a class needs to have any `init`s from

protocols be `required`, which means they need to be in the class definition. (SR–6803) (39199726)

- Runtime query of conditional conformances is now implemented. Therefore, a dynamic cast such as `value as? P`, where the dynamic type of `value` conditionally conforms to `P`, will succeed when the conditional requirements are met. (SE–0143) (40349058)

```
protocol P {}

struct DoesntConform {}
struct Conforms: P {}

struct Generic<Param> {}
extension Generic: P where Param: P {}

print(Generic<DoesntConform>() as? P as Any) // nil
print(Generic<Conforms>() as? P as Any)
// Optional(main.Generic<main.Conforms>())
```

- When a Swift method is exposed to Objective-C with a custom selector beginning with "new", "copy", or "mutableCopy", it will correctly return an owned (+1) value rather than an autoreleased (+0) one when called from Objective-C, as expected by Objective-C ARC. (SR–6065) (34834291)

- Properties in the header generated by the Swift compiler are now correctly annotated with OS availability information. (37090774)

- A conditional conformance does not imply conformances to any inherited protocols, unlike unconditional ones. This is because the optimal conditional requirements are likely to be different between the two protocols if the inherited protocol can use less restrictive requirements (SR–6569). For instance:

```
protocol Base {}
protocol Sub: Base {}
struct Generic<Param> {}

extension Generic: Base where Param: Base {}
extension Generic: Sub where Param: Sub {}
```

  The first extension has to exist, because the second one does not imply a conformance of `Generic` to `Base` (if it did, it would imply it with the same conditional requirements as the `Sub` conformance, that is, `Param: Sub`). This particularly affects hierarchies like `Hashable: Equatable`, and `BidirectionalCollection: Collection`, and helps achieve precision in the bounds of types conforming to them. (36499373)