Creating Accessible eCourses

eCourse and Storyline Checklist

September, 2019

**This document is intended to assist developers in the creation of eCourses that are accessible by University of California standards, with additional guidance provided for Articulate Storyline development. It reflects but does not represent official UC Electronic Accessibility policy or an official effort toward policy compliance. Similarly, it reflects, but is not intended to represent, Federal ADA or 508 law, or WCAG 2.0 standards.*

While the requirements and recommendations covered in this checklist can be applied to the development of any eCourse, Storyline-specific development guidance is provided in Appendix A: Storyline How-to Guides.

If you are unfamiliar with accessibility/Storyline accessibility basics, please review that section first.

Accessibility Standards Checklist/Table of Contents

[1] "Synchronized audio" is audio that has other events synched to it, such as animations, content reveals, etc. For instance, if a slide has a bullet list, the bullet list is hidden when the slide starts and each bullet item within the list is set to appear during a certain portion of the audio, the bullet list is synched to the audio making the audio "synchronized audio".

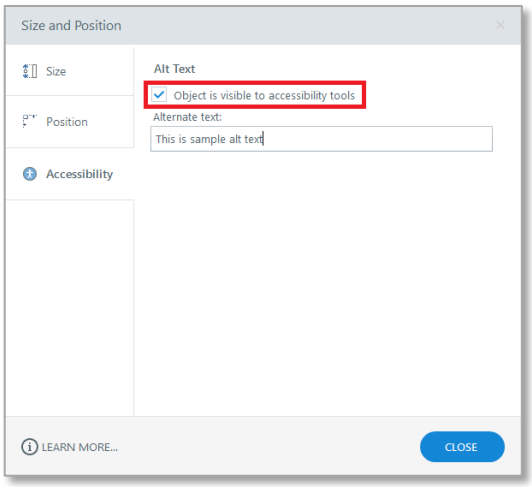
15. Provide all the necessary link information
16. Implement keyboard shortcuts to facilitate common in-course actions
17. Provide (accessibility) instructions
18. Allow users to easily restart a slide or slide content
19. Be aware of, and work around, bugs and other shortcomings that affect accessibility features
20. Bypassing browser-based manual play setting in Storyline (for pre-August 16, 2018 update outputs)

Accessibility and Storyline Accessibility Basics

Accessibility Tool/Assistive Technology Device Users

The terms "Accessibility tools" and "assistive technology devices" refer to the various mechanisms users with disabilities may employ to help them use a computer, navigate within a web browser and complete an eCourse. These mechanisms include the keyboard (used in lieu of a mouse), screen readers, speech recognition software and more. Within this training, users who employ these mechanisms will be referred to as "accessibility tool users."

Accessibility Tool Visibility



Accessibility tool visibility dictates whether an object can receive keyboard focus and be read by a screen reader.

It is most often controlled through the **Object is visible to accessibility tools** setting within the **Accessibility** tab of the **Size and Position** panel[2] (when the object you wish to affect is selected).

This setting can also be unselected from within the **Tab Order** panel by deleting an object's instance within that panel.

However, due to a current Storyline bug, you may experience inconsistencies between what accessibility tool visibility setting you think you've given an object and what you actually find when testing the slide or course. If you do encounter such inconsistencies, or if you want to ensure you avoid them, open **Edit States** mode and check the object's setting and alt text within its **Normal** state. The setting/alt text you find there may be different than, and will take precedent over, what you observed or established outside of the **Edit States** mode. In such cases, you

[2] Power-user notes: Use keyboard shortcut Ctrl+Shift+Enter to open the **Size and Position** panel.
Also, you do not need to close this panel to save setting changes made within it, and you can select other elements while it is already open. If you need to change multiple elements' accessibility tool visibility, alt text, size or position, save steps/time by keeping this panel open as you select and edit different elements.

need to make the non-state[3] and **Normal** state settings/alt text the same *and* to continue to maintain them in both places if future changes are made.

An element that is not visible, such as a shape with no fill or a 100% transparent fill, may still be visible to accessibility tools[4].

Keyboard and Tab Navigation

Users must be able interact with a course using only their keyboard (referred to here as "keyboard navigation"). In Storyline outputs, this is done through the Tab key, which is used to cycle keyboard focus forward through slide elements and the course player controls. Shift+Tab is used to cycle backwards.

Keyboard Focus

Keyboard focus[5] determines which object you can interact with or read using a screen reader. Keyboard focus is indicated in Storyline outputs with a yellow highlight around an object's entire boundary box.

Objects that are positioned outside of a slide but are still visible to accessibility tools may not show keyboard focus highlighting, but they will still receive keyboard focus and be read by screen readers.

[3] I.e., when you are not in **Edit States** mode

[4] These objects are handy when you need to provide screen reader-only information or keyboard navigation support. E.g., text links created by selecting text and using the Insert > Hyperlink feature don't work with keyboard navigation; instead, place a rectangle over the text and in the **Fill** tab of the **Format Shape** panel, set the fill color **transparency** to 100%. Note: a shape with no fill color (as opposed to a 100% transparent fill color) will support keyboard navigation but cannot receive a mouse click.

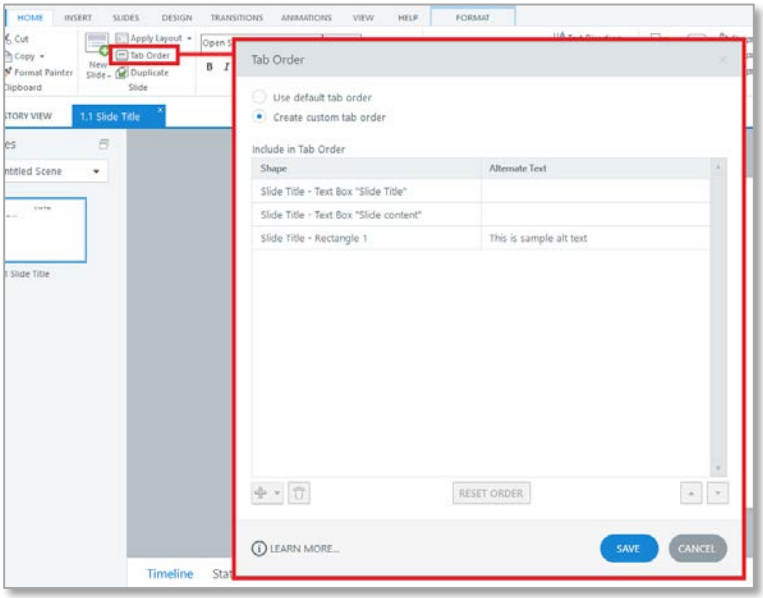
[5] The ability to receive keyboard focus is sometimes referred to as having, or being, a "tab stop," since the Tab key is used to cycle through these objects, and focus "stops" on an object after you Tab to it.

Tab Order

Tab order refers to the sequence of detectable objects within a slide.

By default, objects are ordered top to bottom, left to right, layers[6] then **base layer**[7], but this sequence can be re-ordered using the **Tab Order** panel, accessed via the **Tab Order** button in the **Home** ribbon.

Select the **Create custom tab order** setting to manipulate the order, then drag objects to their desired location within the order or select them and use the arrow buttons in the panel's lower-right corner.



Element Hierarchies

If features are available to program element hierarchies and relationships, they should be used in accordance with proper hierarchical structure[8].

Timing

Unless a time-limit is an essential feature of an exercise you're creating, avoid time-based requirements, such as requiring users to answer questions within a certain amount of time, since it can take more time to perform tasks using accessibility tools.

If a time-limit is an essential feature of an exercise you're creating, provide an alternative version that does not have a time limit.

Additionally, do not show/hide elements too quickly, as this can trigger certain conditions (e.g., photosensitive epilepsy) or cause important content to be missed.

Design and Visual Style

Certain design and visual style rules should be followed to ensure accessibility.

- Contrast: there must be sufficient contrast between background and foreground elements.

---

[6] In this guide, "layer(s)" is meant to describe a slide layer other than the base layer.
[7] In this guide, a slide's specific base layer will always be referred to as the "base layer".
[8] I.e., if an authoring program allows you to program Heading 1's, Heading 2's, Heading 3's, etc., you should do so in a way that is internally consistent and adheres to basic hierarchy tenants (e.g., Heading 3's reside within Heading 2's which reside within Heading 1's; not a Heading 1 that resides within a Heading 3 that resides within a Heading 4)

    - The WebAIM Color Contrast Checker can help you evaluate colors and offers guidance as to text size, color and contrast
- Size: text must be large enough to be easily readable. If possible, avoid using text smaller than 12 pt. 14 pt and 16 pt are preferable minimum sizes for large amounts of text.
- Conveying information with color: color can be useful for conveying information, but it can present a problem when it is the sole means for doing so since users may have color blindness[9]. Use other labels in addition to, or instead of, color to convey information.
- Moving text still needs to be read: make sure it moves slowly enough, or stays stationary and visible long enough, that *everyone* can read it. Consider including functionality that will pause the motion on user request so that the text can be read.
- Certain rapid and/or repetitive visual effects, like flashing text or a very quick slideshow, can induce photosensitive epilepsy and should be avoided.

Accessibility Standards for eCourses

Provide Alternative Format Accommodations

There is yet to be a perfect solution for making fully accessible eCourses that will satisfy everyone's needs, so it's best to create an alternative format version of a course during its initial development process[10] to accommodate users who prefer or need a different course experience.

A common alternative format is an accessible PDF. Word or Powerpoint files may also be an option as long as they're fully accessible. However, keep in mind that alternate format versions still need to provide the same learning experience, or as similar a learning experience as possible[11]. Some developers have even turned to Google Surveys to facilitate the end-of-course testing that often occurs within eCourses.

---

[9] According to the National Eye Institute, "as many as 8 percent of men and 0.5 percent of women with Northern European ancestry have the common form of red-green color blindness."

[10] As opposed to waiting for an accommodation request. Advantages of creating the alternative format accommodation during the initial course development include: a) having the same developer/knowledge team produce both versions of the course; and b) having the accommodation ready to go, so you're not trying to meet the mandated deadline of an unanticipated accommodation request while juggling other concurrent projects and deadlines.

[11] For example, multiple choice questions need to be delivered in the same sequence: you must give the question, then all the potential answers and *then* feedback as to which answers were right or wrong. You cannot provide each answer's feedback when you first introduce the answer choice, because that sequence does not allow learners the opportunity to evaluate all the answers before determining which are correct and incorrect.

Ensure Course Can Be Read by a Screen Reader

Test the authoring program output to ensure it and all its contents can be read by a screen reader in accordance with other basic guidelines[12] within this checklist.

If possible, test the output through a server, or a SCORM server, to simulate the end user experience as closely as possible[13].

Storyline

Both Flash and HTML5 outputs from Storyline 3 and 360 should be screen reader compatible, depending on which screen reader and browser are being used. Storyline 2's Flash outputs tend to be more compatible than their HTML5 outputs.

Articulate provides a test server — tempshare.articulate.com — that can be used to test published Storyline projects that include a Flash output (i.e., projects published **HTML 5 Only** cannot be uploaded to this server at this time). Granted, with Flash being phased out, you'll likely only be producing **HTML5 Only** Storyline outputs to upload to your LMS. However, before you do that, you can still publish an **HTML5 / Flash** version of your course and upload it to Articulate's tempshare server for testing, since, as long as the HTML5 output of your course opens from the tempshare server[14], the results you find in it will be nearly identical to what you can expect from an **HTML5 Only** version uploaded to your LMS.

Articulate states that Storyline 3 and 360 outputs are compatible with the JAWS screen reader and that they do not guarantee compatibility with other screen readers; however, if JAWS is unavailable, our own testing has found the NVDA screen reader to be decently reliable[15] within Storyline's preview mode and Chrome when testing through the Articulate tempshare server (NVDA still has issues in other browsers and in Chrome when viewing a course through our UC Learning Center LMS). Consider using NVDA for quick, mid-development testing and the JAWS demo[16] for final testing.

Other Authoring Programs

It's recommend that you personally test, or have UC services test, outputs from other programs, in multiple browsers, through the UC Learning Center LMS to ensure they can be read by screen readers.

---

[12] E.g., elements are read in the order you dictate; elements have the alt text you've assigned; etc.

[13] This is also important because test results may vary depending on the method of access. For instance, Storyline 2 outputs presented significantly different results when tested as a preview in Storyline, tested locally after publishing, and tested again when uploaded to and accessed from a server. In rare instances, Storyline 3 findings have similarly differed when tested on SCORM and non-SCORM servers.

[14] Check the course URL after it opens; if it contains "story_html5.html" near the end, it's the HTML5 version.

[15] NVDA may give you some "noise", such as adding "section" to the end of text frames, and it may have problems with templated interactions (including quiz slides) that JAWS does not, but overall it should be sufficient to test screen reader visibility (i.e., whether or not an element is visible to accessibility tools), content sequencing (tab order) and alt text.

[16] The JAWS demo requires you to restart your computer before use and only allows for 40 minutes of use before you have to restart again, so it is not ideal for quick testing.

Don't fully rely on the authoring program's self-proclaimed capabilities, as these can be misleading, vary by browser and/or screen reader, or fail to account for variables like SCORM or LMS hosting.

Ensure accessibility Tool Users Can Interact As Needed With the Course Content

Drag and Drop interactions, Hot Spot interactions, hover reveals (i.e., using the **Hover** state or **Mouse hovers over** triggers to reveal essential and/or non-decorative content) or any other sort of interactions that require mouse control and/or sight in order to complete them are inherently not accessible. Try to think of other user actions that are accessible that can accomplish the same intended purpose[17].

In general, it's recommended that you strive for "universal design"[18], but if an inaccessible interaction type is absolutely necessary, try to provide an accessible alternative[19].

Make sure to test courses using only keyboard navigation to ensure all aspects can be completed as intended.

Associated Storyline How-to Guide

- Fully Accessible Video Controls

Elements Are Purposefully Sequenced to Maximize the Learner/User Experience

Slide elements should be sequenced in the way that best enables accessibility tool users to understand the content and engage with interactive exercises.

---

[17] E.g., instead of revealing something with a **mouse hovers over** trigger, why not use a **user clicks** trigger instead? Or, instead of using a literal drag-and-drop interaction, you could simulate one with selections: have the user select the drag object and then select the drop location.

[18] Universal design: designing something in a singular way so that it's accessible to, usable by and beneficial to the greatest number of people possible including those with different abilities, as opposed to designing different versions of the same product, each suited to a different audience based on their abilities. Curb cuts are a classic example of effective universal design in how they benefit so many different segments of the population: people in wheelchairs, people with strollers, people with wheeled luggage, the elderly, skateboarders, etc.

[19] Think long and hard about whether an inaccessible interaction type really is *absolutely necessary*. Remember that universal design benefits not only your audience but also you as a designer, since building and maintaining a single version of an interaction likely takes less time than building and maintaining multiple versions.

Since it provides helpful context, the slide title often should be the first element encountered by accessibility tools in a slide[20].

To determine the best sequence of (other) in-slide elements, envision how you would want to encounter those elements if you had no prior knowledge of what they were, could not see them and instead had them read to you one after another. In most cases, simple logic will prevail; buttons should go after whatever introduces them and what they do, layer elements should go after whatever caused the layer to open, etc. Don't hesitate to seek input from others on what the most effective and logical element sequence may be.

Storyline

In Storyline, the in-slide element sequence is controlled through the **Tab Order** panel.

Hide Decorative Elements and Minimize Tab Stops[21]

Decorative elements[22] should not be visible to accessibility tools.

Besides hiding decorative elements from accessibility tools, try to find other ways to minimize the number of tabs stops in each slide, such as using alt text to convey more than one element's information or by combining multiple portions of text into a single text frame (use typographic settings like paragraph spacing and indents to achieve the desired layout).

Storyline

Keep in mind that the decorative elements (e.g., background shapes, active state indicators, etc.) in many template interactions, including quizzes and quiz feedback layers, are visible by default and need to be made not visible[23].

---

[20] In some cases, it may be more practical to have the slide title conveyed, as the first thing a screen reader user encounters in a slide, through another object's alt text as opposed to through the slide title text box. For instance, take the example illustrated in the Preventing Non-consensual Media section: in that case, the manual play button is the only in-slide object screen reader users encounter until they play the slide. For slides like that, the play button should include the slide title in its alt text—e.g., "Play Welcome to the Course slide"—and be ordered *after* the slide title text box in the tab order; that way, users don't encounter the slide title twice in a row as would be the case if the play button was ordered before the slide title text box (they'd encounter the title once in the play button's alt text and again once the base layer elements, including the slide title text box, are made visible) and instead, they'll encounter the slide title in the play button's alt text, followed by whatever element is next in the tab order, and they'll only re-encounter the slide title if they purposefully back-tab to it.

[21] Having to hit tab numerous times each slide can be frustrating for screen reader users and potentially painful for keyboard navigators.

[22] I.e., elements whose only purpose is to achieve visual styling; they do not convey information or host interaction triggers.

[23] This may require venturing into **Slide Master** or **Feedback Master**.

Use Layer Settings and Other Means to Hide Base Layer and Other Elements When Appropriate

In certain situations, it's best to hide base layer or **slide layer** content from accessibility tools when a new layer opens. This helps streamline the accessibility tool user experience, eliminating "noise" and enabling users to only encounter the elements that are most actionable in a given moment. For example, in a tab interaction slide, you don't want a user to be able to read a previously opened tab's contents when a new tab is opened. Or, in an interactive slide, you don't want a user to be able to continue engaging with the interactive in the base layer if an error message has popped up in another slide layer.

Associated Storyline How-to Guides

- Managing Layer Visibility

No Non-consensual Media

Audio and/or video should not play in a slide without a user's consent[24], such as is exercised by clicking a play button or engaging in such a way that can reasonably result in audio being played[25]—that is to say, *users must manually play all audio and video*.

This can be a very impactful requirement. It's also creates a universal design challenge in that, if a course has audio in every slide, having to click a play button on each slide is likely not the ideal experience for many users.

So, as part of the implementation of this requirement, consider also including a feature that allows users to turn "manual play" off—so that audio and/or video play automatically when the slide opens ("autoplay")—and back on, if desired. Just keep in mind that the manual play setting needs to be the course's default setting, which can then be turned off.

Associated Storyline How-to Guides

- Preventing Non-consensual Media
- Providing an "Autoplay" Option
- On/Off, True/False Toggle Button(s)

[24] Clicking the previous slide's Next button is not sufficient consent.

[25] For example, a buzzer sound effect when a user submits an incorrect choice, or feedback narration playing when a user submits a correct answer and is shown the correct answer feedback.

Enable Pausing/Stopping, Resuming and Scrubbing of Media

Users should be able to easily pause/stop and resume complex media, such as slideshows, videos and audio narration[26], since such media can interfere with the use of some assistive technology devices, like screen readers.

Ideally, users should also be able to scrub (fast forward and rewind) time-based media like audio and videos using keyboard controls.

It's recommended that you implement keyboard shortcuts to assist with pause/stop and resume functionality, since a course/slide may require multiple keyboard navigation steps to reach built-in pause/play buttons[27].

Provide Captions for Videos and Synchronized Audio

Captions must be provided for videos and synchronized audio (i.e., audio that has other events synched to it, such as a portion of text becoming visible within a slide at a certain point in the audio).

Captions can be "open," meaning they're always visible, or "closed," meaning users can turn them off and on. Closed captions should, by default, start on (i.e., visible) and require users to turn them off if they so desire[28].

Storyline

Storyline 3 and 360 have a built-in captions feature that can accommodate .srt, .vst and other common caption formats. The feature also allows you to select portions of a media element and manually type in captions for the selected segment.

However, the built-in captions are not fully compliant with the UC's accessibility requirements[29], so in order to satisfy those requirements, you would need to create your own captions.

Associated Storyline How-to Guides

- Creating Fully Accessible Captions

[26] This is not necessary for less-complex media like sound effects.
[27] For instance, in a Storyline course with standard player features, not even counting the tab stops associated with in-slide content, you may have to press the Tab key as many as six times to reach the pause button.
[28] This approach exemplifies a universal design principle that accessibility-focused designers should keep in mind and seek to enact as much as possible: products should, by default, be as accessible as possible, and if users want to roll-back accessibility features, they have to act to do so; except in unique circumstances, products should not have a less accessible default state and require accessibility features to be turned on.
[29] The captions cannot receive focus, so they cannot be processed by a braille conversion tool or be read by a screen reader.

Provide Transcripts for Video and Audio

Transcripts must be provided for all video and audio.

Allow Early and Easy Access to the Slide Transcript

This means a few things:

1. Transcript access should be available as early as possible within a slide's tab order
    a. It is generally suggested that transcript access be ordered immediately after the slide title.

Storyline

In Storyline, this also means:

2. The built-in **Notes** feature should not be used to display slide transcripts[30].
3. You will need to implement your own method for making each slide's transcript available, such as having the transcript in a **slide layer** that opens when a user clicks a button within the **base layer**; that button, the "transcript access," would need to be available as early as possible in the slide's tab order. It helps to also have show transcript (layer) keyboard command.
4. Consider also making available, early and easily, a transcript of all the slides in a course.

Associated Storyline How-to Guide

- Example: Transcript Access and Tab Order

Provide Alt Text or Audio Descriptions for Video

If a video contains visual content that is not sufficiently described by the video's audio (and subsequently, by its closed captions)[31], the video will need alt text[32] or an audio description[33] to convey the visual content to non-sighted learners.

---

[30] Since, as a player control, users have to hit tab too many times in order to arrive at it.

[31] For example, imagine a recorded lecture that features a graph. If the lecturer sufficiently describes the graph—i.e., what its x-axis and y-axis represent, what information is plotted on the graph, what this information as a whole tells you, etc.—the video may not need additional, non-visual support, but if the lecturer simply says something like, "take a look at this graph; what do you think?," this video needs non-visual support such as alt text or an audio description.

[32] Due to other considerations, it may be appropriate to assign the video's alt text to a different element. For instance, if you've hidden the video from accessibility tools so it can only be paused/played through the player controls, you may have to place a transparent rectangle over the video and could assign the video's alt text to the rectangle.

[33] Audio descriptions can be captioned but do not require it, since they convey information that is already communicated visually through the video.

Provide Alt Text Descriptions for Visual Elements

Use alt text to describe visual elements. Alt text should convey whatever meaning sighted users would take away from looking at the visual element. For interactive elements, like buttons, alt text should convey the action performed by the element.

Try to do so in a way that minimizes the number of tab stops in the slide[34].

If a chart, graph, diagram or complex image is used in conjunction with audio and/or video, remember that, in accordance with universal design, the most effective way to relay the graphic's meaning is to describe it in all meaningful detail within the audio and/or video[35].

Provide All Necessary Link Information

When screen reader users encounter a link, they need to know what the link directs to—e.g., a specific website, webpage or file—and, in many cases, where the link will open, so that if it does open in a new window, they're aware of that.

Never use "click here," "read more," etc. as link text since they do not provide necessary context as to a link's nature and destination.

For links that will open in new windows/tabs, consider using "Open *website/webpage name* in a new window" alt text.

For files: since whether the file downloads automatically, opens a download/save panel or opens within a new browser tab depends on the individual browser's settings, consider just using the file name and file type as the link text or alt text. E.g., "UC Clery Act Policy pdf".

Storyline

In Storyline, you have to associate links with objects—such as text frames, shapes or images—not with text segments within text frames, since those are not recognized by accessibility tools[36]. The best way to convey link text is through the alt text associated with the linked object.

---

[34] For instance, if a diagram is composed of four individual elements each of which could be given its own alt text, instead of having each element read separately, it may be better to hide all of them from accessibility tools, group them and give the group an alt text description that describes the diagram as a whole.

[35] For example, imagine a narrated slide that includes a chart. Instead of having the narration say, "this chart shows that sales increased throughout Q1," and having the chart's alt text describe the chart's displayed sales averages at the start, midpoint and end of Q1, have that same alt text for the chart, but improve the narration so it describes all the chart's meaningful details, such as, "This chart shows that sales increased throughout Q1. At the start of Q1, average sales were 4. By the midpoint of Q1, average sales increased to 6, and by the end of Q1, sales surged even more rapidly to 12."

[36] See Be Aware of, and Word Around Bugs… Insert Hyperlink for more details.

Implement Keyboard Shortcuts to Facilitate Common In-course Actions

Keyboard shortcuts help accessibility tool users move through and interact with a course more efficiently. For instance, imagine if in a Storyline course, you could press the N key to move to the next slide as opposed to having to press the Tab key five times, let's say, to reach the Next button in the course player and then press the Enter key to activate the button and move to the next slide.

If possible, assign keyboard shortcuts to as many common in-course actions as possible, such as:

- N = Moving to the Next slide
- P = Moving to Previous slide
- T = Opening the slide transcript
- X = Closing the slide transcript or a layer, such as pop-up layers and quiz feedback layers
- S = Stopping/Pausing the slide
- R = Resuming/Playing the slide
- Q = Submitting a quiz question
- M = Open a menu[37]

Associated Storyline How-to Guide

- Keyboard Shortcuts

Provide (Accessibility) Instructions

Provide instructions for accessibility controls and features, especially if either are unique to a particular course or course authoring program such as Storyline. Some accessibility-related instructions are useful for all users; let users know how/where to access transcripts and of any keyboard shortcuts that enable them to navigate through the course more efficiently. Explain the Manual Play/Autoplay settings and how to switch from one to the other.

Storyline

Inform users how to navigate forwards and backwards through course components using the tab key and shift+tab. Describe the **skip navigation** button that screen reader users will encounter as well as what the button does[38].

Allow Users to Easily Restart a Slide or Slide Content

Users should be able to easily (i.e., with minimal clicks or keystrokes) restart a slide—i.e., reset the slide to its initial state—and/or any media or interactions within it.

---

[37] This will typically refer to a custom menu you've created within a slide, as most menus that are built into course players cannot be opened with keyboard shortcuts.

[38] It returns users to the top of the slide's tab order.

Associated Storyline How-to Guide

- Restart a Storyline Slide

Be Aware of, and Work Around, Bugs and Other Shortcomings That Affect Accessibility Features and Other Components

Not all accessibility features work as advertised or in all circumstances, so it's imperative that you thoroughly test your courses with accessibility tools, document bugs and develop work-around solutions.

Storyline 3

The following bugs currently exist in Storyline 3[39][40] and need to be accounted for:

Tab Order Reset in Layers

The August 16, 2018 (Build 3.5.16548.0) Storyline 3 update includes a critical bug that makes it very difficult, if not impossible, to sufficiently control tab order within a slide that includes layers (beyond the Base Layer).

The bug causes tab order to restart at the beginning of the tab order hierarchy whenever a layer opens[41].

It's recommended that you contact Articulate and request a download file for the May 8, 2018 (Build 3.4.15731.0) update, which is the prior update.

Object State Accessibility Tool Visibility and Alt Text Settings

1. It is possible for an object to have different accessibility tool visibility or alt text settings for its **Normal** state and its non-state. In such cases, the **Normal** state settings will take precedence, so:
    a. Make sure the **Normal** state settings match your intentions.
    b. When diagnosing incorrect accessibility tool visibility or alt text findings, remember to check the object's **Normal** state.

[39] Storyline 360 has not been tested to establish if these bugs exist in it as well.
[40] Articulate has been made aware of these bugs and is currently in the process of resolving them.
[41] E.g., let's say the first element in a slide's tab order is the slide title. #2 is an open transcript button that opens a layer containing the transcript text. #3 is transcript text. So, if this bug were present, a user would tab to the slide title, then to the transcript button and select that button. The transcript layer would open, and when the user pressed tab, (assuming layer settings did not prevent them from engaging with base layer elements) tab focus would go first to the slide title, then the transcript and then the transcript text.

    c. Be especially aware of this with buttons, since by default, they have multiple states, so the **Normal** state and non-state settings will inherently be different with many common button-creation workflows[42].
    d. If you plan on adding states to an image or shape and want each state to have the same accessibility tool visibility and/or alt text settings, establish those settings before you create the new states; that way, the new states will have those settings as opposed to you having to establish them individually in each state.
2. Buttons and shapes cannot have unique accessibility tool visibility or alt text settings for *both* the built-in **Selected** and **Visited** states...
    a. Because, when a button or shape is in one of those states, its non-state settings apply, so…
    b. If you want to give unique settings to one of those states, assign the desired settings to the non-state[43].
    c. If you want to use both the **Selected** and **Visited** states and have unique accessibility tool and/or alt text settings for each, you would have to use a custom state and triggers to mimic one of those two states[44].

Matching Drop-Down

Matching drop-down questions should not be used because they do not provide screen reader output while the user is engaged with them[45].

It is, however, possible to mimic **matching drop-down** questions using layers, states and/or variables, and freeform questions.

---

[42] E.g., If you create a button, then adjust its non-state alt text, the Normal state alt text will be different. If you create a button, then add a button icon, the two alt texts will be different. If you create a button, then add a button icon, then adjust the non-state alt text before adding button text, the two alt texts will be different. The most reliable way of keeping both alt texts aligned is by using button text that is the alt text (i.e., if you create a button, then add button text before customizing any alt text, the button text will be the alt text for all states and non-state; this is true even if you add a button icon before button text), but unfortunately, once you manually adjust either state's alt text, they unsynch, and often you need alt text to be more descriptive than button text, so rarely can this workaround actually be employed.

[43] For example, let's say an object has a **Normal** state and a **Visited** state. Give the **Normal** state the settings you want the object to have before it's visited, and give the non-state the settings you want it to have after it's been visited.

[44] For example, let's say an object has a **Normal** state, a **Visited** state and **Selected** state, and that it needs different settings in each of those states. You can use either the built-in **Visited** state or the **Selected** state but not both. So, you use the built-in **Selected** state and give the non-state the settings that you want applied to the **Selected** state. Then, you'll need to create a custom visited state (not named **Visited** since naming a state **Visited** applies the built-in **Visited** state), as well as a variable and set of triggers that will assign the object the custom state after it's been visited.

[45] I.e., when you arrow through the drop menus, the screen reader will not declare which menu item focus is on or has arrived on, so screen reader users will not know which item within the menu they have selected.

Multiple Response Questions

Multiple Response questions should be set to **Shuffle: Answers** to avoid a bug that causes all question choices to be read when a user tabs forward onto the first question choice.

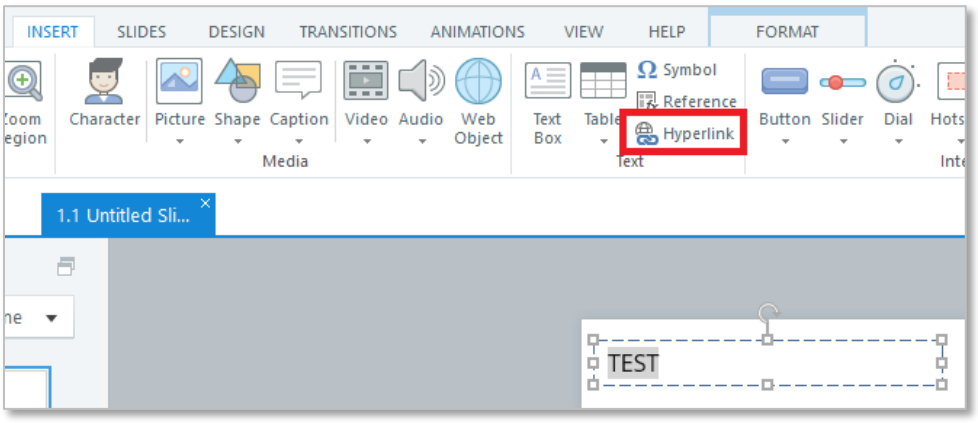
Keyboard Shortcut/Commands

Be aware that a current Storyline bug will in some cases temporarily prevent keyboard shortcuts from working, but there's a simple way to rectify this if learners are made aware of it.

For keyboard shortcuts to work, focus must be somewhere within the slide. For instance, if focus is on the slide title within the slide, a keyboard shortcut will work, but if focus is on the **Prev** button in the player controls, which is not within the slide, a keyboard shortcut will not work. Sometimes when you move between slides or layers, focus goes outside the slide. In such cases, users need merely hit the tab key to re-enter the slide, thus re-enabling keyboard shortcuts.

Insert Hyperlink

A hyperlink added by selecting text and using the **Insert** > **Hyperlink** feature will not be accessible, since it cannot receive keyboard focus and be read by accessibility tools.



To create a text hyperlink:

1. Give the text a unique style[46] that will visually convey it is a link.
    a. E.g., a unique color and underlined.
    b. You can use the **Insert** > **Hyperlink** feature to give the link text a unique style.
2. Place a rectangle shape over the link text.
    a. Make sure the rectangle is large enough that it can be clicked but not so large that it covers other elements that are not intended to act as links.
3. With the rectangle selected, open the **Format Shape** panel.
4. In the **Fill** tab, set the fill color **Transparency** to 100%.
5. Set the shape outline to **No outline**.
6. Give the rectangle the necessary alt text for the link.

Note: a shape with no fill color (as opposed to a 100% transparent fill color) will support keyboard navigation but cannot receive a mouse click.

[46] Just remember style rules like maintaining sufficient foreground-background contrast and not using color/contrast as the sole means of conveying information

Bypassing Browser-based Manual Play Setting in Storyline

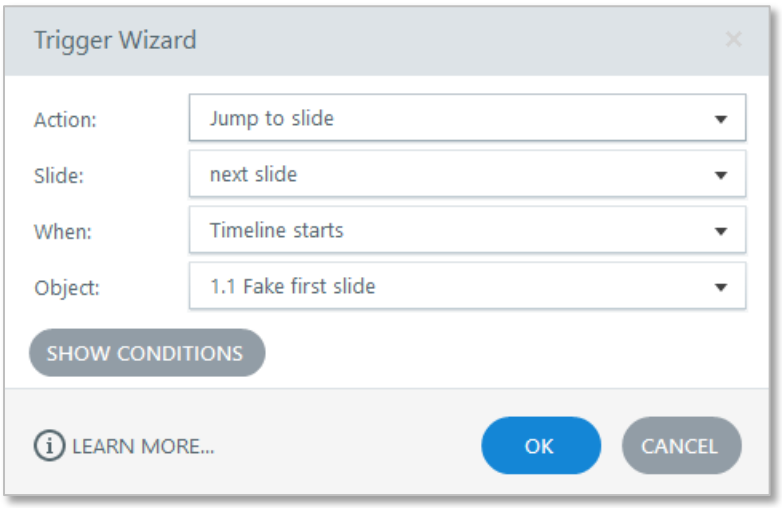
PLEASE NOTE: this issue was partially resolved with the August 16, 2018 update but not enough to meet UC standards[47], and because of the Tab Order Reset bug, it's recommended that you avoid the August 16, 2018 regardless.

A May, 2018 Storyline 3[48] update included a "fix" that causes courses to (re-)open in certain browsers—most notably Firefox, Safari and often Chrome—with its content grayed out and only a play button visible *if* the course's opening slide contains audio or video.

This fix created a critical accessibility problem in that the play button is not visible to accessibility tools, so users relying on those tools cannot click it, get past it and start the course.

To bypass this feature, you need to provide an opening slide that does not contain audio or video. There are a number of ways you can do this, but here is perhaps the simplest solution.

1. Create a new slide and move it so that it is the very first slide in the course's opening scene.
2. Give the slide a neutral light gray background, such as #D9D9D9.
    a. You could even give it a gray gradient background to match the default Course Player background (#E9E9E9 to #CFCFCF).
3. Add this trigger to the slide: **Jump to slide**, **next slide**, when the slide **timeline starts**.



4. Open the **Menu** tab of the **Player Properties** and delete this slide from the **Menu**.

[47] The play button is now visible to accessibility tools but doesn't have alt text that sufficiently describes its function.
[48] We do not know if this setting is currently in place with Storyline 360.

Appendix A: Storyline How-to Guides

Fully Accessible Video Controls

Videos present a unique challenge because there a few different methods for including them in a slide/course, but only one has been found to be sufficiently compliant[49].

This is the recommended method for including a video in a Storyline course such that it can be fully controlled by accessibility tools and exhibit the best possible universal design. A key aspect of this method is that the video becomes synched to the slide's **seekbar**, so that the slide seekbar can act as the video's seekbar and allow all users to scrub forwards and backwards through the video using seekbar.

1. Insert the video as a file into a slide.
2. In the **Video Tools Options** tab, make sure that the video is set to:
    a. **Show Video**: **In slide**
    b. **Play Video**: **Automatically**[50]
    c. **Show Video Controls**: **None**
3. Hide the video from accessibility tools[51][52].

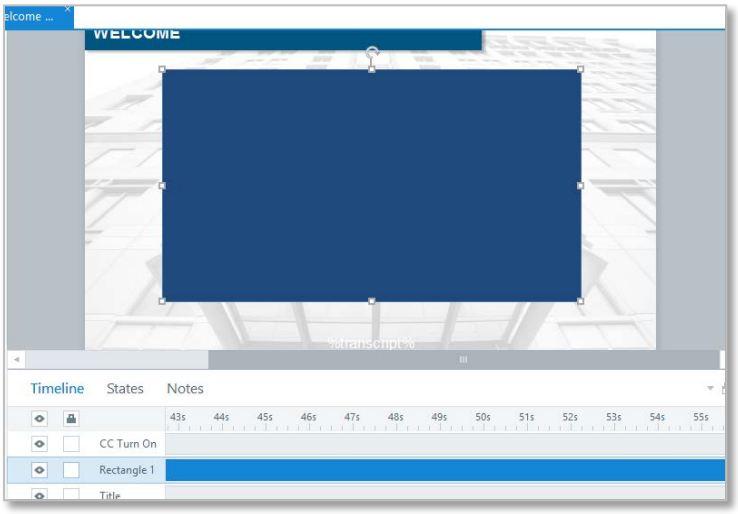
[49] Using the video's built-in controls is not an option because those controls are not compatible with accessibility tools. Using the Insert Video as Web Object (most frequently using YouTube embed code) is not an option because the in-Storyline YouTube controls do not provide sufficient screen reader outputs (though they are, more-or-less, compatible with keyboard navigation). If you want to include a YouTube video, it is best to link to it within the YouTube site.

[50] This setting is required to tie the video to the slide's seekbar, but as you may remember from the No Non-consensual Media section, videos should actually only start if a user manually starts them; you'll use other settings to ensure that the video is paused as soon as it automatically starts, so that the user can then manually play it.

[51] This may seem counterintuitive, but if users can access the video with accessibility tools, they can play/pause it without the seekbar also playing or pausing which would cause the two to unsnych, disabling the user's ability to reliably scrub the video using the seekbar. Or, you could leave the video visible to accessibility tools and do some custom work with states, variables and triggers to have clicking on the video also pause/play the slide seekbar.

[52] In older versions of Storyline 3, videos also needed specific start and stop triggers to replay properly in Firefox. If you encounter replay issues, first try updating Storyline to the most recent version. If that doesn't correct the issues, try this: add a trigger to the video that will play the video when the video completes (this replays the video). Next, create a shape outside the visible slide area; hide it from accessibility tools, give it a trigger that will stop the video from playing when the shape's timeline starts, and position the shape in the slide Timeline so it appears .01-.10 seconds after the video's end and lasts ≤ .25 seconds until the slide Timeline ends (this essentially pauses the replay so the video begins from its start when the slide is reset to its initial state).

4. Create a rectangle that has the same size and position as the video and is arranged in front of the video[53].



5. With the rectangle selected, bring up the **Format Shape** panel (right click on shape > **Format Shape**, or Ctrl+Enter).
6. In the **Fill** tab, set the **Transparency** to 100%.
7. In the **Line Color** tab, set the **Line Color** to "**No line**."
8. With the rectangle still selected, open the **Size and Position** panel, **Accessibility** tab; give the rectangle the alt text you would've given the video: e.g., "*Video title* video; use the course play button, volume slider and seekbar to control this video."
9. In the slide **Timeline**, set the rectangle's timing so it is present for at least the duration of the video, if not the entire slide.

Make sure to also explore the Captions and Preventing non-consensual media sections, as they have important information relevant to including video in a slide/course.

Preventing Non-consensual Media

There are many different ways you can achieve this standard within Storyline.

Depending on how you're using audio or video within a slide, one of the biggest challenges you may face when implementing this standard is keeping the audio or video tied to the slide's timeline/seekbar so that they start and end together, and so the user can reliably scrub forwards and backwards through the media using the slide's seekbar.

[53] This rectangle will be used to accomplish two things: 1) it will be placed atop the video and prevent users from pausing/playing the video via mouse clicks, which would cause it to unsynch from the slide seekbar (similar to how you'll hide the video from accessibility tools so keyboard users won't pause/play it); and 2) it can house the alt text you would give the video.

An effective solution is to have the media play automatically[54] within a slide or layer but to also have the slide or layer pause (using slide/layer settings or triggers) as soon as it starts so that the user manually "plays" the media by, in effect, un-pausing (i.e., resuming) the slide or layer timeline. However you achieve this, make sure the media and slide timeline/seekbar stay synched throughout subsequent pausing and resuming.

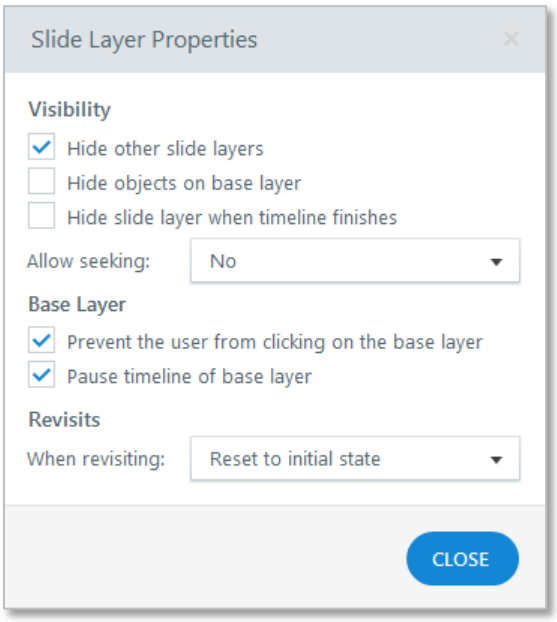
Example: Slide with video and base layer content purposefully obscured and non-clickable until audio is played

Again, there are many different ways of enacting this standard and many different considerations to weigh when deciding which method is best for your slide/course/learner experience. This example is intended to illustrate one method—which happens to use layer settings to pause the timeline/seekbar—nuy is not intended to suggest that this is the method you should use in all circumstances.

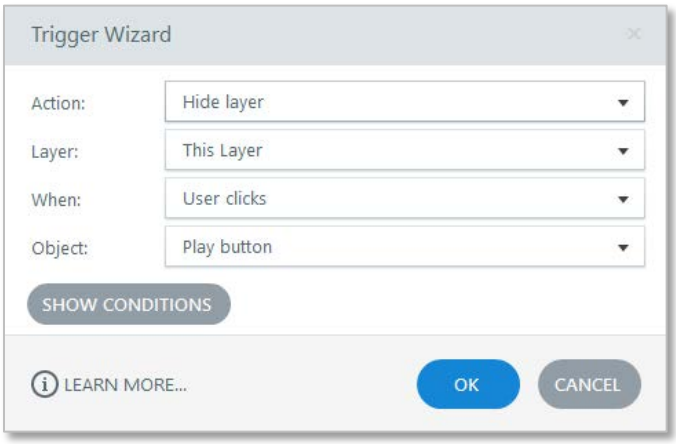
1. Create a new slide layer—let's call it the "manual play layer"—and give it the following **Slide Layer Properties**:
    a. **Hide other slide layers**: checked
    b. **Hide objects on base layer**: unchecked (in this example but overall, optional[55])
    c. **Hide slide layer when timeline finishes**: unchecked
    d. **Allow seeking**: **NO**
    e. Prevent users from clicking on the base layer: checked[56][57]
    f. Pause timeline of the base layer: checked[58]

[54] Unless audio is set to play via a trigger, it will play automatically when it is encountered within the timeline. Video can be set to play automatically through the "Play Video" setting in the "Video Tools Options" tab.
[55] This setting does not affect the overall functionality of what you're creating, so it's more of an overall design decision. I like to leave these objects visible, then place a full slide, semi-transparent rectangle in the background of my manual play layer so that people can see the base layer contents but understand that they're unavailable until they interact with the play button in front of the transparent rectangle.
[56] This setting is the best method for restricting accessibility tool users' access to the base layer content. If it is checked, base layer objects will be hidden from accessibility tools until the layer is closed. The setting affects sighted users as well, but the method of placing a semi-transparent rectangle over the entire slide is perhaps better for restricting sighted users' access since the rectangle restricts access while also visually conveying that access is restricted.
[57] Do you have to prevent access to base layer objects? Not necessarily, but in this case, we realized that if users could access the base layer contents, they could also access our Show Transcript button, which (depending on the transcript layer's settings) would either hide the manual play layer, causing the video to start technically without user consent since neither the Show Transcript button's appearance nor alt text convey that it plays the video, or it would wreak havoc with tab focus and tab order, making it difficult, if not impossible, for users to tab to and use the play button after opening the transcript layer.
[58] This setting is key to this method; it is the means by which the base layer is paused until the user actively chooses to play the slide and audio/video within it.

g. **When revisiting**: **Reset to initial state**



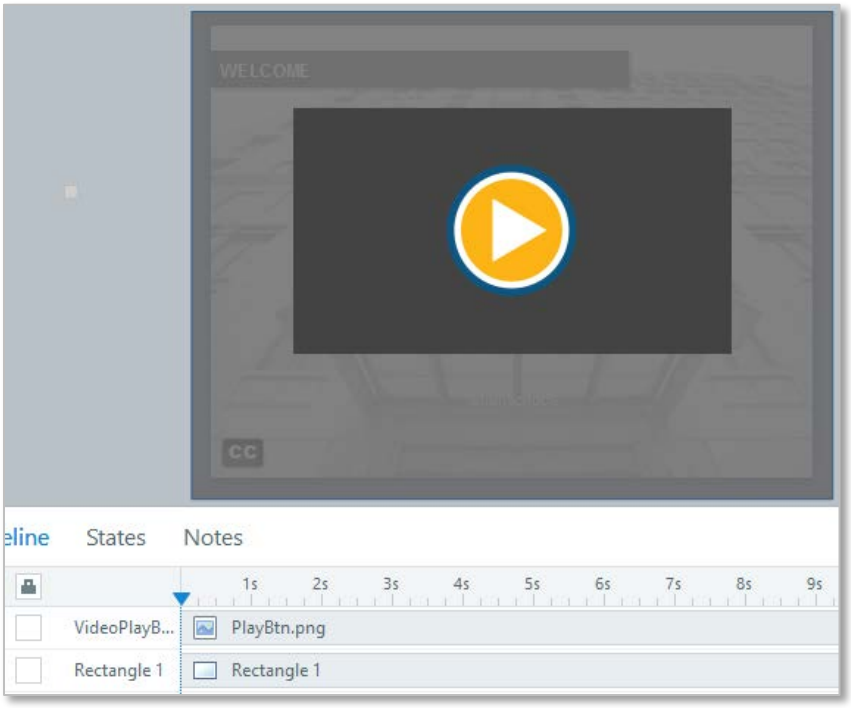
2. Insert a play button into the slide layer[59].
3. Edit the button's alt text so it conveys its function including the title of what will play: if it will play the slide audio, use something like "Play *slide title* slide"; if it will play a video, use something like "Play *video title* video"[60].
4. Give the button a trigger that hides the manual play layer (**This Layer**) **When: user clicks** the button.



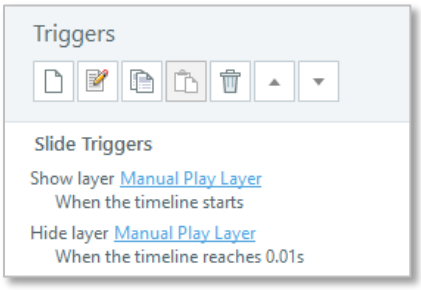
[59] This can be done using the **Insert** > **Button** feature, then adding button text and/or a play icon in the Button Tools Format tab. It can also be done by using a shape, though you wouldn't be able to add an icon as you could with a button, or you can could create a "button" outside of Storyline and insert it as an image.

[60] Keep in mind that, due to a Storyline bug, you may need to apply this alt text to the button's **Normal** state. This is absolutely true if you use the **Insert** > **Button** feature, since (until this bug is corrected) those buttons by default having differing non-state and **Normal**-state alt text.

5. If you have not hidden objects on the base layer using that **Slide Layer Properties** setting, consider adding a semi-transparent rectangle over the entire visible slide area, beneath the play button; this will help users understand that they need to click the play button in order to engage with the base layer contents.



   a. If you add a semi-transparent rectangle, make sure to hide it from accessibility tools.
6. In the base layer, create a slide trigger that will show the manual play layer when the slide timeline starts (shown in the screenshot after step #9).
7. In the **Triggers** pane, order this trigger so that it is the very first **Slide Trigger**.
8. Create a second trigger that hides the manual play layer when the slide's timeline reaches 0.01 seconds[61].
9. In the Triggers pane, order this trigger so it is the second slide trigger, immediately after the other trigger you created.



[61] This trigger is necessary because users could play/un-pause the slide by clicking the play button in the course player controls instead of the play button you created in the manual play layer; if they did so without this trigger being present, the slide would play, but the manual play layer would remain visible.

10. In the slide's **Tab Order** panel, move the manual play layer's Play button to its necessary position within the tab order[62].

Copy/Pasting to Other Slides

Copy/pasting this feature to other slides is relatively quick and easy, but it's best to proceed in this specific order to cut down on unnecessary work.

11. If you want to provide an "autoplay" option, complete those steps (detailed in the very next section) before copying to other slides.
12. Copy the manual play layer and paste it into the new slide.
13. Adjust the new slide's tab order so that the copied manual play layer's play button is ordered where it should be.
14. In the original slide, copy the **Hide layer**, manual play layer, **When: Timeline reaches** .01 seconds trigger, and position it first within the new slide's **Triggers** pane[63].
15. In the original slide, copy the **Show layer**, manual play layer, **When:** slide **Timeline starts** trigger, and position it first within the new slide's **Triggers pane**, above the previously copied "Hide layer…" trigger.
16. When pasted into the new slide, this "Show layer…" trigger will actually lose its layer assignment, so it will appear as "Show layer, **unassigned**, when…" in the **Triggers** pane. In the Triggers pane, click the red, **unassigned** text and select the manual play layer from the resulting drop menu.

---

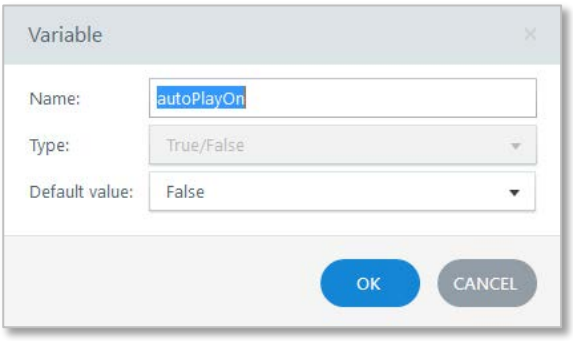
[62] As alluded to in the Elements are Purposefully Sequenced… section, I would actually make this play button second within the slide's tab order and leave its slide title within the base layer first, because the play button's alt text, "Play Welcome video," references the slide title, "Welcome." With this ordering, screen reader users will encounter the play button first, because I've used layer settings to prevent users from clicking/encountering base layer objects while the play button's layer is visible, and once they click the play button and close its layer, they won't have to encounter the slide title again when they move forward but still can re-encounter it, if they want to, by tab navigating backwards through the slide's tab order.

[63] Copying this trigger before the other is recommended so that when re-ordering both triggers in the **Triggers** pane, you can click the "Up" button an indiscriminate number of times without having to worry about positioning; whereas, if you copied the other trigger first, when positioning this trigger, you'd have to make sure to hit the "Up" button the specific number of times it took to put it in the second position.

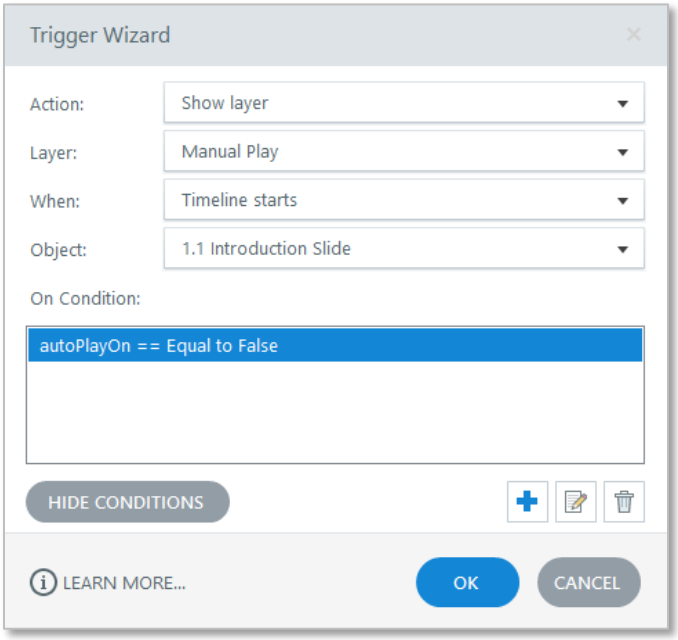
Providing an "Autoplay" Option

"Manual play" will, with some courses, not be the ideal experience for many users, so in those cases, it's recommended that you also implement a feature that allows users to switch from the default manual play setting to an autoplay setting that allows audio/video slides to start automatically, without having to first click a play button.

1. Create a **True/False** variable that will track whether autoplay is on or off.



   a. E.g.,
   b. Just remember, the variable's default value should reflect manual play being the default course setting.
2. Edit the trigger that pauses a slide's audio/video and add a condition so that it activates only if manual play is on/autoplay is off.



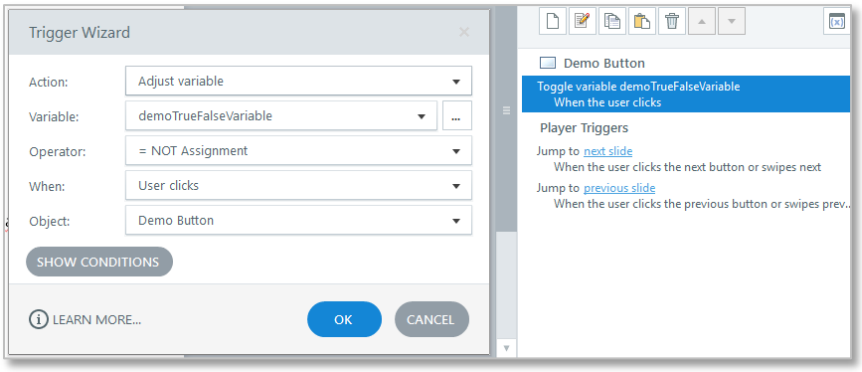
3. Add a button, or add buttons, to the slide base layer so that users can toggle between manual play and autoplay.
   a. Jump to the next two sections, or use the links above, to find instructions on adding a button or buttons. Add your button(s) before continuing with these instructions.

4. If you haven't already (as part of the add button(s) instructions), add conditional triggers to your button(s) so that the correct button state, or button, displays when the slide starts depending on the current manual play/autoplay setting. Adding these triggers to the button(s), and not to the slide, allows them to be easily copy/pasted to other slides by copy/pasting the button(s) (i.e., you don't have to manually copy/paste triggers).
    a. If you use one button:
        i. Trigger 1: **Change state** of button to [state that allows autoplay to be turned on], **when** button **timeline starts** on the condition that autoplay is currently not on (i.e., on the condition that the True/False autoplay variable equals the value associated with autoplay being off; e.g., autoPlayOn = false).
        ii. Trigger 2: **Change state** of button to [state that allows Autoplay to be turned off/Manual Play to be turned back on], **when** button **timeline starts** on the condition that Autoplay is currently on.
    b. If you use two buttons:
        i. Turn autoplay on button:
            1. Trigger 1: **Change state** of button to **Normal**, **when** button **timeline starts** on the condition that Autoplay is currently not on.
            2. Trigger 2: **Change state** of button to **Hidden**, **when** button **timeline starts** on the condition that Autoplay is currently on.
        ii. Turn autoplay off/turn manual play back on button:
            1. Trigger 1: **Change state** of button to **Hidden**, **when** button **timeline starts** on the condition that Autoplay is currently not on.
            2. Trigger 2: **Change state** of button to **Normal**, **when** button **timeline starts** on the condition that Autoplay is currently on.

On/Off, True/False Toggle Button(s)

Single On/Off, True/False Toggle Button

There are a few different ways you can set up a single button so that it acts as an On/Off button or toggles a True/False variable, but most of these methods hinge on one particular trigger: **Adjust Variable**, T/F variable name, **= NOT Assignment**, when user clicks the button—or, "**Toggle variable** when user clicks," as it appears in the Triggers pane.



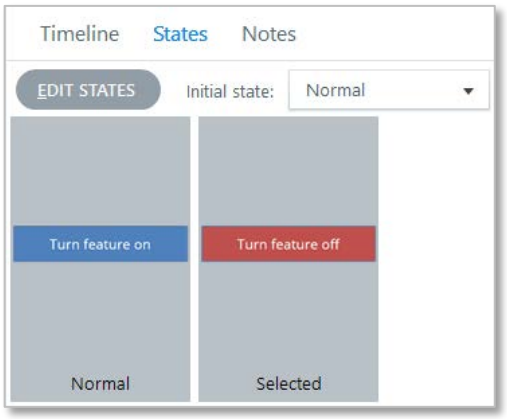
In most cases, it will be necessary that this Adjust/Toggle Variable trigger be ordered, in the Triggers pane, after other **on mouse click** triggers you may associate with the button so that the variable value changes only after triggers that are conditioned on the variable's value have been executed.

These single-button methods also involve the button having two appearances: one associated with "turn on/make true," the other with "turn off/make false."

The single-button approach requires less development (fewer triggers and less visual design work), but it comes with the trade-off of a limited ability to also use Hover and Down states, since changing the visual design of those states beyond their built-in styling may cause them to not work reliably for one or both of the button's appearances[64]; although, this design limitation can be partially overcome by limiting Hover and Down to one of the button's functions (either turn on/make true or turn off/make false) by applying a "Change button state to [state associated with the appearance/function that will not have Hover or Down states] when mouse is hovered over the button, do not **restore on mouse leave**, when the T/F variable = [value associated with chosen appearance]." This trigger prevents the Hover and Down states from showing when the button is in its chosen appearance, enabling you to use those states just for the other appearance *and* apply any design choices to them you'd like.

Using Selected State

An easy way to handle the button's design would be to use its **Normal** state for one appearance and alt text description, and its **Selected** state for the other appearance and (*see next paragraph)* alt text description—that way you can utilize built-in functionality and potentially[65] not have to create additional triggers beyond the Toggle Variable trigger shown previously.



[64] To understand this better, try this experiment: insert a button; add a **Selected** state to it; give that **Selected** state a noticeable difference, such as a different fill color and/or different text; preview the slide and note how the **Hover** and **Down** states are applied to the button when it is **Normal** and **Selected** (**Hover** and **Down** "effects" should be applied to the **Normal** and **Selected** states but in an integrated, not superseding, way); now, change the **Hover** state's color and change the **Down** state to a different color; preview again and notice how the **Hover** and **Down** states no longer integrate with the **Normal** and **Selected** states but rather supplant both.

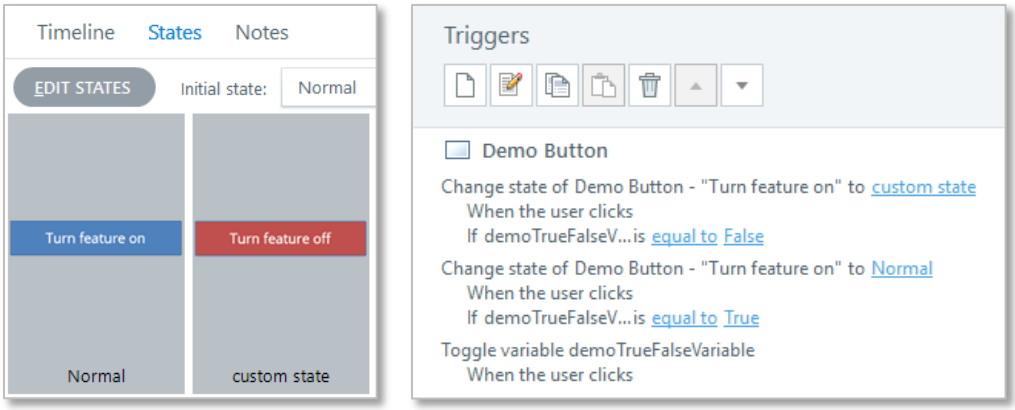
[65] This "potential" assumes the button will be used just to toggle a variable true/false. If secondary button functionality is intended, like hiding/showing closed captions, additional triggers would need to be added to achieve that functionality.

However, due to a bug that presently exists in Storyline 3[66], special consideration needs to be taken with this approach.

The bug prevents a button or shape's **Selected** state and **Visited** state alt text from being read by a screen reader; instead, the screen reader will read the non-state alt text[67][68]. So, delete the **Visited** state and ensure that the non-state alt text aligns with the alt text you'd give the **Selected** state[69].

Using a Custom State

Or you could create a custom state and triggers to change the button's state depending on a variable's value; just make sure the triggers are ordered correctly in the Triggers pane, with the "**Toggle variable**…" trigger ordered last:



Persistent, Course-wide Setting

If this button is associated with a persistent, course-wide setting (for example, if the button controls the visibility of a feature that is in all slides, then if you turn the feature off in one slide, it should remain off—with the "turn off" state hidden and the "turn on" state visible so it can be turned back on—in the next slide), you'll also need to include the following triggers. And while these triggers could be associated with the slide instead of with the button itself, having them associated with the button simplifies copy/pasting, since it enables both the button and these triggers to be copy/pasted together and not separately, as would be necessary if the triggers were associated with the slide instead of the button.

---

[66] Storyline 360 has not been checked yet to see if this bug exists in it as well.

[67] For example, let's say you have a button with the following states and alt text. Non-state alt text = "True", **Normal** state alt text = "Currently unselected, True. Click to select," and **Selected** state alt text = "Currently selected, True. Click to unselect." When the button is in its **Selected** state and focus is placed on it, a screen reader would read "True."

[68] Pictures appear to no longer have this problem; their unique **Selected** state and **Visited** state alt text will be read (but you should still test this in your own courses if applicable).

[69] To follow up on the previous example, the non-state alt text needs to = "Currently selected, True. Click to unselect."

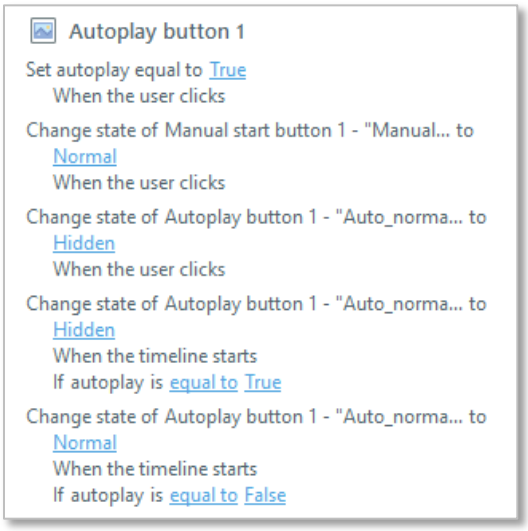
1. **Change state** of button to [state (either **Normal** or **Selected**/custom) associated with using the button to "turn on"], w**hen** button **timeline starts** if T/F variable is = to value associated with currently being off.
2. **Change state** of button to [state (either **Selected**/custom or **Normal**) associated with using button to "turn off"], **when** button **timeline starts** if T/F variable is = to value associated with currently being on.

On/Off, True/False Toggle Button Pair

You can also use a pair of buttons to turn a feature on/off or toggle a True/False variable. The advantage of this method is that both buttons can have **Hover** and **Down** states without any design/style limitations.

1. With this method, you only need to create a True/False variable that you'll associate with the buttons if you want the buttons to control a persistent setting—i.e., a setting that persist from slide-to-slide (e.g., you turn the setting off in one slide, it remains off in the next slide) or within the original slide on revisits (e.g., you turn the setting off in the slide, leave the slide and return to it or hit the revisit button in the player controls, and the setting remains off).
2. Create two buttons and give them both the exact same position and size. Style them as desired.
3. Give each button the following set of triggers:
    a. Button 1, which will turn the feature on and/or toggle the T/F variable to true:
        i. *If button controls a persistent setting:* Set T/F variable to assignment "True" when user clicks on this button
        ii. **Change state** of Button 2 to **Normal, when user clicks** on this button
        iii. **Change state** of Button 1 to **Hidden**, **when user clicks** on this button
    b. Button 2, which will turn the feature off and/or toggle the T/F variable to false:
        i. *If button controls a persistent setting:* Set T/F variable to assignment "False" when user clicks on this button
        ii. **Change state** of Button 1 to **Normal**, **when user clicks** on this button
        iii. **Change state** of Button 2 to **Hidden**, **when user clicks** on this button
4. If the buttons are also associated with a persistent setting you'll also need to include the following triggers for each button. And while these triggers could be associated with the slide instead of with the buttons themselves, having them associated with the buttons makes copy/pasting the buttons much easier.
    a. Button 1
        i. **Change state** of this button to **Hidden**, **when** timeline of this button starts if T/F variable = True
        ii. **Change state** of this button to **Normal**, **when** timeline of this button starts if T/F variable = False
    b. Button 2
        i. **Change state** of this button to **Hidden**, **when** timeline of this button starts if T/F variable = False

ii. **Change state** of this button to **Normal**, **when** timeline of this button starts if T/F variable = True



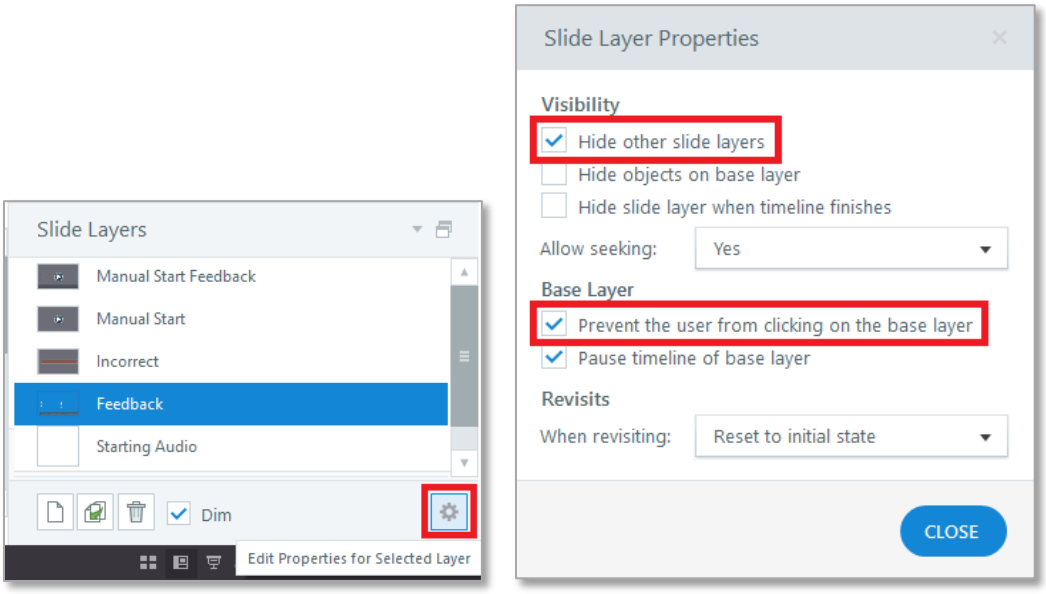
5. Give the buttons any other triggers necessary to it fulfilling its function.
6. If necessary, re-order each button's triggers so that the trigger that hides the button when the button is clicked is the last button[70].
7. Make sure each button has the appropriate alt text for its function (adjust and align non-state and **Normal**-state alt text, if necessary).

[70] If those triggers are not ordered last, any **when user clicks** trigger that comes after them may not execute.

Managing Layer Visibility

Use the **Edit Properties for Selected Layer** button to open the **Slide Layer Properties** panel[71], in which you can change the visibility of other slide layers when the selected slide layer is shown—using the **Hide other slide layers** setting—and/or **Prevent the user from clicking on the base layer** using that setting—which hides objects on the base layer from accessibility tools, but doesn't hide them visually[72], while the selected layer is shown.

In some, more complicated instances, you may need to use custom triggers and states to adjust an element's accessibility tool visibility.



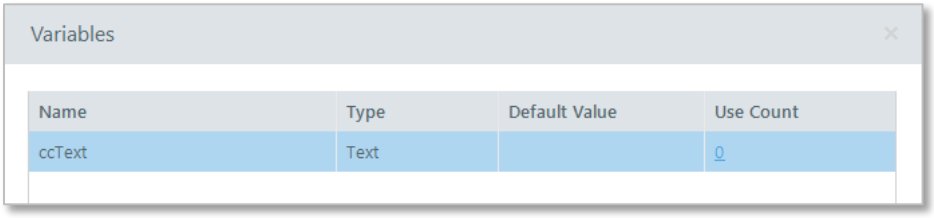
[71] Power-user note: You do not need to close this panel to save setting changes made within it, and you can select different slide layers while it's already open. If you have to edit multiple layers' properties, save steps/time by keeping this panel open as you go from layer to layer making your changes.

[72] The **Hide objects on base layer** setting hides the base layer both visually and from accessibility tools.

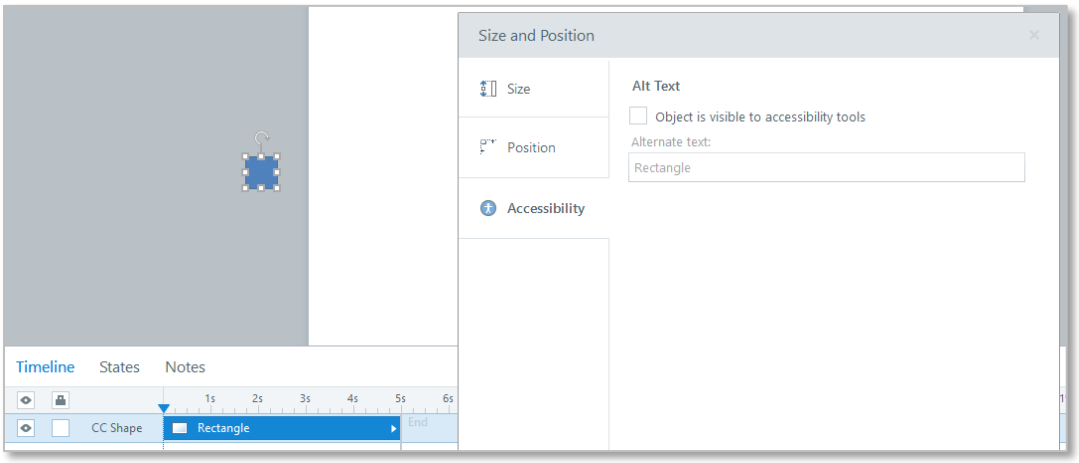
Creating Fully Accessible Captions

Unfortunately, Storyline's built-in captions are not fully compliant with the UC's accessibility requirements[73], so in order to satisfy those requirements, you would need to create your own captions. Here's one way you can do that:

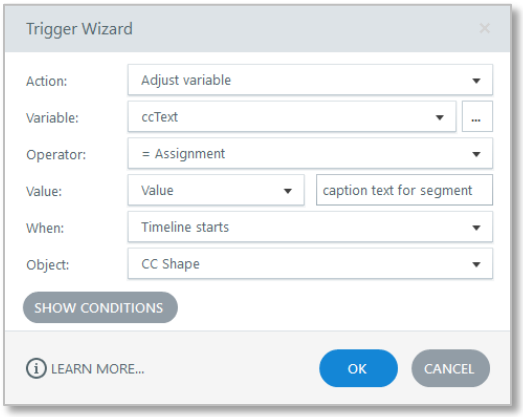
1. Create a **Text** variable with no default value; this will be the "caption variable."



2. Create a shape, position it outside the slide's visible area and hide it from accessibility tools.

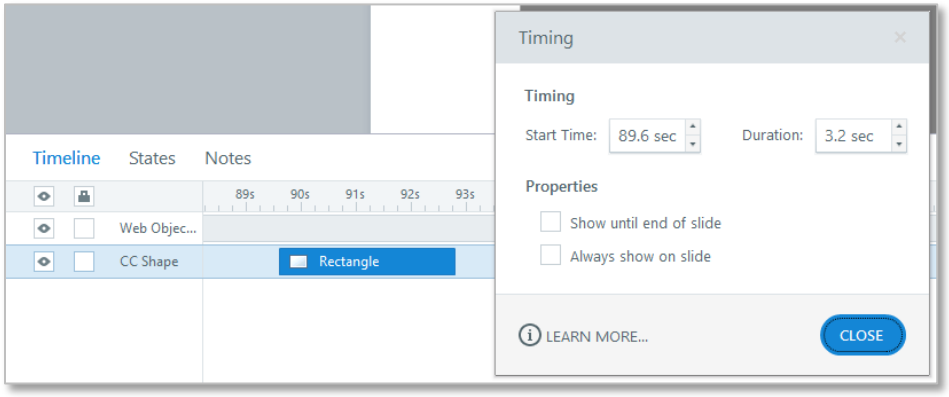


3. Create a trigger associated with the shape that will adjust the caption variable's value when the timelines of the shape starts, with the newly assigned variable value being a specific segment of caption text.



[73] The captions cannot receive focus, so they cannot be processed by a braille conversion tool or be read by a screen reader.

4. Position the shape and adjust its duration within the slide's timeline so that it is present during the stretch of audio that corresponds with the caption text[74].
    a. For example, let's say the caption text is, "in the eighteenth century they were always adding 'ical' on to the end of things[75]," and that this portion of video starts at the one minute, 29 second, 600 millisecond mark and ends at the one minute, 32 second, 800 millisecond mark. So…
        i. The shape should be timed to start after 89.6 seconds, and…
        ii. The shape should have a duration of 3.2 seconds



        iii. And the corresponding trigger would have "in the eighteenth century they were always adding 'ical' on to the end of things," as the assigned variable value.
2. Next, copy the shape you created and paste another instance of it outside of the slide's visible area.
3. The copied instance you just created will automatically have the same trigger—**adjust variable when timeline starts**—associated with it, but you'll need to change the assigned variable value in this instance to the next segment of caption text[76][77].

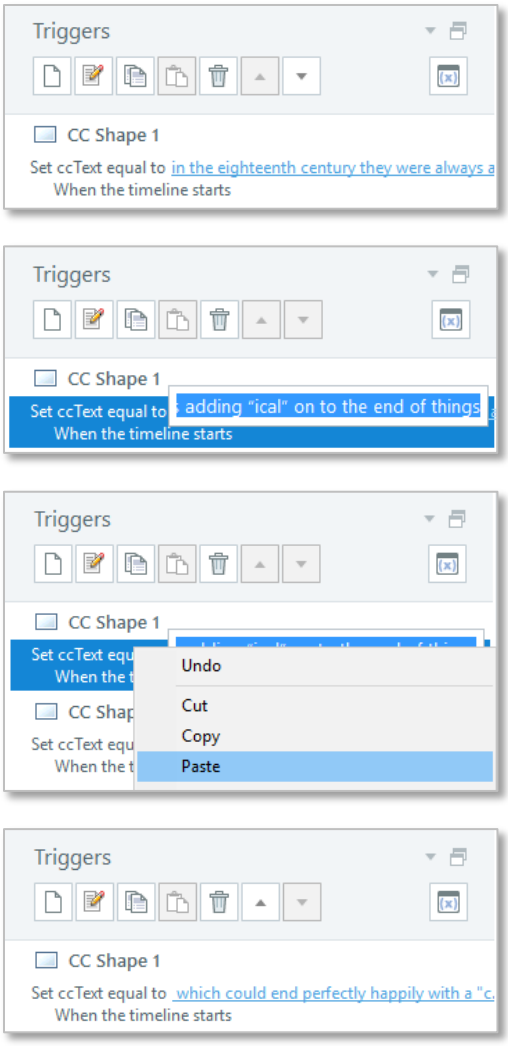
---

[74] It's possible to do this by manipulating the shape's presence within the timeline, but for the most accurate synchronization, it's recommended that you use the Timing panel, accessed by right-clicking on the shape within the Timeline and selecting "Timing…" from the resulting menu.

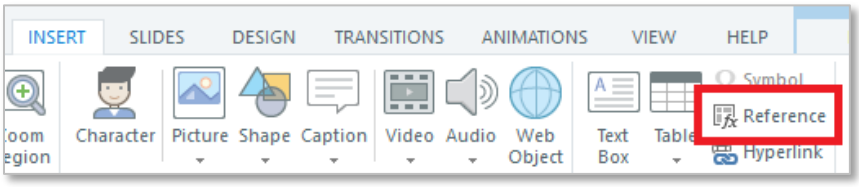
[75] This sample caption text has been borrowed from Lecture 01 of Yale Professor Joanne Freeman's wonderful Hist 116 Open Course on the American Revolution: YaleCourses. (2011, March 18). *1. Introduction: Freeman's Top Five Tips for Studying the Revolution* [video file]. Retrieved from https://www.youtube.com/watch?v=shTBSGoYtK0, 21:16.

[76] Beware: the variable **Value** field, whether edited in the **Trigger Wizard** panel or the **Triggers** pane, will not accept paragraph breaks. If you copy a text segment that contains a paragraph break and attempt to paste it into the **Value** field, only the pre-paragraph break text will come through. I would suggest that in such cases you need to remove paragraph break before copying the text segment, except that in most cases, two portions of text separated by a paragraph break should not be joined together in a single caption. They should be in separate caption segments so that users see the separation of ideas same as they would if they were looking at the original text which contained the paragraph break.

[77] Power-user note: as illustrated by the provided screenshot sequence on the next page, trigger properties that appear as blue, underlined text within the **Triggers** pane can be edited directly within the pane, without you having to select the trigger, click the **Edit Triggers** button and use the **Trigger Wizard** panel. Assigned value is one of these directly editable trigger properties, so when creating captions, save yourself some steps by pasting the next segment's caption text directly into the corresponding trigger within the **Triggers** pane.



4. Position the new shape and adjust its duration within the **Timeline** so that it begins and ends in-synch with the corresponding audio segment.
5. Repeat this process as many times as needed so that all audio is captioned.
6. Create a text box (let's call it "CC box") and include in it a reference to the caption text variable you created[78].



[78] Your cursor must be within a text box for the **Insert** > **Reference** feature to be available. You can also enter a variable reference into a text box by typing the variable name with a % symbol immediately before and after it.

7. Give this text box the width, position, shape fill and text properties you desire for the caption text.

8. To give the text box the necessary height, place your cursor after the variable reference and press the Enter key so that the number of lines within the text box corresponds with the number of lines needed to fit the longest segment of caption text[79].

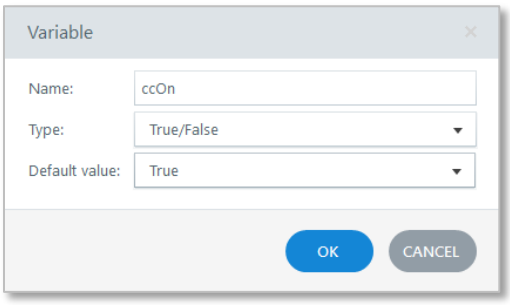
If these are intended to be open captions, your work is done, but if these are intended to be closed captions, you'll need to create a button, or buttons, to turn the captions off (i.e., change the text box's state to hidden) or on (i.e., change the text box's state to normal).

Turning Closed Captions Off and On

Closed captions should be visible be default and require users to turn them off if they're not wanted.

Follow the instructions in the previous Single On/Off, True/False Toggle Button and On/Off, True/False Toggle Button Pair sections to create either a button or buttons that will be used to hide and show the closed captions.

1. Create a **True/False** variable that will be associated with caption visibility. In these instructions we'll call it "caption variable."
    a. The variable's default value should reflect closed captions being visible by default.



2. If you intend for closed caption visibility to be a persistent setting course-wide (i.e., if you turn closed captions off in one slide, they remain off in another, similar to how the Autoplay/Manual

[79] It's recommended that you set the text box's vertical alignment to Middle, to avoid segments that require fewer lines of text having an awkward appearance.

Play setting persists slide-to-slide), add the following trigger to the CC box. If it will not be a persistent setting, you do not need to add this trigger, but you will need to create a unique caption variable for each slide that has captions:

   a. *If caption visibility persists:* **Change state** of CC box to **Normal**, **when** CC box **timeline starts** if caption variable **= true**.
   b. *If caption visibility persists:* **Change state** of CC box to **Hidden**, **when** CC box **timeline starts** if caption variable **= false**.

3. Add triggers to your button(s) to change the variable value and show/hide the captions text. Since trigger order will be important and some triggers may or may not need to be present depending on if caption visibility will be a persistent setting or not, we'll show all the triggers that would need to be associated with each button, highlighting those that are new to this purpose:
   a. A single On/Off, True/False toggle button using the **Normal** state to turn captions off and **Selected** state to turn captions on:
      i. *New:* **Change state** of CC box to **Hidden**, **when user clicks** this button if caption variable **= true**.
      ii. *New:* **Change state** of CC box to **Normal**, **when user clicks** this button if caption variable **= false**.
      iii. **Adjust variable**, caption variable, to =**NOT Assignment**, **when user clicks** this button.
      iv. *If caption visibility persists*: **Change state** of this button to **Normal**, **when** slide **timeline starts** if caption variable **= true**.
      v. *If caption visibility persists*: **Change state** of this button to **Selected**, **when** slide **timeline starts** if caption variable **= false**.
   b. A single On/Off, True/False toggle button using the **Normal** state to turn captions off and a custom state to turn captions on:
      i. *New:* **Change state** of CC box to **Hidden**, **when user clicks** this button if caption variable **= true**.
      ii. *New:* **Change state** of CC box to **Normal**, **when user clicks** this button if caption variable **= false**
      iii. **Change state** of this button to custom state, **when user clicks** this button if caption variable **= true**
      iv. **Change state** of this button to **Normal** state, **when user clicks** this button if caption variable **= false**.
      v. **Adjust variable,** caption variable, to =**NOT Assignment**, when user clicks this button.
      vi. *If caption visibility persists*: **Change state** of this button to **Normal**, **when** slide **timeline starts** if caption variable **= true**.
      vii. *If caption visibility persists*: **Change state** of this button to custom state, **when** slide **timeline starts** if caption variable **= false**.
   c. A pair of buttons to toggle On/Off, True/False
      i. Turn captions off button

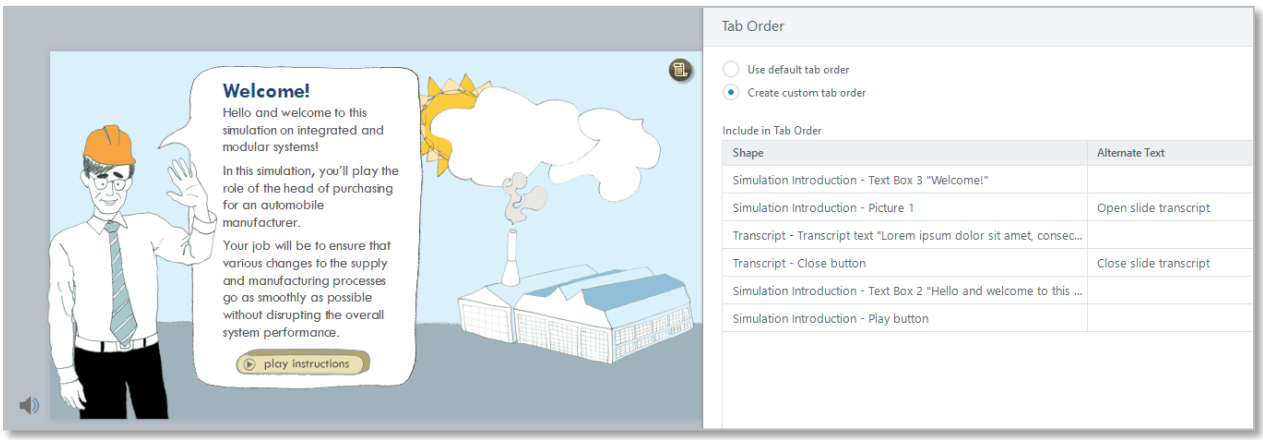
1. *New:* **Change state** of CC box to **Hidden**, **when user clicks** this button.
2. *If caption visibility persists*: **Adjust variable**, caption variable, to value true, **when user clicks** this button.
3. **Change state** of turn captions on button to **Normal**, **when user clicks** this button.
4. **Change state** of this button to **Hidden**, **when user clicks** this button.
5. *If caption visibility persists:* **Change state** of this button to **Normal**, **when** button **timeline starts** if caption variable **= true**.
6. *If caption visibility persists:* **Change state** of this button to **Hidden**, when button **timeline starts** if caption variable **= false**.

ii. Turn captions on button

1. *New:* **Change state** of CC box to **Normal**, **when user clicks** this button.
2. *If caption visibility persists*: **Adjust variable**, caption variable, to value false, **when user clicks** this button.
3. **Change state** of turn captions off button to **Normal**, **when user clicks** this button.
4. **Change state** of this button to **Hidden**, **when user clicks** this button.
5. *If caption visibility persists:* **Change state** of this button to **Normal**, **when** button **timeline starts** if caption variable **= false**.
6. *If caption visibility persists:* Change state of this button to Hidden when button timeline starts if caption variable **= true**.

Example: Transcript Access and Tab Order

Here's an example of an effective ordering of transcript elements:

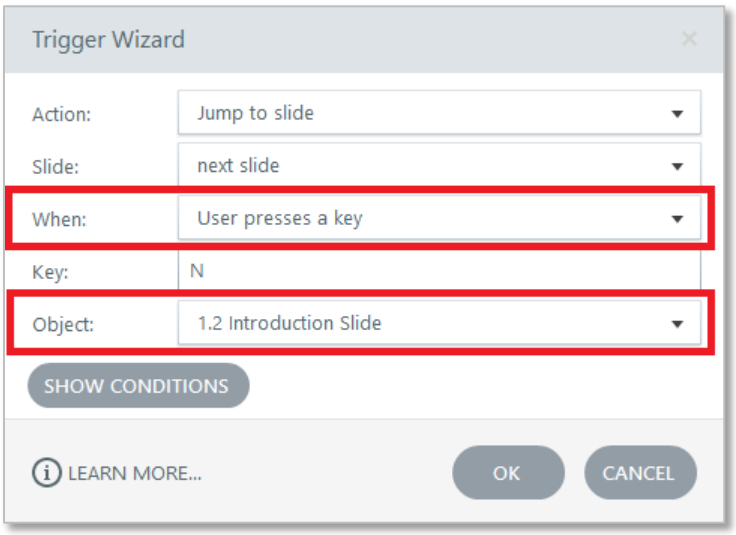


1. The slide title, "Welcome!" is order first in the slide, for context.
2. The "Open slide transcript" button is ordered second, immediately after the slide title.
    a. This button will show the "Transcript" layer.
3. The transcript text will be next if the "Transcript" layer is open…

4. Followed by the button that closes the “Transcript” layer.
5. The remaining slide contents follow.

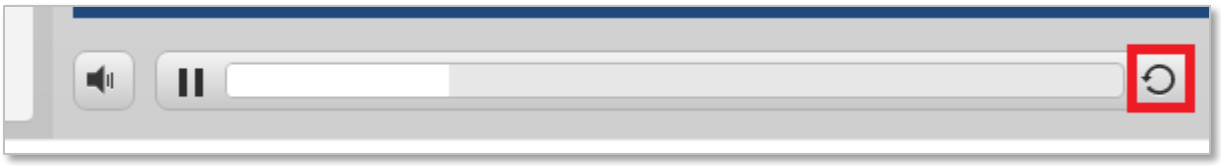
Keyboard Shortcuts

In Storyline, keyboard shortcuts are created through triggers that use the **When**: **User presses a key** property along with the **Object** being set to the slide[80].



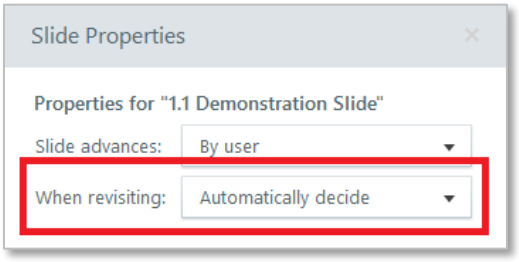
Restart a Storyline Slide

The simplest way to restart a Storyline slide is to use the revisit slide button, which will be available if the seekbar is enabled.



[80] It’s important that keyboard shortcut triggers are set to the slide and not to objects within the slide because if such triggers are set to an object, they’ll only work when keyboard focus is on that object.

When used, this button will execute whatever slide revisit setting has been established in the **Slide Properties**:



If the slide is set to **Resume saved state**, nothing will appear to happen when the button is clicked (since the slide will resume from wherever it was when the button was clicked).

If the slide is set to **Reset to initial state** and the revisit slide button is clicked, the seekbar play head will return to the start of the slide's timeline, objects will return to their initial states (as defined within the States pane), and any **When: Timeline starts** triggers will re-execute.

If the slide revisit button is not available *or if you want to provide an instance of that button that you can control the tab order of (allowing you to offer it earlier within a slide's tab order, making it more accessible*[81]*)* you can create a custom button within a slide and give it a **Jump to slide**, whatever slide the button is in, **When: User clicks**, this button trigger.

In more complex situations, such as when media, complex interactions and/or variables are involved, you may have to use a custom button and additional custom triggers.

[81] Which is a best practice