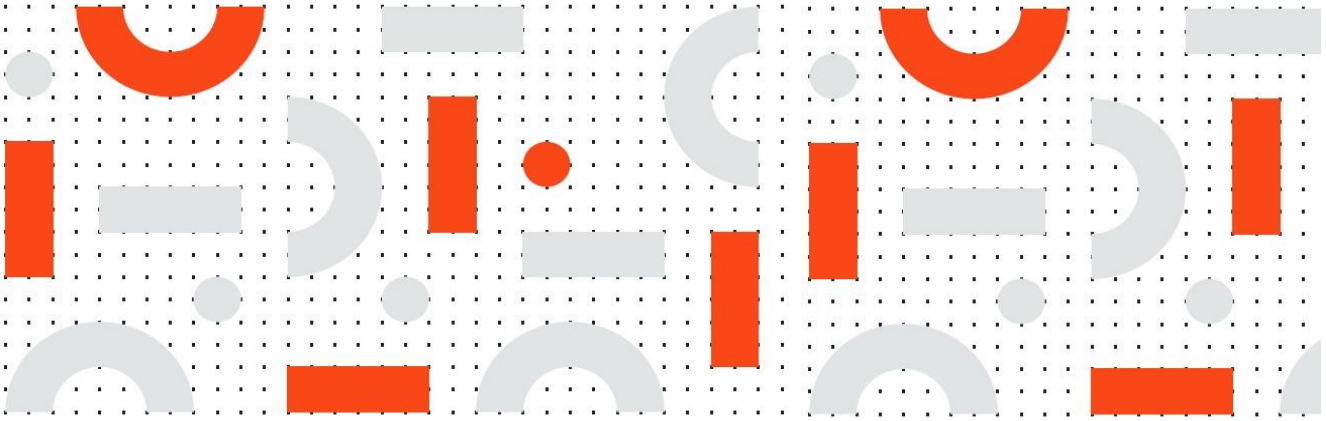
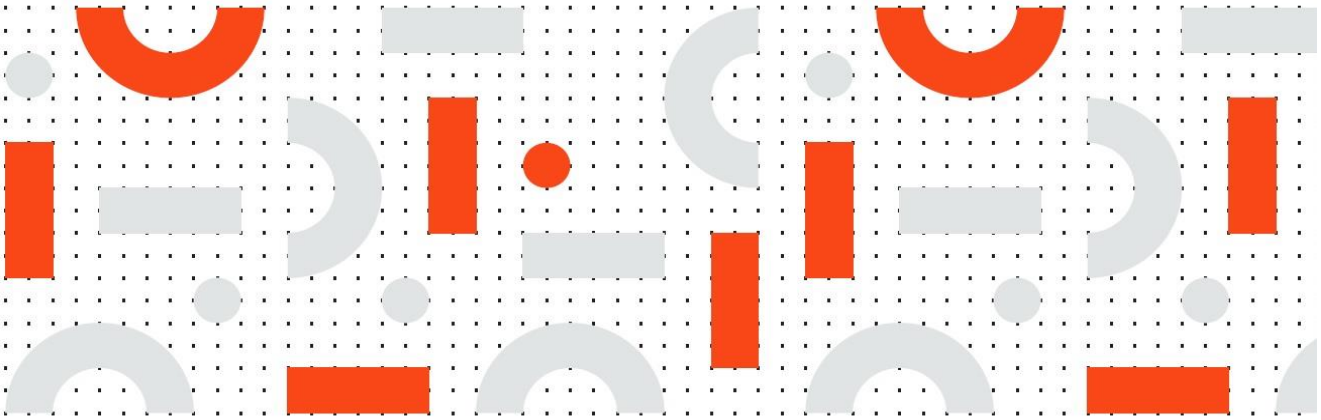


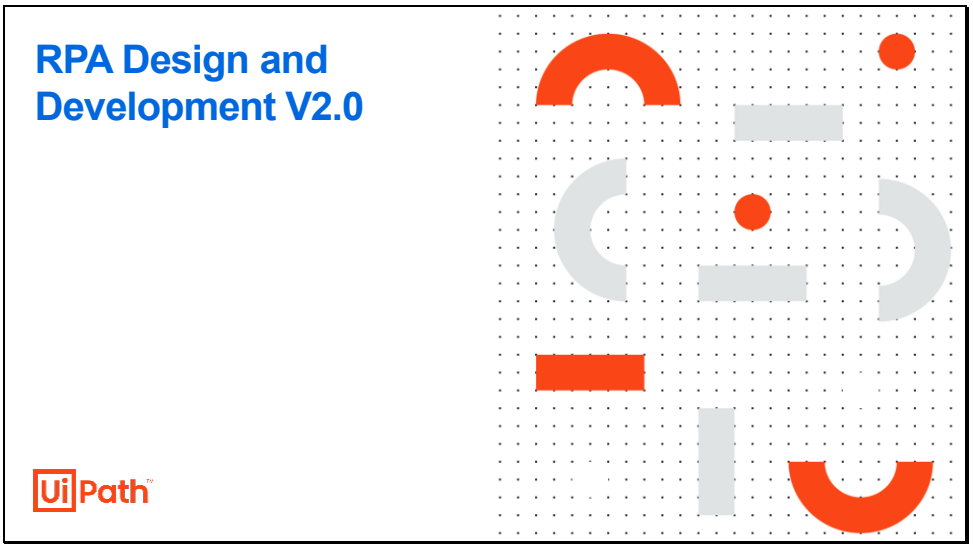
GUVI
tech deserves you

RPA Design & Development V2.0

Student Manual

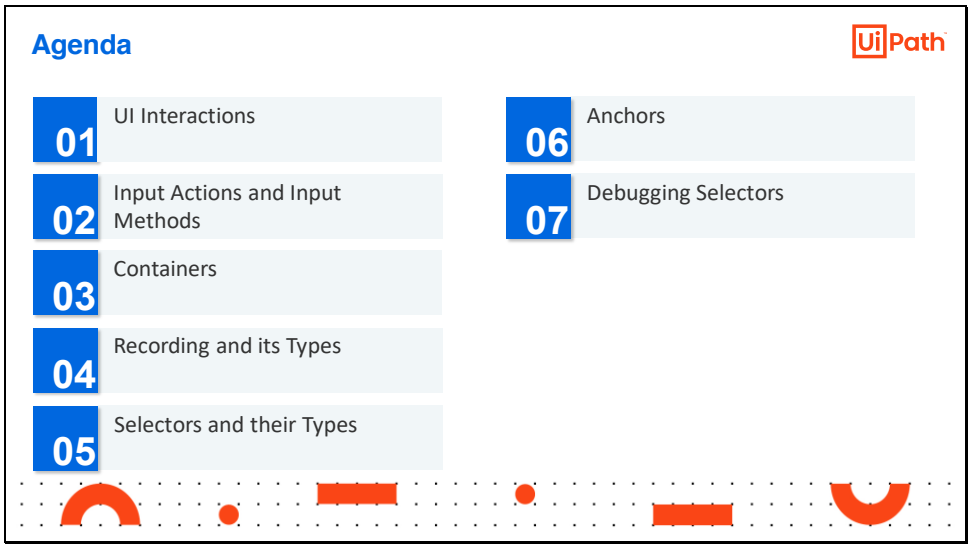


UiPath™ Learning Partner



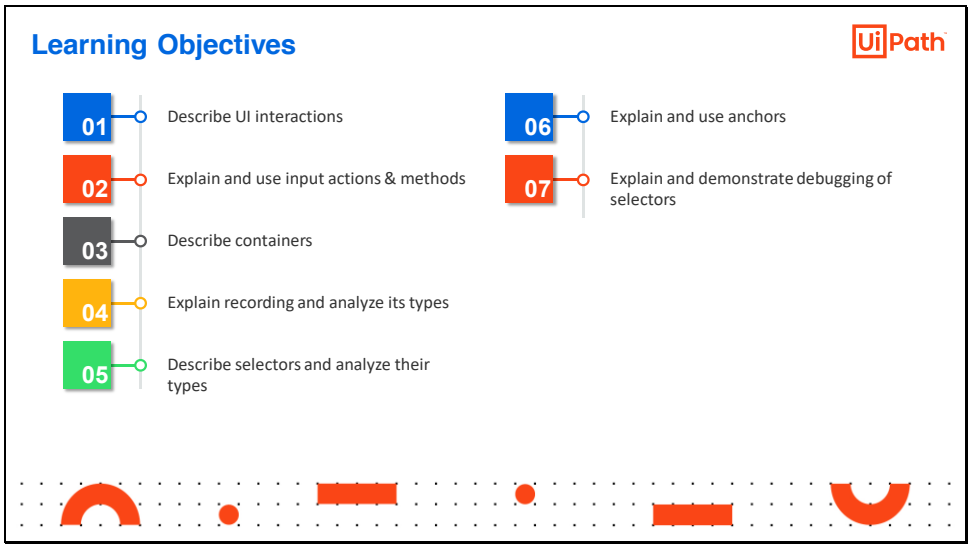
Welcome to 'RPA Design and Development Course'.

The fourth lesson of this course is Selectors.



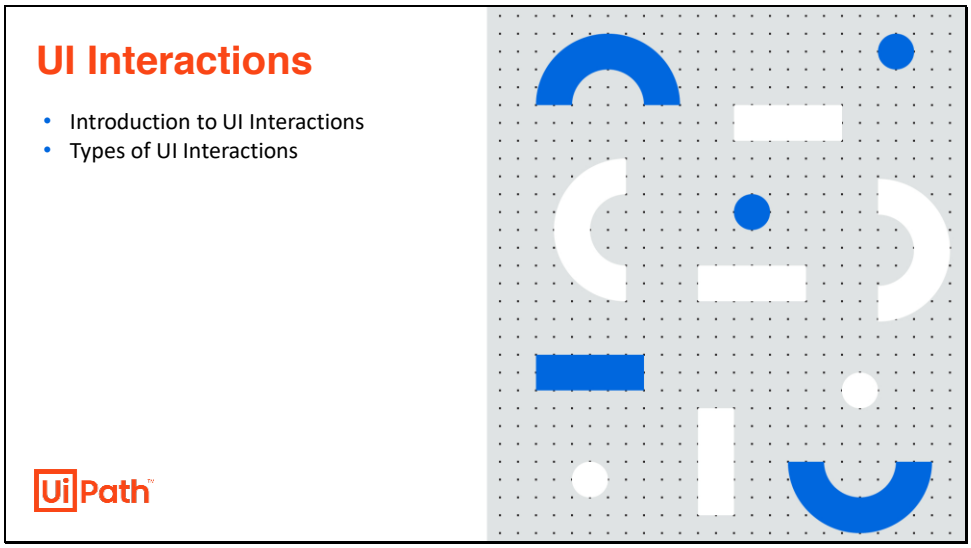
The agenda of this lesson is:

- UI interactions
- Input Actions and Input Methods
- Containers
- Recording and its types
- Selectors and their types
- Anchors
- Debugging Selectors

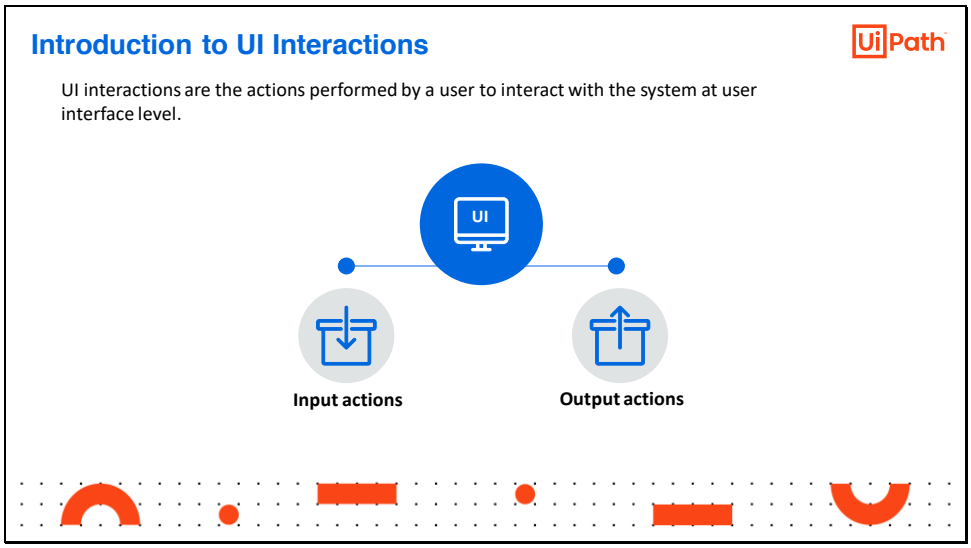


By the end of this lesson, you will be able to:

- Describe UI interactions
- Explain and use input actions and methods
- Describe containers
- Explain recording and analyze its types
- Describe selectors and analyze their types
- Explain and use anchors
- Demonstrate debugging of selectors

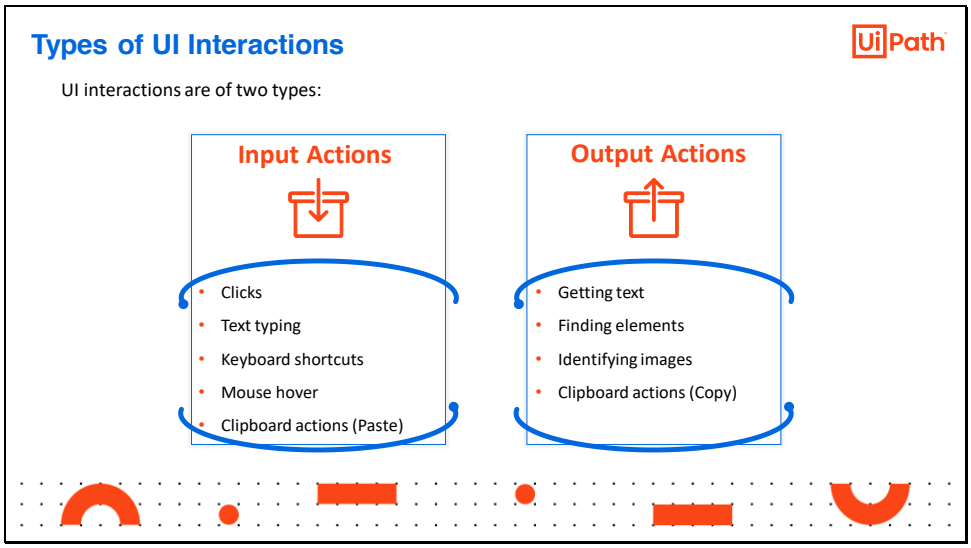


This section introduces UI interactions and their types.



UI elements refer to all graphical user interface pieces that constitute an application (such as windows, check boxes, text fields or drop-down lists). UI automation can be easily implemented by using UI interactions which are the actions performed by a user to interact with the system at the user interface level.

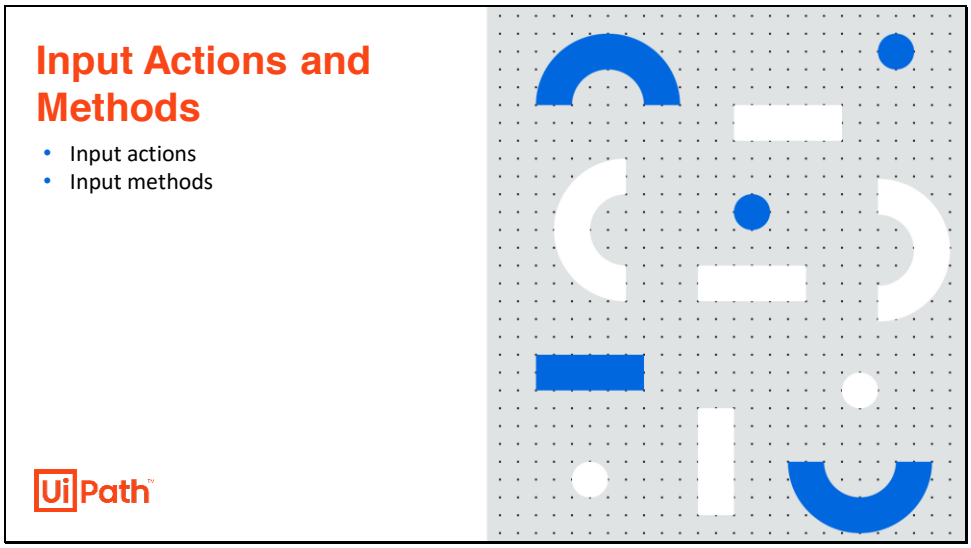
There are two types of UI interactions that can appear in an automation: **input actions and output actions**.



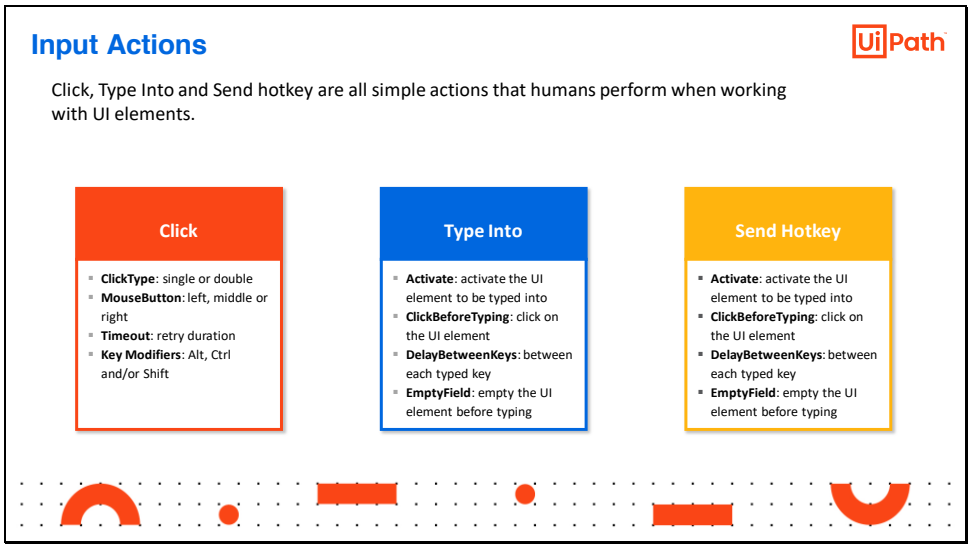
- **Input actions** are those actions through which the user sends data to an application (by clicking or typing) to produce an outcome. It means inserting data into an application. Every time the user clicks on a UI element, information is sent to the application and there is an outcome.
  These actions include:
  - Clicks
  - Text typing
  - Keyboard shortcuts
  - Mouse hover
  - Clipboard actions (Paste)

- **Output actions** are those actions through which the user takes out data from an application. It means reading data from an application. These actions include:
  - Getting text
  - Finding elements
  - Identifying images
  - Clipboard actions (Copy)

(Output actions and methods are explained in detail in the next lesson as this lesson focuses on Input methods).



This section introduces input actions and input methods in UiPath.



In UiPath, the main input actions are Click, Type Into and Send Hotkey which are also the main actions that a human user performs to input data in an application.

1. Click: Clicks a specified UI element.

- **Click Type**: Specifies the type of mouse click (single, double, up, down) used when simulating the click event. By default, single click is selected.
- **Mouse Button**: The mouse button (left, right, middle) used for the click action. By default, the left mouse button is selected.
- **Timeout**: Specifies the amount of time (in milliseconds) to wait for the activity to run before the SelectorNotFoundException error is thrown. (Default value= 30000 milliseconds).
- **Key Modifiers**: Enable the users to add a key modifier. The available options are: Alt, Ctrl, Shift, Win.

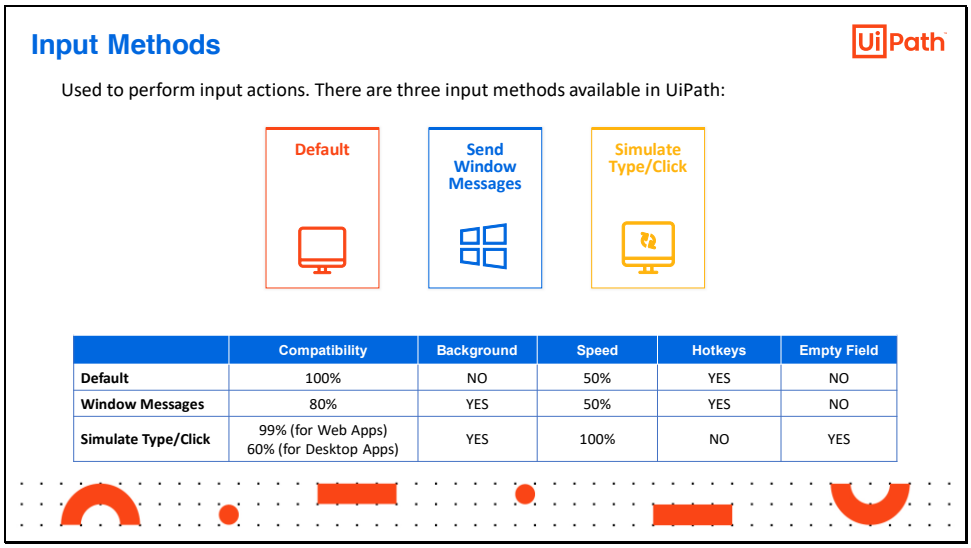
2. Type Into: Sends keystrokes to a UI element.

- **Activate**: On selecting this check box, the specified UI element is brought to the foreground and activated before the text is written.
- **Click Before Typing**: On selecting this check box, the specified UI element is clicked before the text is written.
- **Delay Between Keys**: Delay time (in milliseconds) between two keystrokes. (Default value= 10 milliseconds; Maximum value= 1000 milliseconds.)

- **Empty Field**: On selecting this check box, all previously existing content in the UI element is erased before writing the text.

  Sends keyboard shortcuts to a UI element.
- **Activate**: On selecting this check box, the specified UI element is brought to the foreground and activated before the text is written.
- **Click Before Typing**: On selecting this check box, the specified UI element is clicked before the text is written.
- **Delay Between Keys**: Delay time (in milliseconds) between two keystrokes. (Default value= 10 milliseconds; Maximum value= 1000 milliseconds.)
- **Empty Field**: On selecting this check box, all previously existing content in the UI element is erased before writing the text.

All the input actions share some (common) properties:
- **Delay**: Set a delay before or after the click.
- **WaitForReady**: Wait for the target to become ready by verifying certain application tags.



	Compatibility	Background	Speed	Hotkeys	Empty Field
Default	100%	NO	50%	YES	NO
Window Messages	80%	YES	50%	YES	NO
Simulate Type/Click	99% (for Web Apps) 60% (for Desktop Apps)	YES	100%	NO	YES

All the input actions are presented using clicks or types. A human uses the mouse and the keyboard for input. To replicate these actions, UiPath supports three input methods for performing input actions:

- **Default method:** This method captures the input given while interacting with the UI element of the screen. For example, selecting a UI element on the screen with the help of a mouse or keyboard.

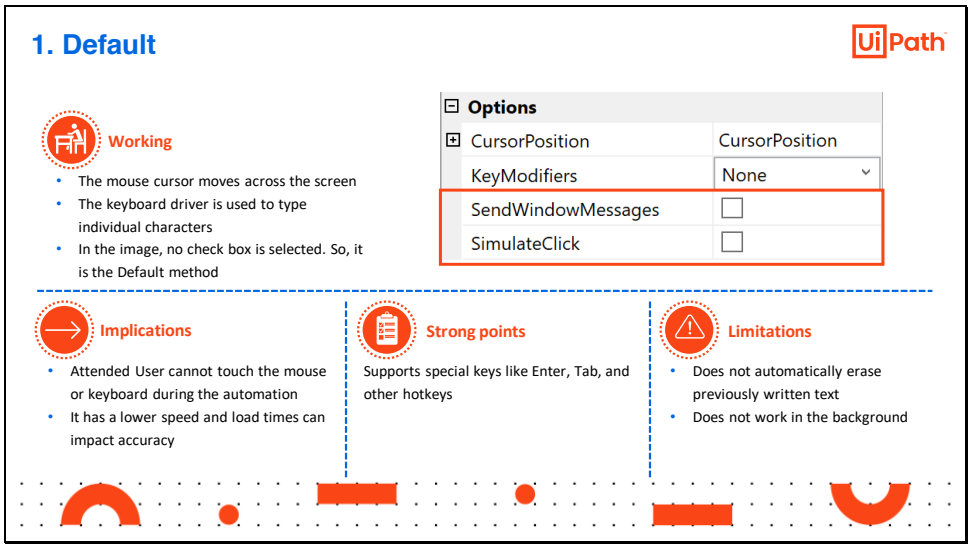
- **Send Window Messages:** This method performs the similar actions as the default method. As you can see on your screen it is less compatible with the application being used. This function captures actions performed using keyboard and mouse both.

- **Simulate Type/Click:** This method is also similar to default and send window messages. The points which differentiate it from the other two are speed and input source. This action is much faster than default and send window messages and it does not capture input given by keyboard.

These methods are similar in nature however, there are differences in their performance. The slide displays a comparison chart for these input methods.
Comparison based on different parameters for these methods:

- **Compatibility**: The default method is 100% compatible, given that it does exactly what a human would do. Simulate Type/Click does a great job for web automation but has significant limitations when it comes to compatibility with the desktop applications.
- **Background**: Both WindowMessages and Simulate work in the background, while the Default doesn't.
- **Speed**: Simulate method correctly identifies the controls used by the target apps, its speed is significantly higher than the speed of the other 2 methods.
- **Hotkeys**: Only Default and WindowMessages can send hotkeys.
- **Empty Field**: Empty field is a property of UI interacting activities. It is used to erase the data already present in the input area of the UI element. For example, a textbox containing previously filled data which is accessed repetitively in the code. When Default or Send Window Message method is used, the user has to ensure that 'empty field' property is checked for erasing the already present data. On the other hand, Simulate method empties the field by default.

The user must understand the differences between all three types of input methods and when to use them.

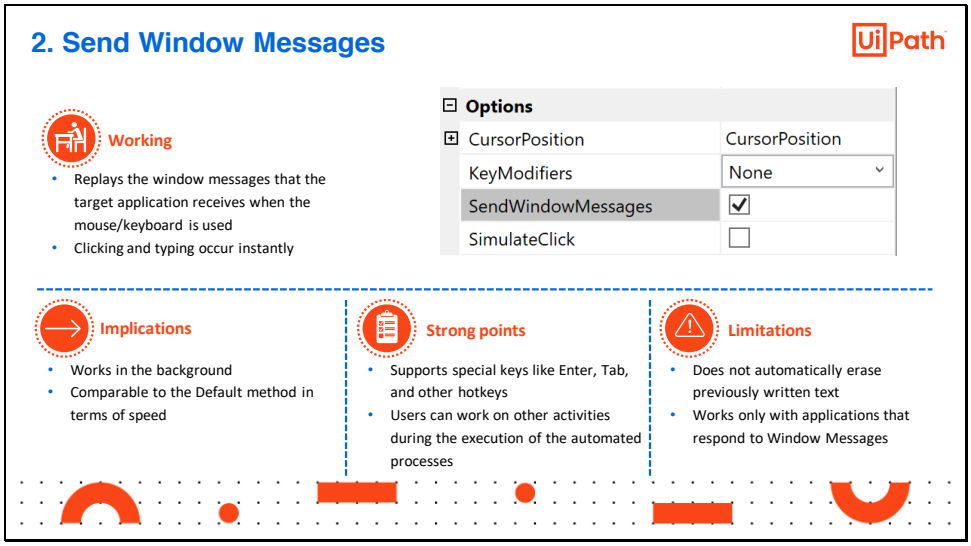


Each input action lets the user switch between the three methods.
'Simulate Type/Click/Hover' and 'SendWindowMessages' can be chosen by checking the corresponding box from the Properties panel; if none of the two boxes is checked, the Default input method is used.

It captures human actions performed with a mouse and keyboard.

It has an advantage over the other two methods that it uses the keyboard driver through which it can replicate the touch of every key, including special keys. Also, this method is 100% compatible with all applications.

In terms of limitations, it doesn't work in the background. The window must be active and on top of other applications or webpages. Moreover, if an editable field already contains text, it doesn't erase it, instead it starts writing after it. Also, since typing each character and moving the mouse across the screen are not the fastest ways to solve a task, this method is the slowest of all the three.



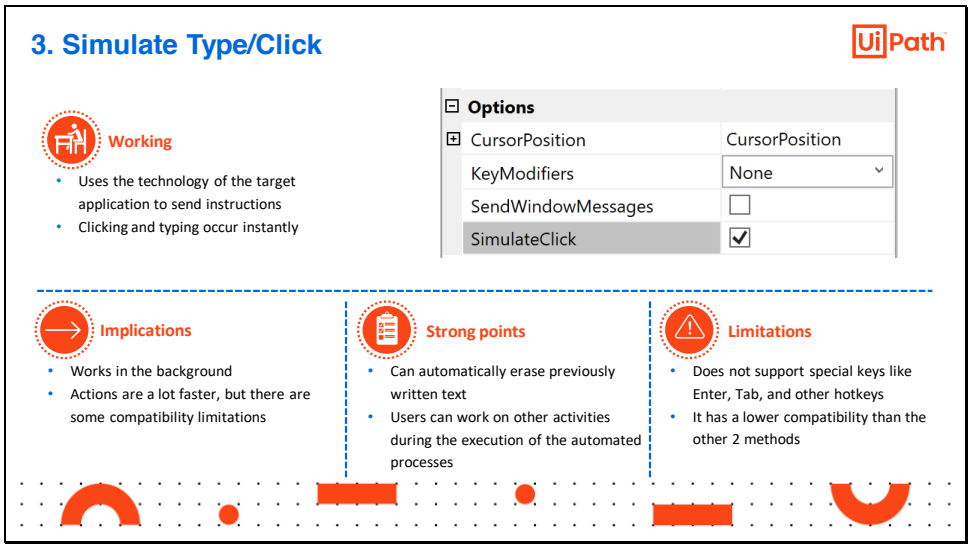
The Send Window Messages input method can be activated from the Properties tab, by checking the 'SendWindowMessages' box.

This method also captures human actions performed with the help of a mouse or keyboard.

The advantage of this method is that it works in the background. As a result, the mouse doesn't move on the screen, and typing isn't done by keyboard simulation. Instead, the clicks and the typing appear instantly.

As the window messages support hotkeys and shortcuts, this method can be used to send special characters in applications.

Just like the Default method, it doesn't erase the text that is already typed in the target UI element by default.

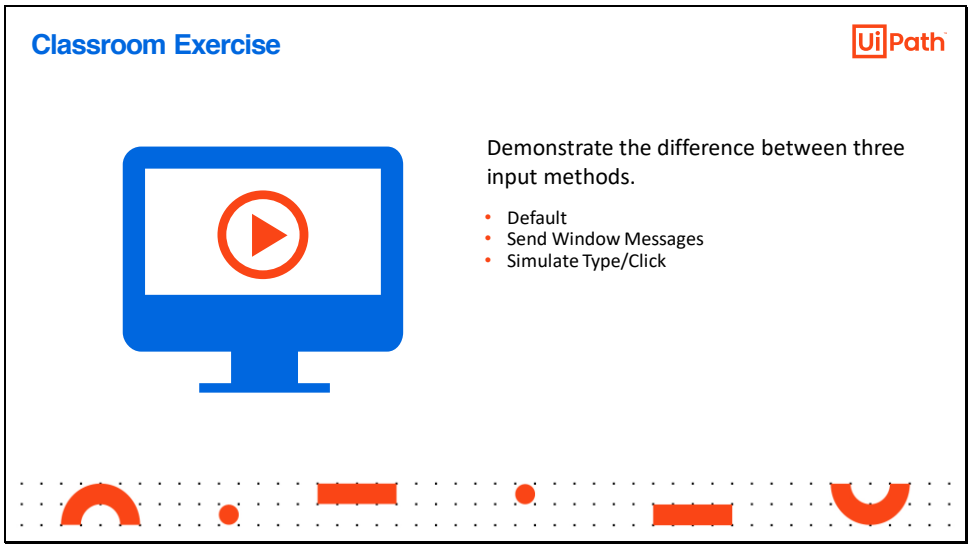


The Simulate input method can be activated from the Properties tab by checking the Simulate Type/Simulate Click box.

This method captures human actions performed with the help of a mouse.

This method uses the technology of the target application for sending instructions. It is the fastest of the three input methods but only works with applications whose controls react to the commands sent. This method can function in the background.

By default, this method erases the existing text in a UI Element but doesn't support hotkeys and special keys. In general, it is used for web automation, where its compatibility is close to 100% (vs desktop applications, it is near around 60%).

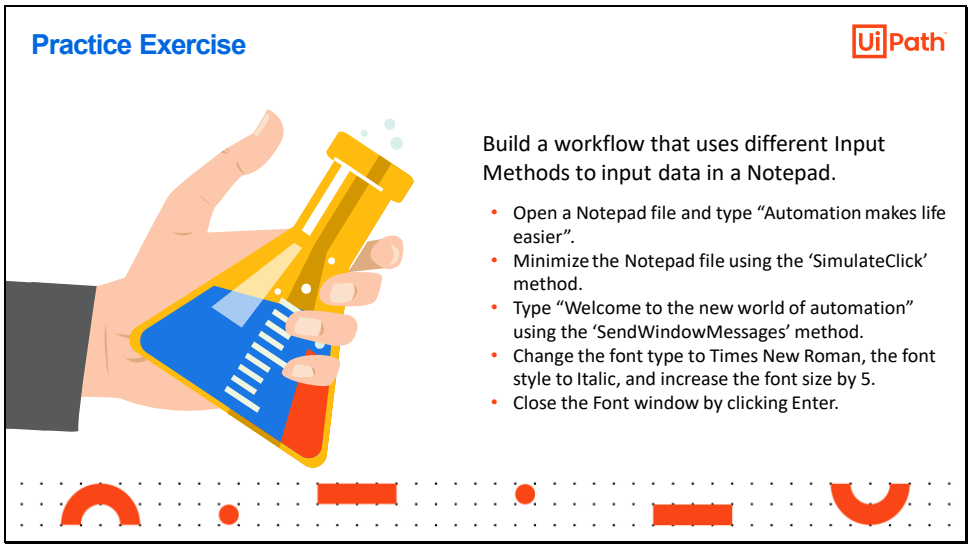


Demonstrate the difference between three input methods – Default, Send Window Messages, and Simulate Type/Click.

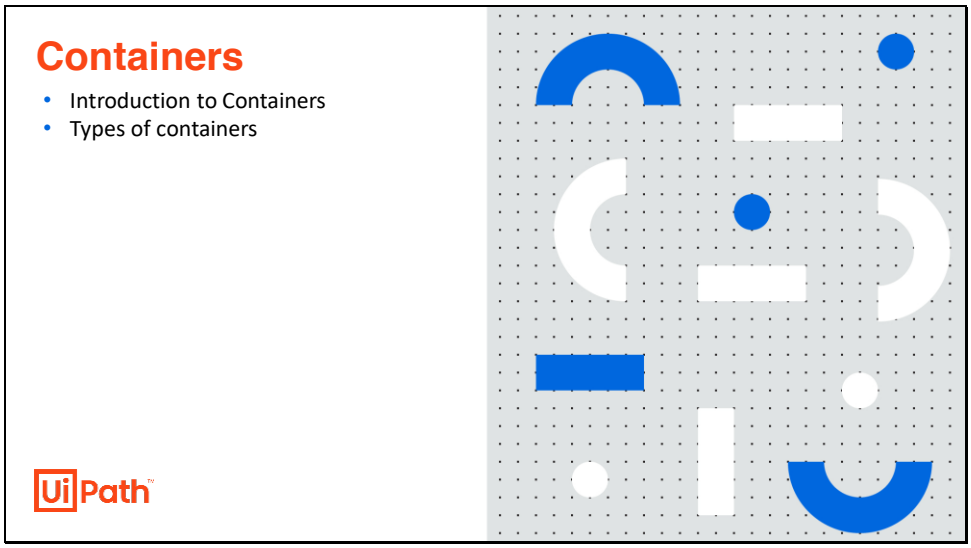
1. Open a Notepad application on your desktop.
2. Go to UiPath Studio, search Open Application activity from the activities panel, and drag it in the Designer panel.
3. Click "Indicate window on screen" link and indicate the already open Notepad window.
4. Search Type Into activity and drag it in the Do container.
5. Click "Indicate element inside window" link and indicate the notepad text editor area.
6. Enter a text, "Automation makes life easier", in the Type Into activity.
7. Drag and drop Click activity below Type Into activity.
8. Click "Indicate element inside window" and indicate the minimize button of Notepad window.
9. Take another Type Into activity, and place it below Click activity.
10. Click "Indicate element inside window" and indicate the notepad text editor area.
11. Enter a text, "Let's learn it".
12. Click "+" icon and select enter.
13. Copy the special key string for enter and move it before the text.
14. Take another Click activity, paste it below Type Into activity.
15. Click "Indicate element inside window" link and click on the notepad icon in the taskbar to restore it.
16. Run the process.

By default input process does not erase pre-existing texts and supports special key. But, writing in the background does not work.

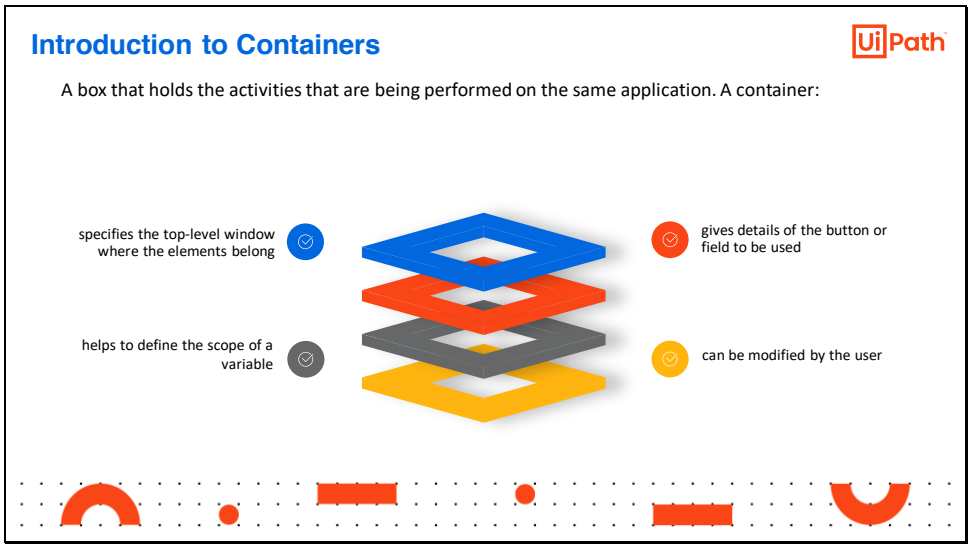
Next, select SendWindowMessages property for the Type Into activities from properties panel and rerun the process. The second text gets written on a new line while notepad was minimized. Here, input method works in the background, supports special keys, and does not erase pre-existing texts.
Next, select SimulateType property and rerun the process. The second text gets written after replacing first text while notepad was minimized. The text shows the special key for enter – [k(enter)]. Here, input method works in the background. But it does not support special keys and erases pre-existing texts.



Build a workflow that uses different Input Methods to input data in a Notepad.
• START
• Use Open Application activity to indicate a Notepad file
• Use Type Into activity to enter "Automation makes life easier".
• Minimize the Notepad window using the 'Simulate Click' method in Click activity
• Use Type Into activity to enter "Welcome to the new world of automation" using the 'Send Window Messages' method
• Use Send Hotkey activity to send "Ctrl + A".
• Step 6: Change font type to Times New Roman, font style to Italic and increase the font size by 5.
• Step 7: Close the Font window by clicking Enter.
• STOP



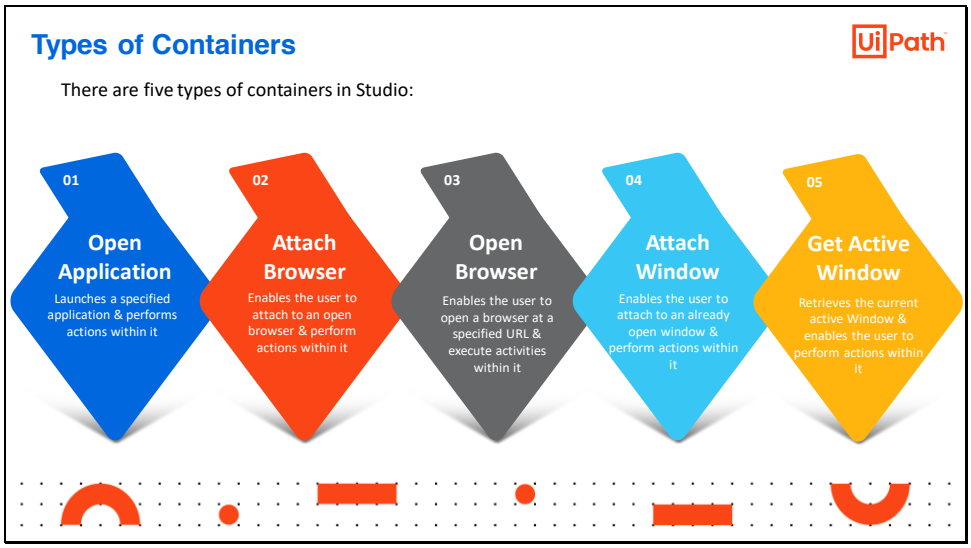
This section explains containers and its types.



The Target UI Automation activity identifies the UI element the activity works with.
Target is composed of multiple pieces: **the container, selector,** and **clipping region,** to ensure that a UI element is correctly identified.

A container is a box that holds the activities that are being performed on the same application. It specifies the top-level window where the elements belong. It gives more context of the intended button or field, so that different windows or different areas of the same app can be identified. A container also helps to define the scope of a variable.

Containers are automatically generated, but users can make changes to them in the Properties panel.

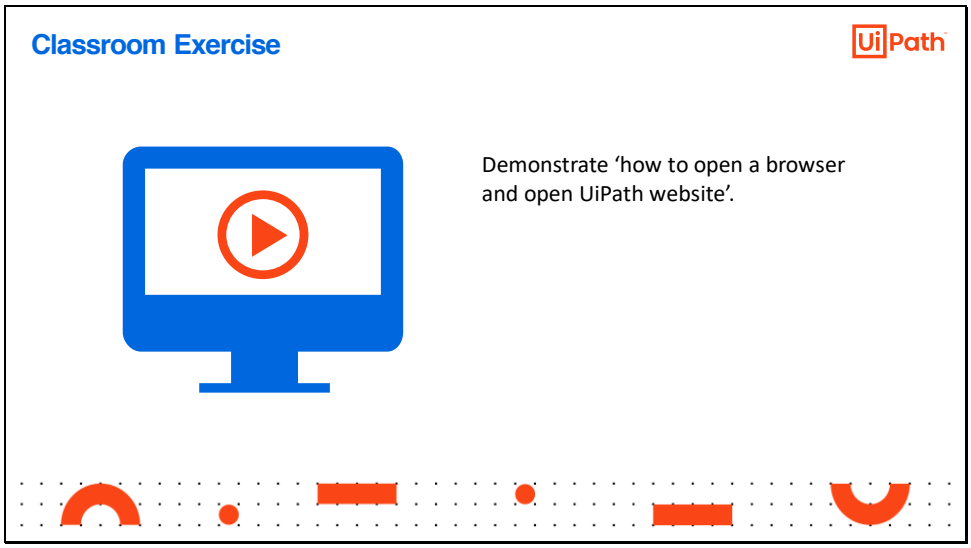


The **Container Usage** activity checks if consecutive activities in the automation are using the same top-level selectors instead of using a container with a single selector.

There are five types of containers in Studio. These are:

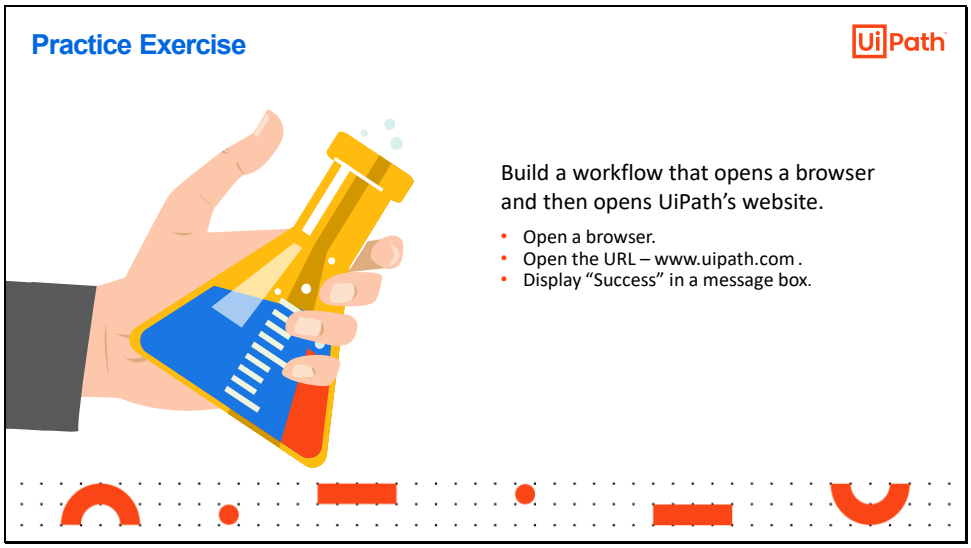
- Open Application: Launches a specified application and performs multiple actions within it. It can also pass a list of arguments to the application.
- Attach Browser: Enables the user to attach to an already open browser and perform multiple actions within it. This activity is automatically generated when using the Web recorder.
- Open Browser: Enables the user to open a browser at a specified URL and execute multiple activities within it.
- Attach Window: Enables the user to attach to an already open window and perform multiple actions within it. This activity is automatically generated when using the Desktop recorder.
- Get Active Window: Retrieves the current active Window and enables the user to perform multiple actions within it.

Scopes such as Open Browser or Open Application must be used when performing multiple actions on the same UI element, as this method is more reliable instead of targeting each activity on the same selector.



Demonstrate 'how to open a browser and a website' by using Open Browser activity.

- Open UiPath.
- Drag and Drop Open Browser activity in the main workflow.
- Enter the website URL "http://www.uipath.com" in the text area.
- Run the program. It will open UiPath's website.

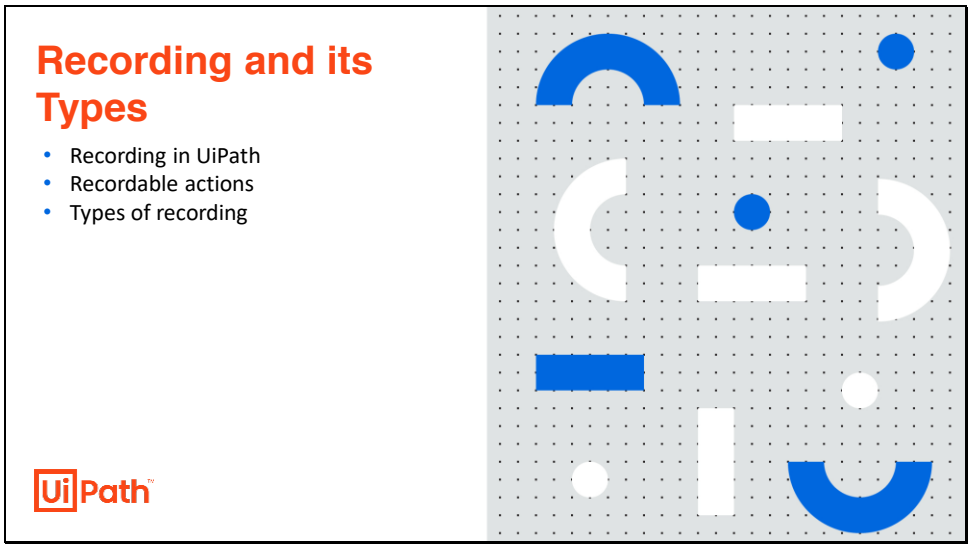


Build a workflow that opens a browser and then opens UiPath's website.

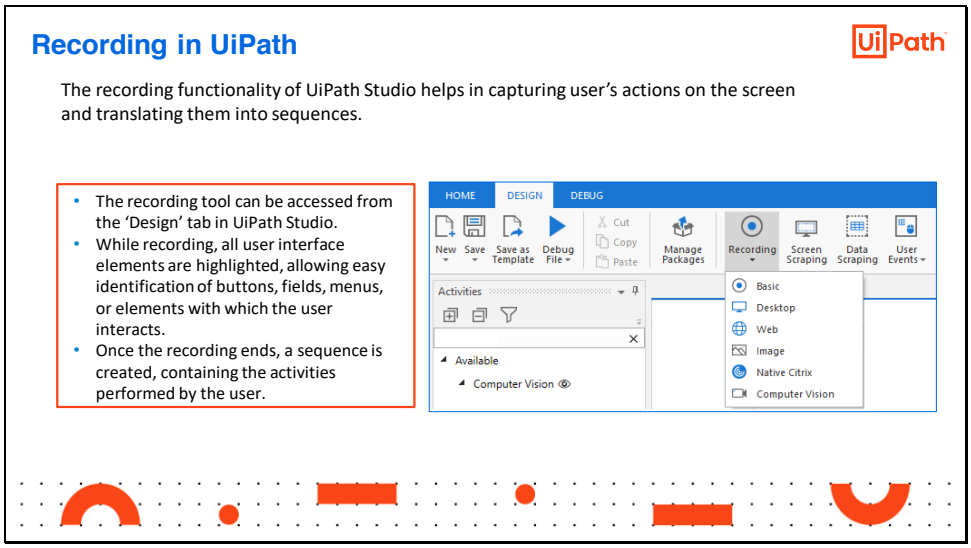
- Open a browser.
- Open the URL – www.uipath.com .
- Display "Success" in a message box.

Algorithm

- START
- Use Open Browser activity and enter website URL – "www.uipath.com"
- Use Message Box activity and enter text "Success"
- STOP



This section explains recording in UiPath and its types in detail.

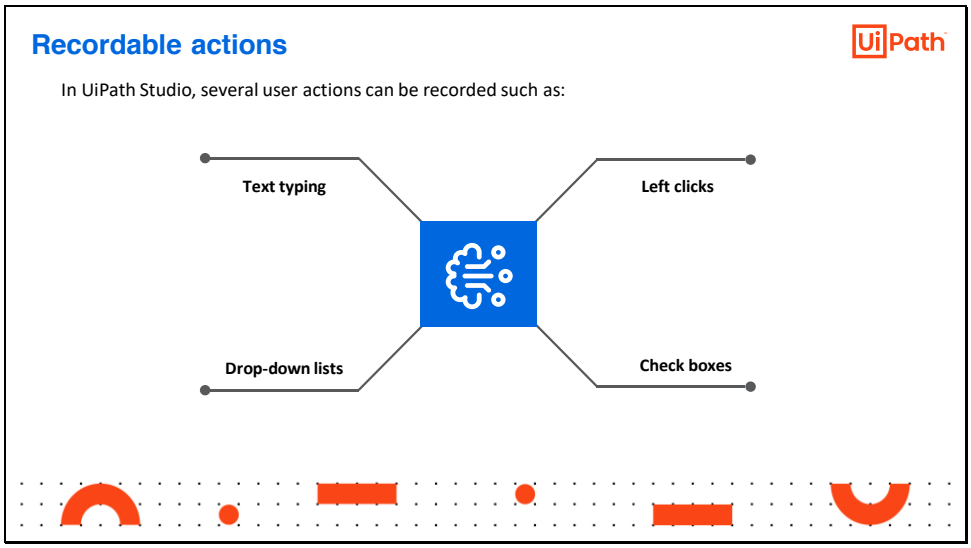


UiPath Studio recording is a technique which helps in capturing the manual actions of the user performed on the screen and translating them into sequences.

Recording is a crucial tool in UiPath Studio as it captures the step-by-step activity of the user. It automatically generates the sequence of all the workflows which are needed for automation. Recording helps in identifying the elements on the screen, reading data from display, and so on. Recording helps users to save a lot of time when automating the business processes.

All user interface elements are highlighted while recording to ensure that correct buttons, fields or menus are selected. The recorder is capable of working with different types of applications, and different types of environments.

Interactions with UI elements yield informative screenshots in the automation. These can be changed, hidden, removed or shown in full size by selecting the respective action from the **Options** menu. All screenshots are automatically saved as .png files in the same location as the project, in a separate folder named ".screenshot".



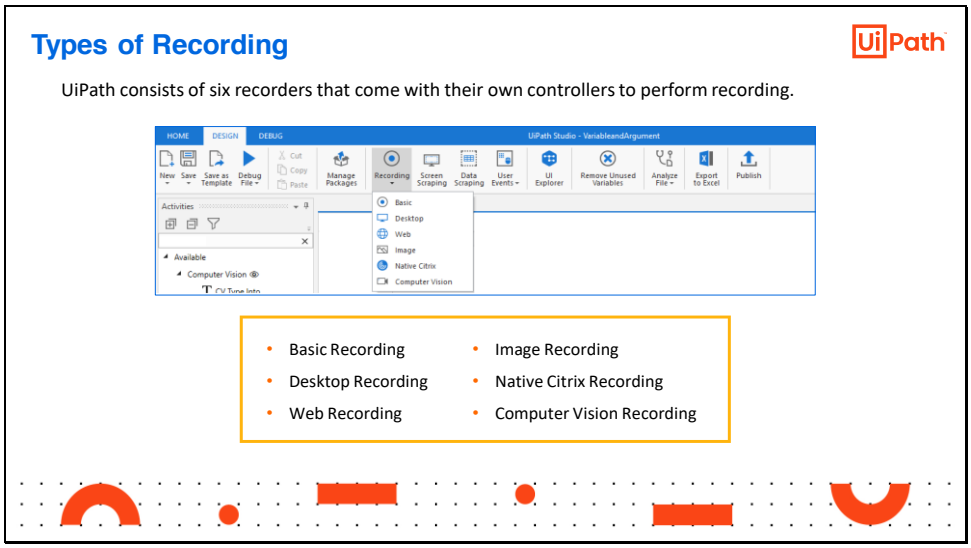
In UiPath Studio, there are several user actions that can be recorded. These include:

- Left clicks on buttons
- Checkboxes
- Drop-down lists
- Text typing

However, there are some user actions that are still not supported by the recording interface. These include keyboard shortcuts (for example, Ctrl+V for pasting data), modifier keys (Ctrl, Shift, and Alt), mouse hovers, etc.

The following **Keyboard shortcuts** are available for use while recording:

- F2 – pauses the recording for 3 seconds. A countdown timer is displayed in the bottom left corner of the screen. Can be useful with menus that automatically hide.
- Esc – exits the automatic or manual recording. If Escape key is pressed again, the recording is saved as a sequence, and you return to the main view.
- Right-click – exit the recording.

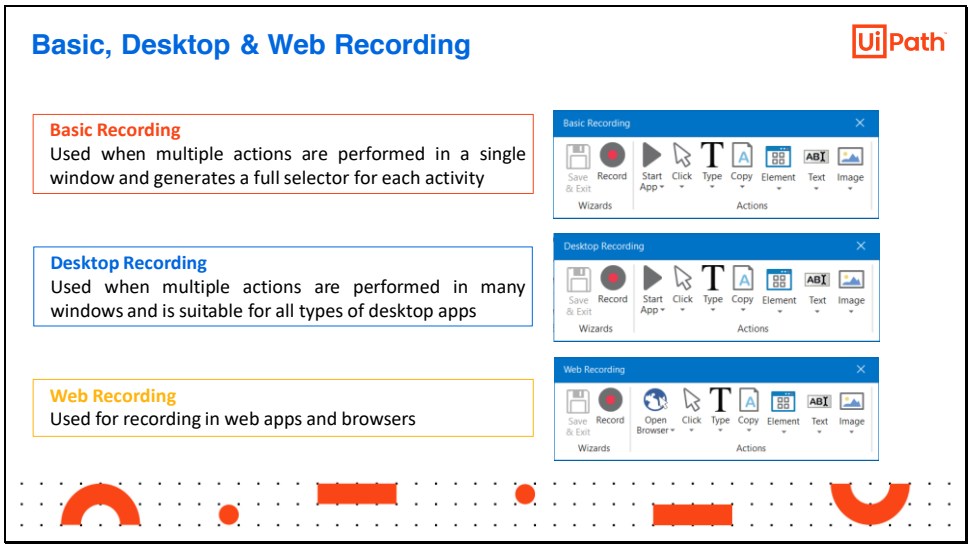


Recording:

In UiPath, there are six types of recorders, and each comes with its own controllers to perform specific recording actions. These are:

- Basic Recording
- Desktop Recording
- Web Recording
- Image Recording
- Citrix Recording
- Computer Vision Recording

These are discussed in detail in the subsequent slides.



A. Basic Recording:
This recording is used when multiple actions are performed in a single window. It generates a full selector for each activity with no container. The resulting automation is slower than the one that uses containers and is suitable for single activities. It is slower as compared to the Desktop recording.

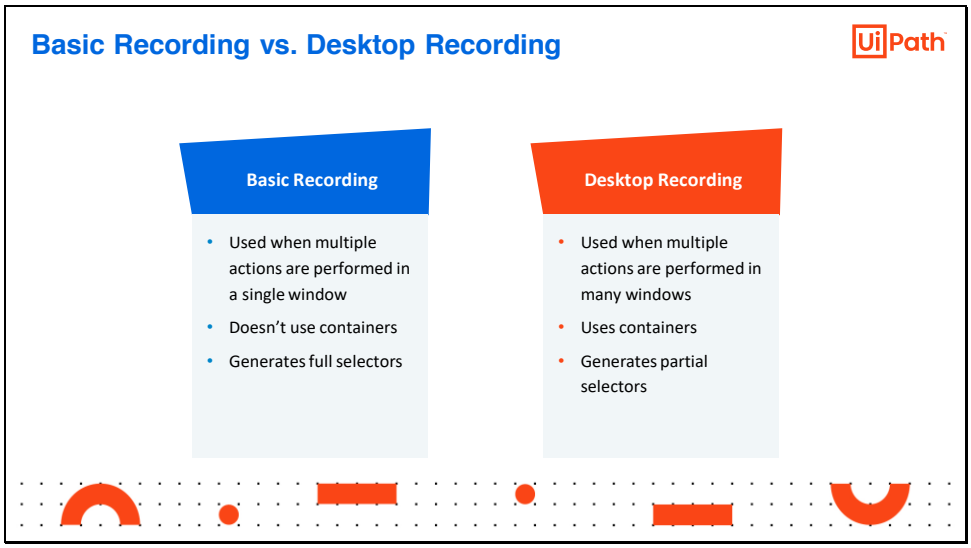
B. Desktop Recording:
This recording is used when multiple actions are performed in many windows. It is suitable for all types of desktop apps. It is faster than the Basic recording and generates a container (with the selector of the top-level window) in which activities are enclosed, and partial selectors for each activity.

C. Web Recording:
It is designed for recording in web apps and browsers, generates containers and uses the **Simulate Type/Click** input method by default.

All these recordings enable the user to:

- Automatically record multiple actions performed on the screen such as Click, Type Into, Select Item and Check.
- Manually record single actions, such as:
    - Starting or closing an application (Basic & Desktop recording) / Starting or closing web browser (Web recording)
    - Clicking an interface element

- Selecting an option from a drop-down list
- Selecting a check box
- Simulating keystrokes or keyboard shortcuts
- Copying text from a UI element or performing screen scraping
- Looking for elements or waiting for them to vanish
- Finding an image
- Activating a window



Difference between the Basic recording and Desktop recording:

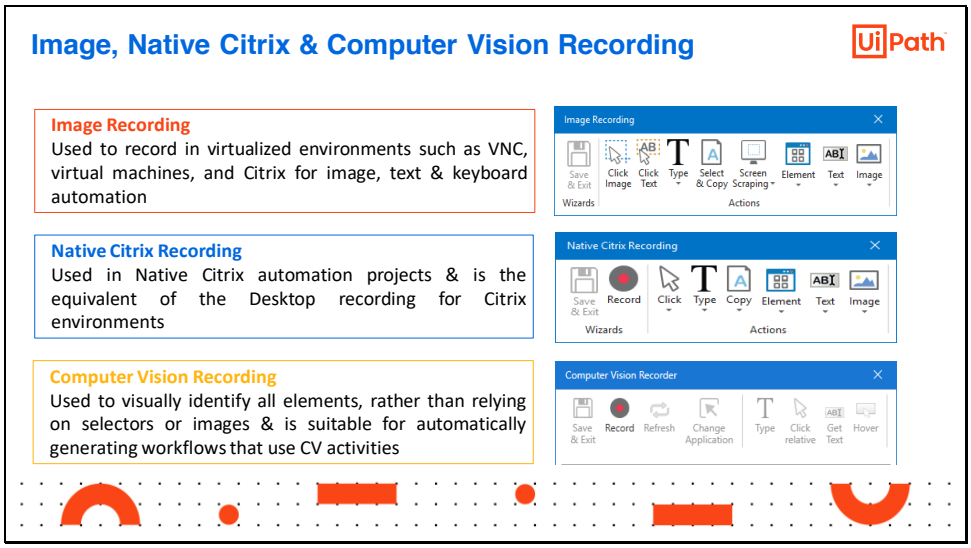
Basic Recording:

- It is used when multiple actions are performed in single window.
- It does not use containers which means other window of the same application may interfere with the workflow.
- It generates full selectors.

In some cases, multiple windows are identical. Suppose the user opens two untitled notepad windows at the same time, then UiPath will perform all actions inside the first opened window because the title of both notepad windows is the same (i.e., "untitled"). This may not be a good practice when working on files having identical names.

Desktop Recording:

- It is used when multiple actions are performed in many windows. It nests all activities inside the Attach Window container.
- It uses containers to ensure that other windows of the same application don't interfere with the workflow. The top-level window selectors are inside the Attach Window.
- It generates a partial selector for activities inside the Attach Window container.



D. Image Recording:

It is used to record in virtualized environments (such as VNC, virtual machines, Citrix, and more) or SAP. It permits only image, text and keyboard automation, and requires explicit positioning. Image recording enables the user to:

- Click an image or text
- Simulate keystrokes or hotkeys
- Select and copy text from a window
- Scrape UI elements
- Look for elements
- Find an image or wait for it to vanish

E. Native Citrix Recording

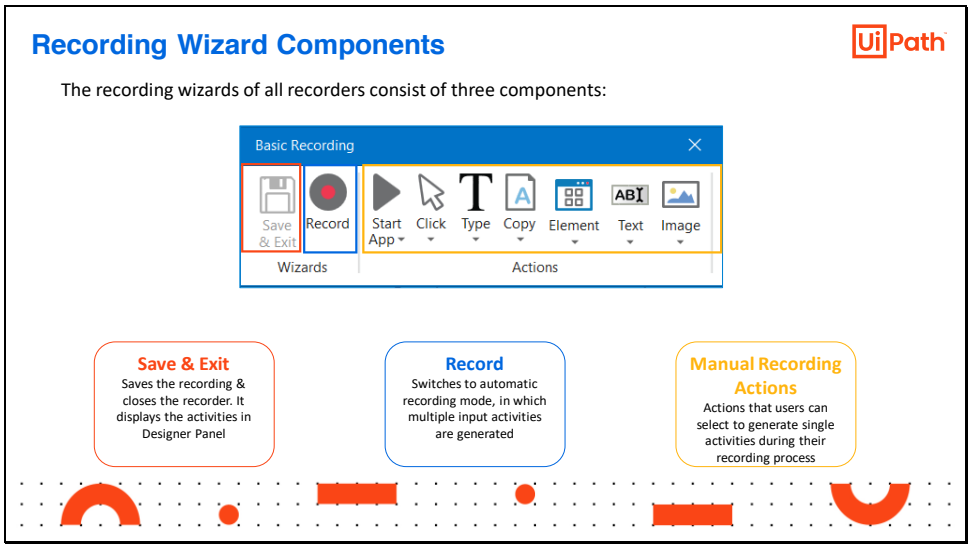
It is the equivalent of the Desktop recording, but for Citrix environments. This is used in Native Citrix automation projects to identify UI elements in Citrix Apps and automate them. This recording enables the user to automatically record multiple actions performed on the screen and manually record single actions, such as clicking an interface element, simulating keystrokes or keyboard shortcuts, copying text from a UI element or performing screen scraping, etc.

F. Computer Vision Recording

Computer Vision (CV) is a feature that allows Robots to "see" the screen and visually identify all the elements, rather than relying on selectors or images. It enables human-like recognition of user interfaces using a mix of Artificial Intelligence, OCR, fuzzy text-matching, a new image-

matching algorithm for icons, and an anchoring system to tie it all together. Computer Vision solves the challenge of reliably automating Virtual Desktop Environments.

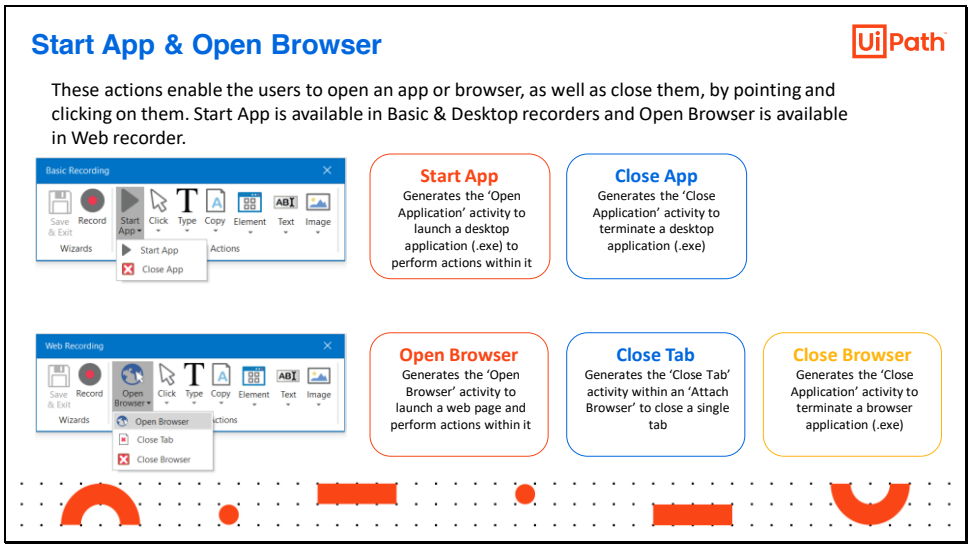
Once the user installs the Computer Vision activities pack, the Computer Vision Recorder wizard becomes available in the Ribbon. This recording is suitable for automatically generating workflows that use the CV activities and offers the full spectrum of capabilities. It enables the user to indicate the application you want to automate, which is used as a target for all subsequent CV activities. Once an application is indicated, the selected window is sent to the CV server, where it is processed, and the wizard enters recording mode.



Components of recording wizard in UiPath:

The recording wizards of all recorders consist of three components:

- **Save & exit**: This option is enabled only when the recorder is on. It closes the recorder and reverts to the 'designer' window, displaying the automatically generated activities.
- **Record**: It starts the automatic recording mode and captures user actions. On completion multiple input activities are generated. The user will see a blue screen over the area where the action is captured and yellow outlines for selectors.
- **Manual recording actions**: The remaining actions in the wizard are manual recording actions. These may be common or different for different recorders. These are discussed in the coming slides.



Start App is available for Basic & Desktop Recording. It has two types of activities:

- **Start app** is used to open the application. It is similar to the 'Open Application' activity of UiPath studio. To use this, select **"Start App"** and then click on the desired application to open it.

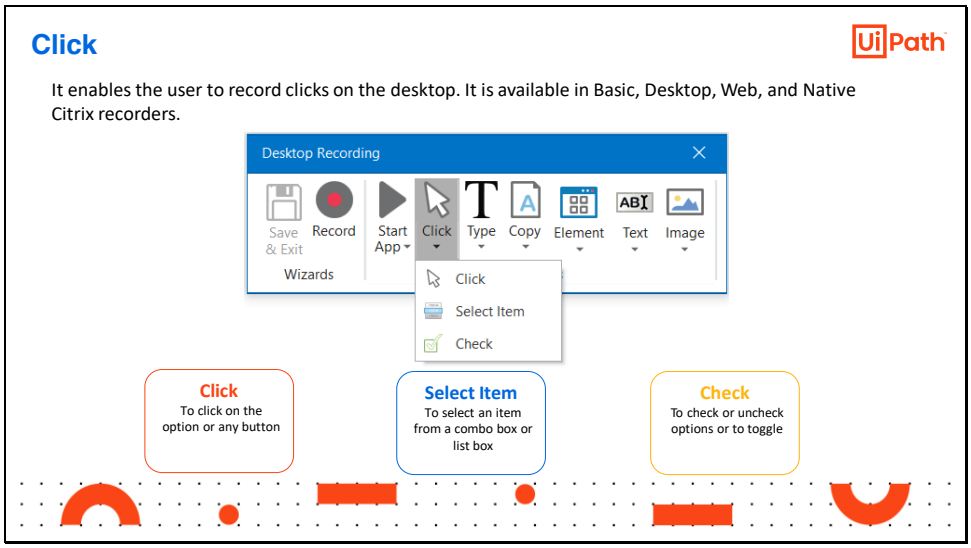
- **Close app** is used to close the already open application. It is similar to the 'Close Application' activity of UiPath studio. To use this, select **"Close App"** and then click on the desired application to close it.

Open Browser is available for Web Recording. It has three activities:

- **Open Browser:** This is used to open a browser. The browser can be chosen (for example Internet Explorer, Firefox or Chrome) or the URL can be specified inside the quotation marks "***", or the user can store the URL inside the string variable and pass that inside it.

- **Close Tab:** Closes the Browser tab specified by user.

- **Close Browser:** Closes the already open browser window.

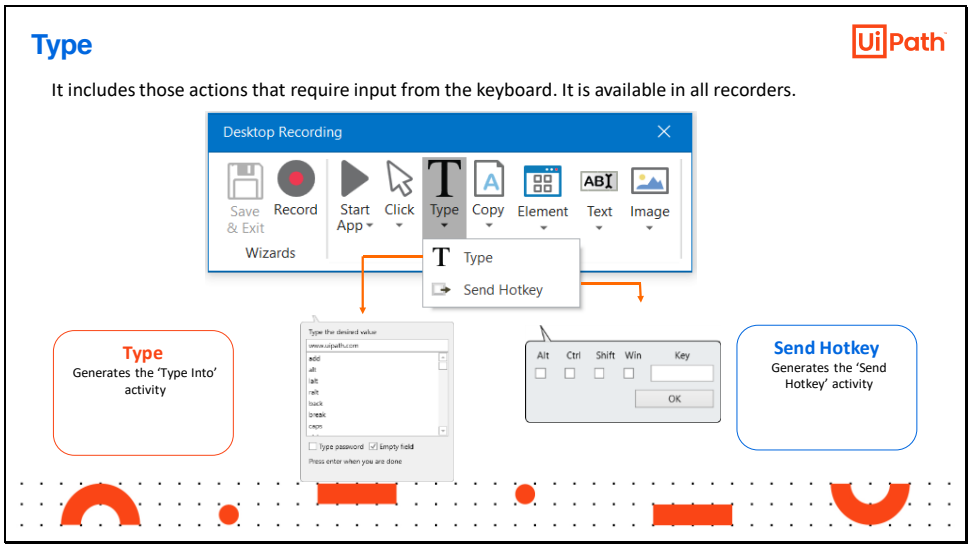


Click: Enables the user to record clicks on the desktop or a running application, select an option from a drop-down list or combo box, and select a check box or a radio button.

It is available in Basic, Desktop, Web, and Native Citrix recorders. The activity is supported by the automatic recorder and it doesn't recognize right clicks.

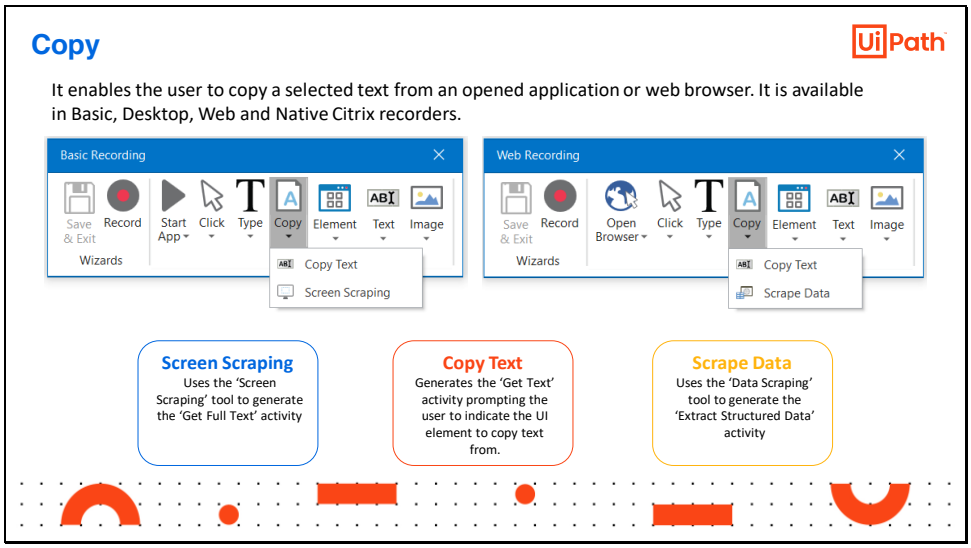
There are 3 options in the 'click' activity .

- **Click -** This activity is used to click on the option or any button.
- **Select Item -** This activity is used to select an item from a combo box or list box.
- **Check -** It is used to check or uncheck options or to toggle.



Type: This includes those actions that require input from the keyboard, such as keyboard shortcuts and key presses. To achieve this, two pop-up windows are used to retrieve the keyboard input.

- **Type:** generates the 'Type Into' activity prompting the user for their desired value.
- **Send Hotkey:** generates the 'Send Hotkey' activity prompting to define a Alt-, Ctrl-, Shift-, or Win + Key to send it to an application.



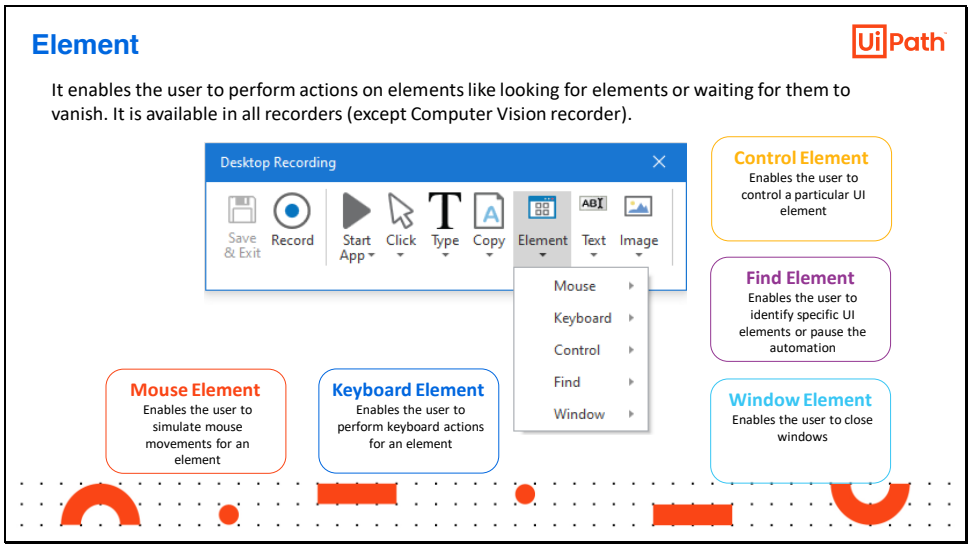
Copy: Enables the user to copy a selected text from an opened application or web browser, so that it can be used later in the project. Screen scraping is also available under the **Copy** menu, as it enables the user to extract images and text from an app or browser.

The Copy activity contains:

- Copy text (same in all three recorders): Generate Get Text activity & recognize the text and copy it for further use as it is stored in a variable.
- Screen Scraping (in Basic, Desktop and Native Citrix Recorder). It is used to read the data from the screen. It is capable of scraping both visible and non-visible UI elements from the screen. The screen scraping tool can automatically detect the position of the text on the screen. It has three methods:

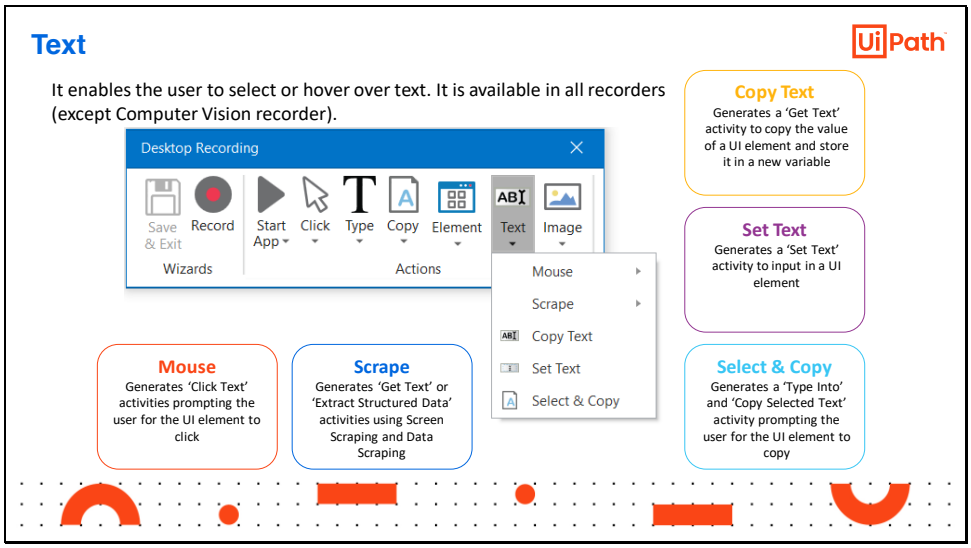
    1. Full text
    2. Native text
    3. OCR

- Scrape Data (in Web Recorder): It allows us to extract any structured data from the web, any application, document, spreadsheet, etc.



Element:

- **Mouse Element**: Enables the user to simulate mouse movements for an element that cannot be recorded but give access to more functionalities, such as right-clicking, hovering or double-clicking.
- **Keyboard Element**: Enables the user to perform keyboard actions for an element. It consists of Type (generates the 'Type Into' activity) and Send Hotkey (generates the 'Send Hotkey' activity) options.
- **Control Element**: Enables the user to control a particular UI element. It consists of Select Item and Check options.
- **Find Element**: Enables the user to identify specific UI elements or pause the automation until a particular window closes or an UI element is no longer displayed.
- **Window Element**: Enables the user to close windows. Studio does this by hooking in the operating system to make sure the application is closed.



Text: Enables the user to select or hover over text to make tooltips visible for scraping, right-click to make the context menu visible, copy and paste text and many others. It is available in all recorders and is known as Get Text in Computer Vision recording.

- **Mouse:** It includes 4 categories for different methods of using the mouse in relation with a text:
    1. Click
    2. Double-click
    3. Right-click
    4. Hover
- **Scrape**: Used for data scraping. It consists of three categories:
    1. Screen Scraping
    2. Data Scrape
    3. Scrape Relative
- **Copy Text:** This is the same activity as the one created under the Copy category.
- **Set Text:** This activity is used to identify a UI element and send a text to it. It is very similar to Type Into, the main difference between both of them is that you can assign keystrokes in Type Into, but in Set Text, you cannot do that.
- **Select & Copy:** This is used to generate two activities with a single step:
    - Select a text in UI element
    - Copy the text and store it in another variable

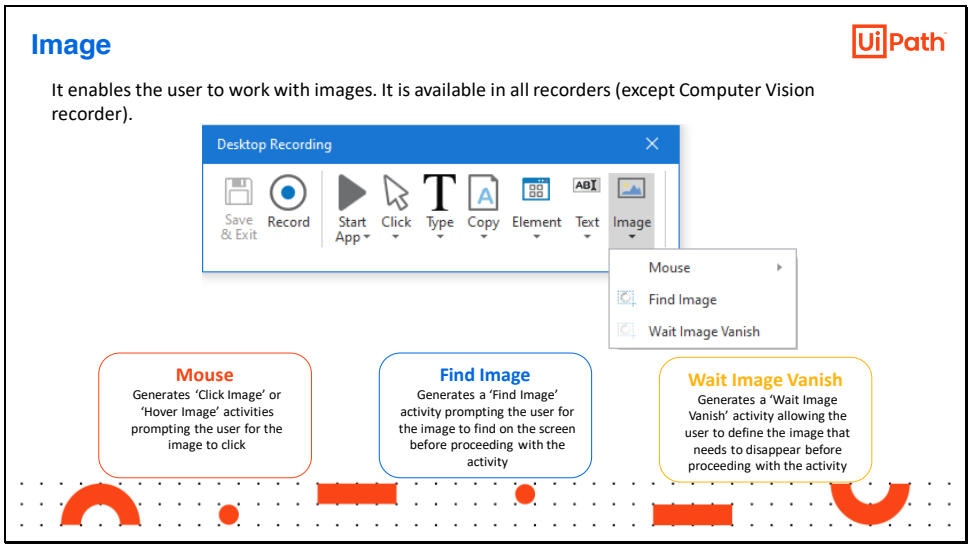


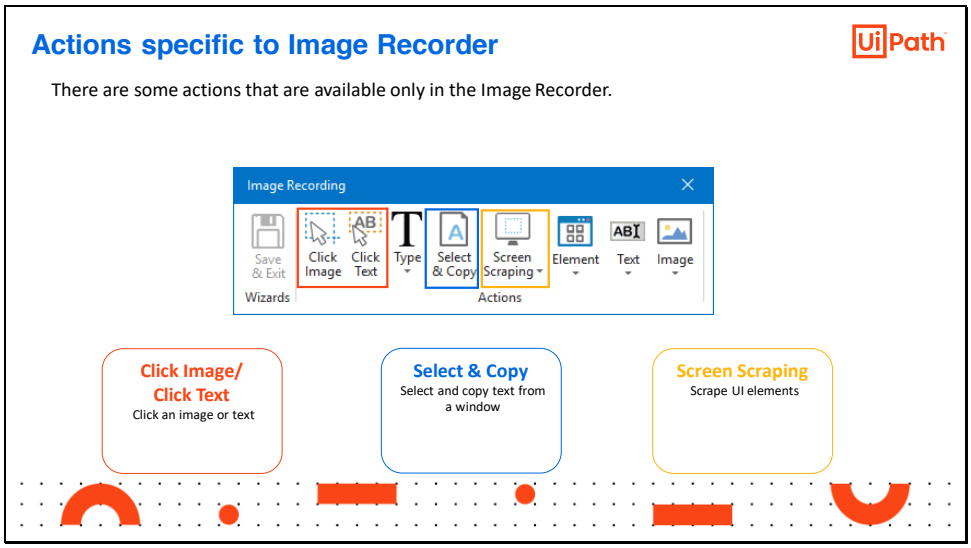
Image: Enables the user to wait for a specified image to disappear, to find a specific image in an app or website, right-click or hover over an image and others (Useful with UI elements that are saved as graphics)

It brings together several events to recognize images and treat them as triggers:

- **Mouse:** The Mouse sub-category contains different methods of using the mouse in relation with images, click, double-click, right-click and hover. By using these properties, these activities can be customized to use modifier keys.

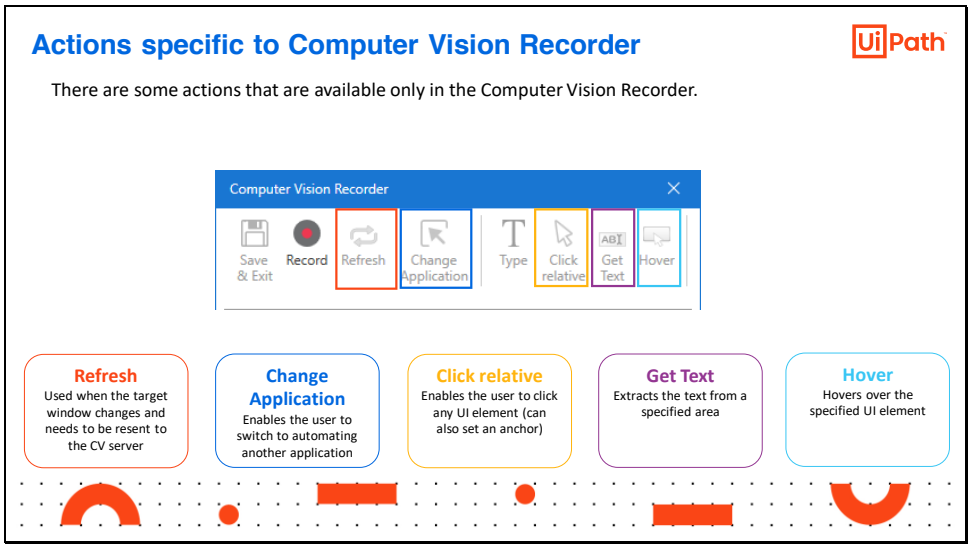
- **Find Image:** It is used to find the image on the screen and returns a Boolean value in the output.

- **Wait Image Vanish:** It waits for the image to disappear from the screen, another activity starts when the image is inactive or invisible from the screen. This activity also has a 30 seconds timeout.



There are some actions that are available only in the Image Recorder. These are:

- **Click Image/Click Text**: Click an image or text
- **Select & Copy**: Select and copy text from a window
- **Screen Scraping**: Scrape UI elements



There are some actions that are available only in the Computer Vision Recorder. These are:

- **Refresh**: Used when the target window changes and needs to be resent to the CV server. This can happen after clicking a button that enters another section of the application.
- **Change Application**: Enables the user to indicate another application window to switch to automating another application.
- **Click relative**: Enables the user to click any UI element. If necessary, the wizard can also set an anchor for the area the user wants to click.
- **Get Text**: Extracts the text from a specified area. This action can also request an anchor if necessary.
- **Hover**: Hovers over the specified UI element.

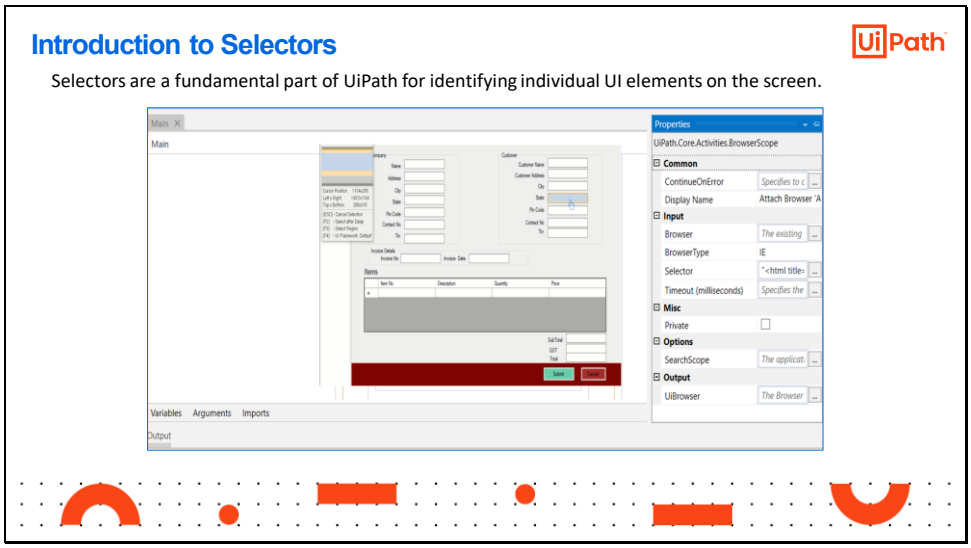
Build a workflow using Web Recorder in UiPath Studio to Sign-in to UiPath's website.

- Ask user for his email address and password.
- Open login page of UiPath's Website
- Sign into the website using the user's credentials

Algorithm

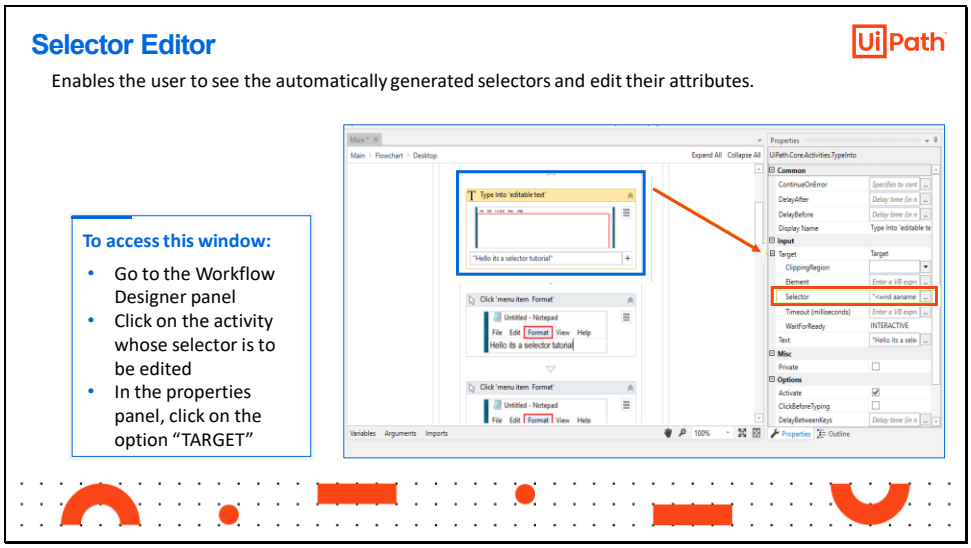
- START
- Use two Input Dialog activities and take email address and password from the user
- Store received input in two variables
- Use Open Browser activity and open URL – "www.uipath.com"
- Use Web Recorder in UiPath Studio to:
    - Click *Try UiPath Free* button
    - Click *Log in* link on the next page
    - Type in the user's email address and password
    - Click *Log in* button
- STOP

This section explains Selectors and Selector Editor.



Selectors are a fundamental part of UiPath as they are used to recognize the UI elements on the user screen. In the UiPath Studio, to start or to execute a specific task in the User Interface, the user requires buttons, windows, drop-down list and advanced features which eventually make up the body of **selectors**. In UiPath studio, the selector activities are available inside the UI automation activity.

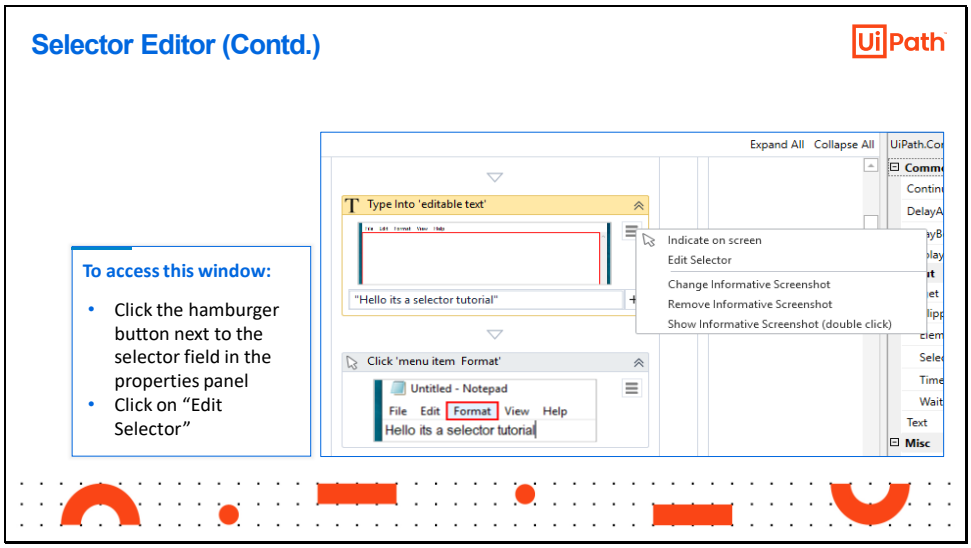
In general, selectors are the address of a UI Element. When a specific task or activity is automated, the user needs to communicate with various UI Elements. The selector is a specific identifier to find a UI Element among multiple applications.



Selector Editor enables the user to access the automatically generated selectors and edit their attributes.

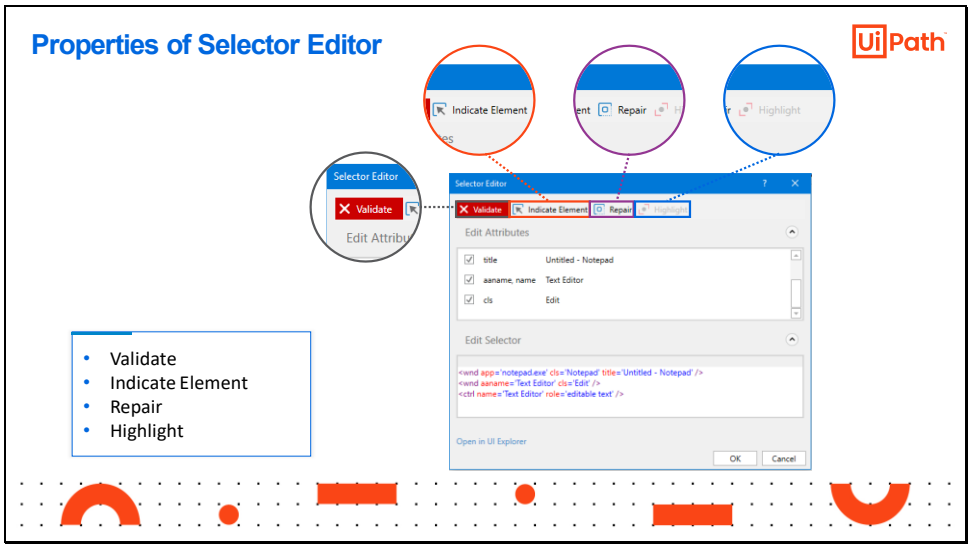
To access selector editor, follow these steps:

- Go to the Workflow Designer panel
- Click on the activity whose selector is to be edited
- Access the properties panel, find the option "TARGET" and expand it



- Further click on the hamburger menu (...) button (At this point, it would display the expression editor on the screen.)

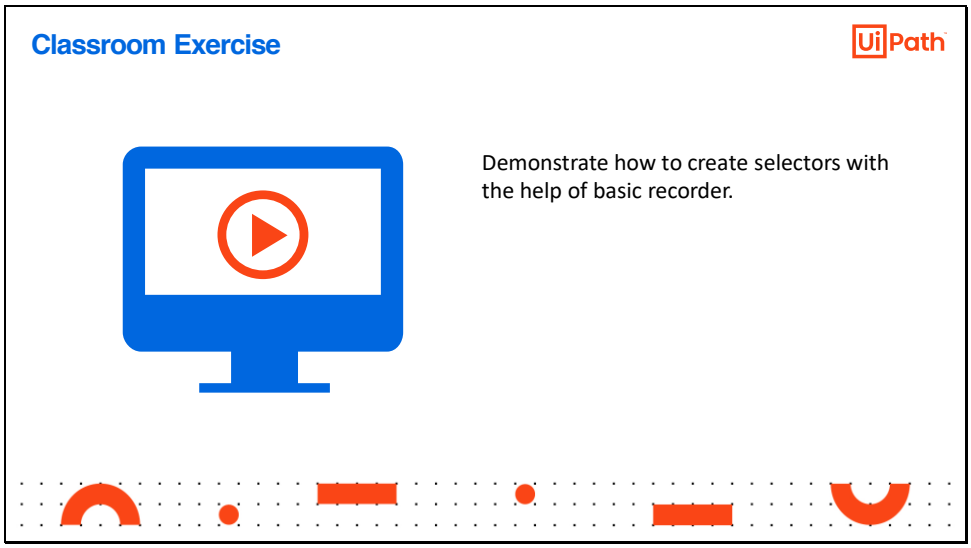
Once the drop-down appears, click on "Edit Selector" to edit the selector.



Properties of Selector Editor:

- **Validate**: The button shows the status of the selector by checking the validity of the selector definition and the visibility of the target element on the screen. The Validate button has following states:
  - O (buffer): Selector is being validated
  - √: Valid selector
  - ×: Invalid selector
  - ?: Modified selector, revalidate

  The button is correlated with UI Explorer validation states.
- **Indicate Element:** Indicates a new UI element to replace the previous one.
- **Repair**: Enables the user to re-indicate the same target UI element and repair the selector. The button is available only when the selector is invalid.
- **Highlight**: Brings the target element in the foreground. The highlight stays on until it's switched off. The button is enabled only if the selector is valid.

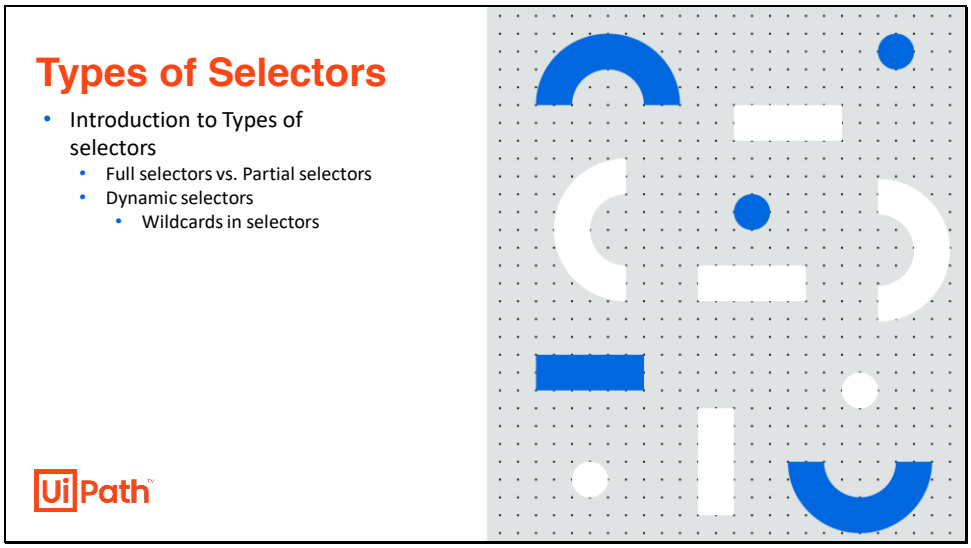


Demonstrate how to create selectors with the help of basic recorder.

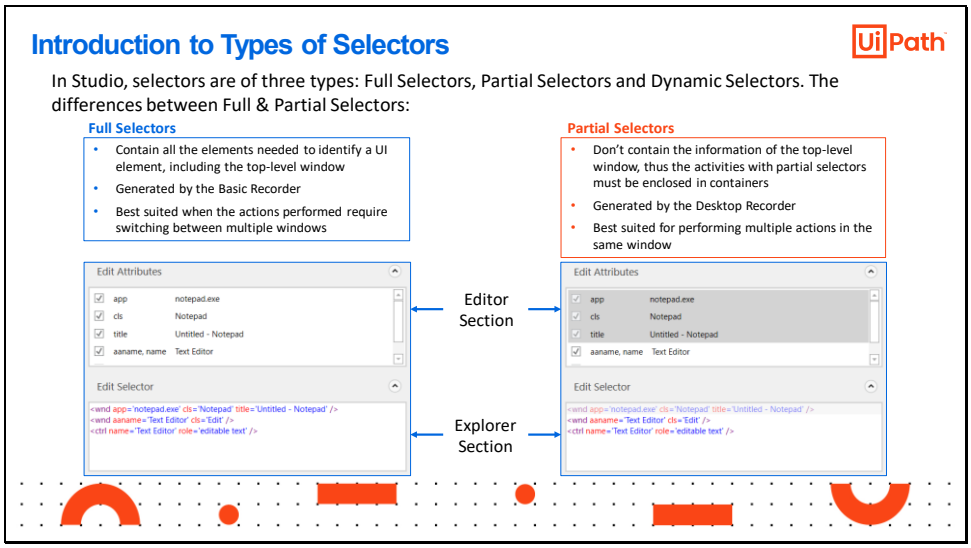
- Open a Notepad window
- In the UiPath Studio, click on Recording from the Design tab, and select Basic from the list.
- From the Basic Recording window, click Type. Select notepad editor.
- Enter a text "Automation increases efficiency" in the small pop-up window, and press enter on your keyboard.
- Click on Type dropdown, and select Send Hotkey. Select notepad editor.
- From the small pop-up window select Ctrl and enter A as the Key.
- Click OK.
- Select Click dropdown, and select Click.
- Select Format from Notepad menu items.
- In the Use Anchor pop-up window, select the check box, and click No.
- Select Click from Basic Recording window.
- Press F2 to pause UiPath for 3 seconds.
- Navigate to Format dropdown of Notepad menu item, and wait for the countdown to go 0.
- Click Font after countdown ends.
- From Type dropdown of Basic Recording window, select Type, and select text area of Font section.
- Enter a font name Century, check Empty field, and press enter on your keyboard.
- Repeat this process for Font style section. Enter font style name as Bold, check Empty field, and press enter on your keyboard.

- Again repeat this process for Size section. Enter size value as 12, check Empty field, and press enter on your keyboard.
- Select Click from Basic Recording window, and select OK button within the Font window.
- Click Save & Exit from Basic Recording window.
- Recording session has generated a Sequence of activities in the Designer panel.
- Run the process.

Outcome: Process executes successfully as recorded.



This section gives an overview of the types of selectors.

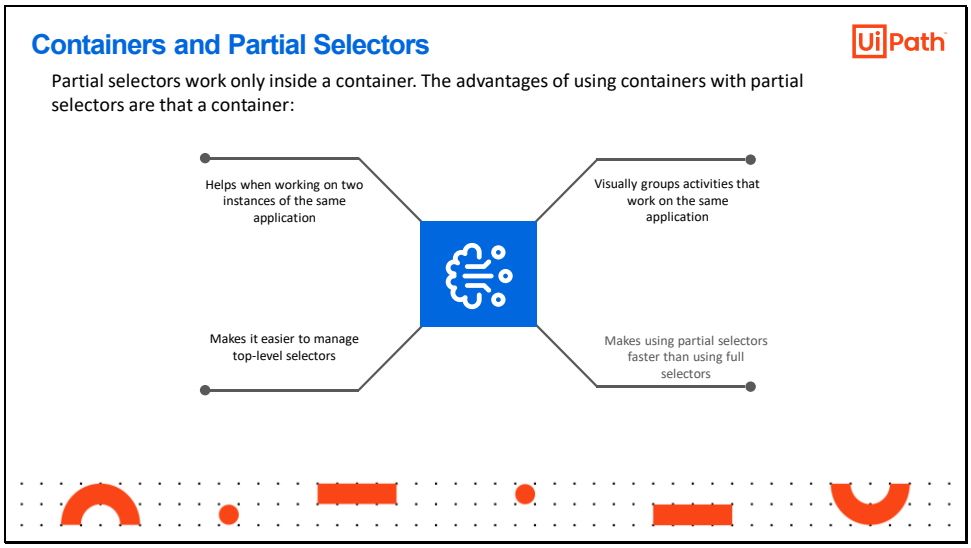


In Studio, selectors are of three types: Full Selectors, Partial Selectors and Dynamic Selectors.

Full and partial selectors are discussed here.

- When automation is created through Basic Recorder, **Full Selectors** are generated. Full selectors start with a window or an HTML identifier and they contain all the information of the UI element. They are suitable when the Robot has to switch through multiple windows and the use of containers would add unnecessary complexity.

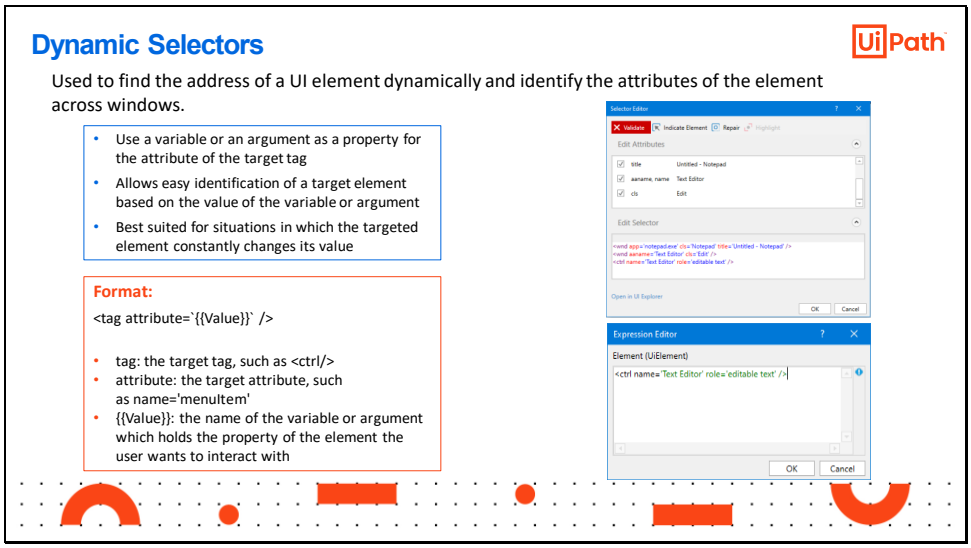
- When automation is created through Desktop Recorder, **Partial selectors** are generated. They do not contain the top-level window information. They are more suitable when the robot has to perform multiple actions in the same window or application. In these cases, the activities will be placed inside a container, such as Attach Browser or Attach Window.



Partial selectors work only inside a container that specifies the top-level window where the elements belong.

The advantages to using containers with partial selectors (instead of full selectors) are:

- Visually groups activities that work on the same application.
- Is slightly faster, not seeking for the top window every time.
- Makes it easier to manage top-level selectors in case manual updates are necessary.
- Helps (and is essential) when working on two instances of the same application.



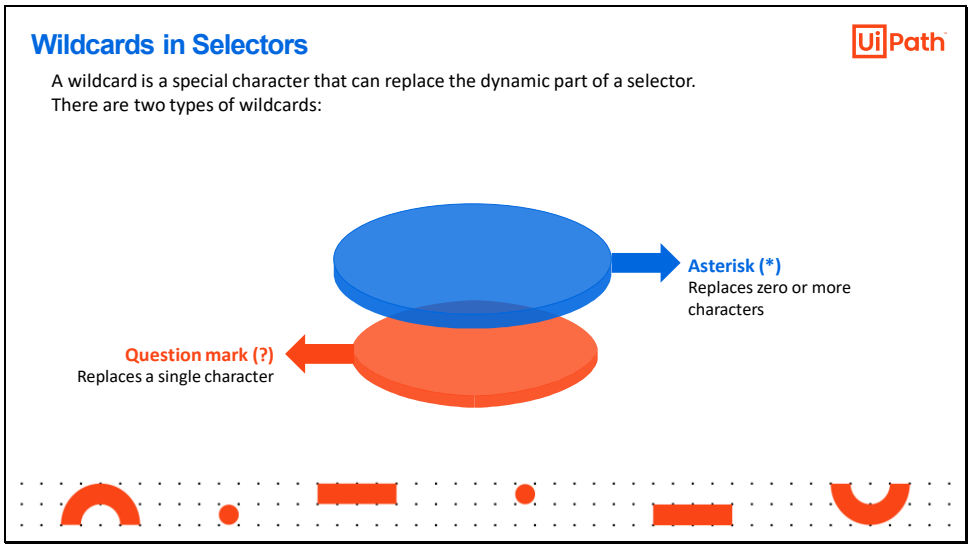
A selector helps to capture the attributes of a particular UI Element on the screen. If the same UI Element is present on another window, then a dynamic selector is used to easily identify its attributes on this window using a variable or an argument.
A dynamic selector uses a variable or an argument as a property for the attribute of the target tag. This allows the selector to easily identify a target element based on the value of the variable or argument. This is best suited for the situations in which the targeted element constantly changes its value as a dynamic selector does not identify the target element based on an exact string, which might change, depending on interactions inside the automation project.
The variable or argument can be changed to interact with a different element, without changing the selector itself.

Format:
<tag attribute=`{{Value}}` />

- tag: the target tag, such as <ctrl/>
- attribute: the target attribute, such as name='menuItem'
- {{Value}}: the name of the variable or argument which holds the property of the element the user wants to interact with

Example: For a calendar on a web page, a user wants to click a specific date. To do this, the user can use the dynamic selector because there is difference in the location of that particular date

due to which selectors attribute may change. So, few changes are required in the attributes to make the selectors of the date to be clicked as dynamic.



Wildcards are symbols that enable the user to replace zero or multiple characters in a string when dealing with dynamically-changing attributes in a selector. When a wildcard is used or a variable is added in between selectors, it is known as customizing selectors.
Types:

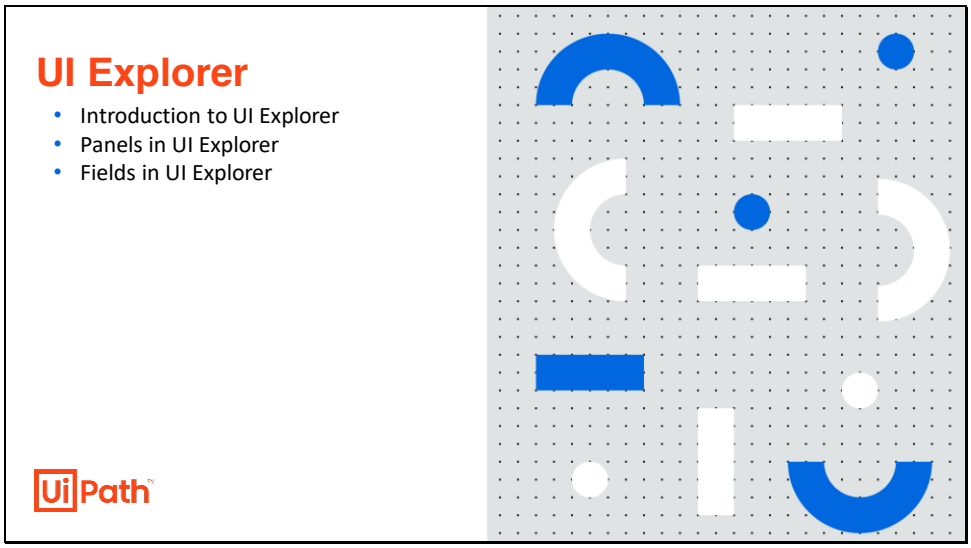
- Asterisk (*) – replaces zero or more characters
- Question mark (?) – replaces a single character

With the help of the below mentioned example we can understand how to use wildcards while building selectors.

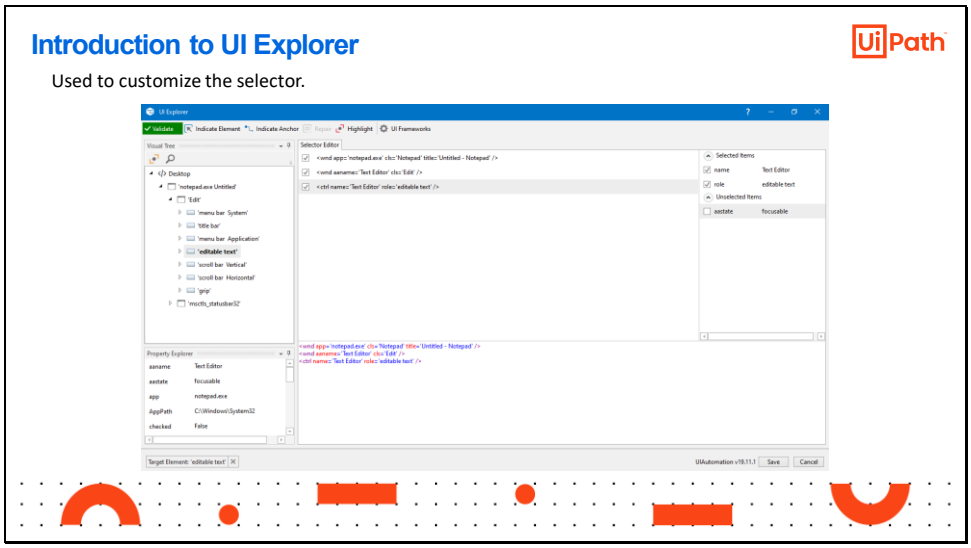
- Open a Notepad file
- Create a new Sequence, give it a proper name and add an annotation
- Use an 'Attach Window' and indicate the Notepad file that is opened. Inside the container:
    - Use a 'Send hotkey' to send Ctrl + h in order to replace 2 spaces with one
    - Use 2 'Type into' activities – one for the first field (with 2 spaces), and another for the second field (with 1 space)
    - Use 2 'Click' activities, one for the 'Replace all' button and the other for 'Close'
- Open the second Notepad file and run the workflow. It will return the 'Cannot find element' error
- Open the selector for the 'Attach window' container. Point out that the title attribute contains the name of the first file
- Use the repair function in the Selector Editor and point out that the part of the date that was different has been replaced by "*"

- Open a third file and re-run the workflow. If it returns another error, replace the entire date by "*"

Please refer: https://docs.uipath.com/studio/docs/selectors-with-wildcards

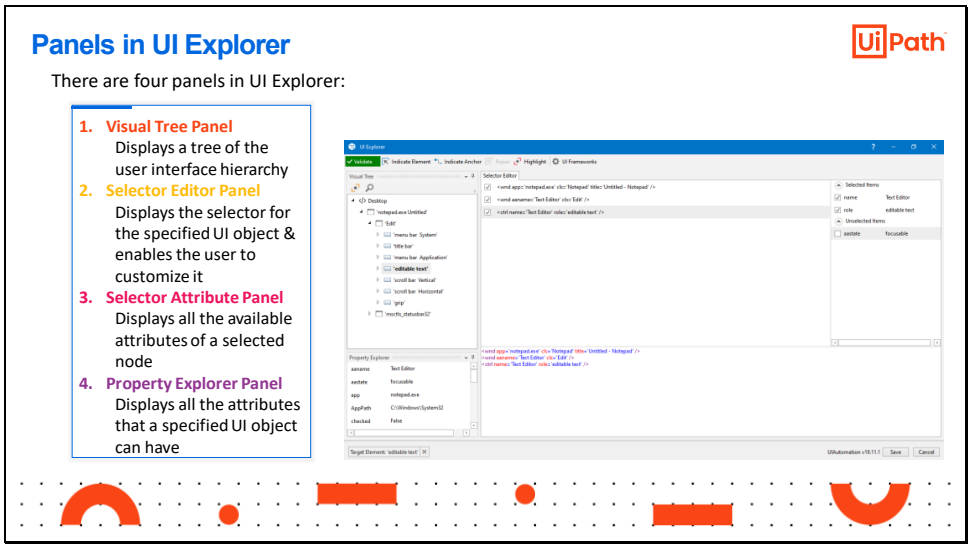


This section gives an overview of UI Explorer.



UI Explorer is an advanced tool that enables the user to create a custom selector for a specific UI element. It can be accessed from the Design tab.

It is available only if the UI Automation Activities package is installed as a dependency to the project.



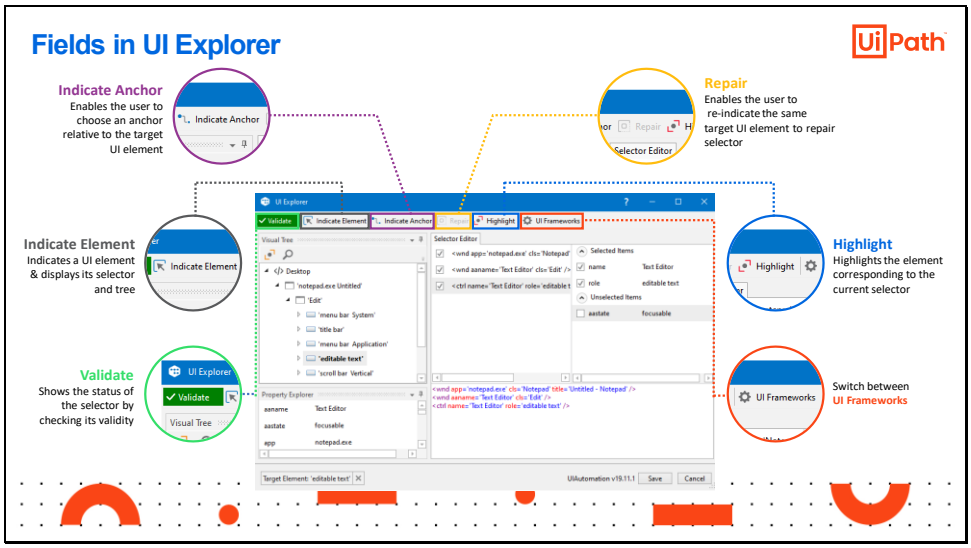
The interface of UI Explorer consists of four panels and several fields.

The four panels in UI Explorer are:

- **Visual Tree Panel**
  - Displays a tree of the user interface hierarchy and enables the user to navigate through it, by clicking the arrows in front of each node. By default, the first time UI Explorer is opened, this panel displays all opened applications, in alphabetical order.
  - The fields here are:
    - **Highlight**: Highlights the selected element from the Visual Tree in real time.
    - **Show Search Options**: Displays the search box and search filter options.
    - **Search Box**: Enables the users to look for a specific string. If an exact match is not found, nodes containing the nearest match are displayed. Wildcards are supported.
    - **Search by**: Filters the search to a selected attribute or a selector.
    - **Children Only**: Limit the search to the first level children of the selected node. (By default, this check box is not selected.)
- **Selector Editor Panel**
  - Displays the selector for the specified UI object and enables the user to customize it.

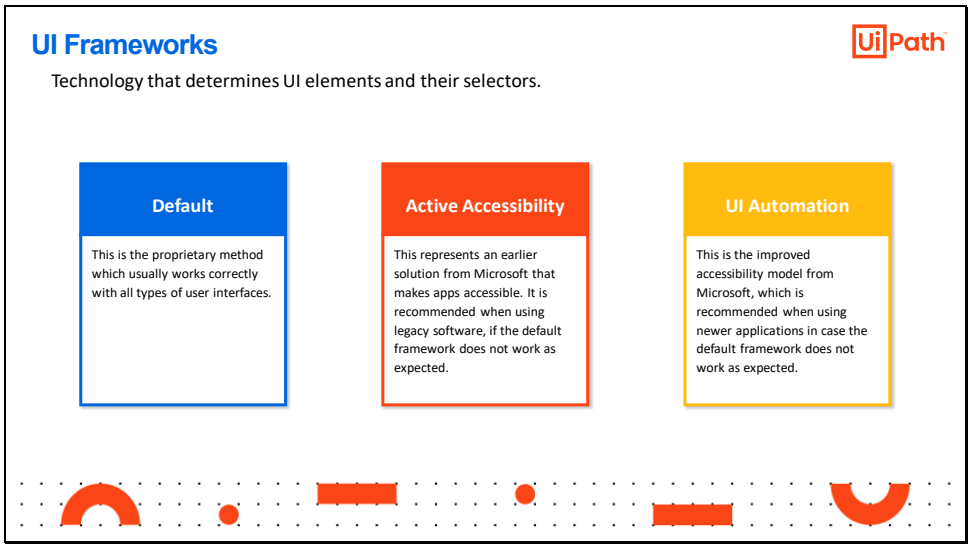
- The bottom part of the panel displays the actual XML fragment that the user has to use in a project.
- The top part of this panel enables the user to view all the nodes in a selector and eliminate the ones that are not necessary by clearing the check box in front of them.
- Selecting a node here displays its attributes in the **Selectors attributes** and **Property Explorer** panels.

- **Selector Attribute Panel**
  - Displays all the available attributes of a selected node
  - The user can add or eliminate some of the node attributes by selecting or clearing the check box in front of each attribute.
  - The user can change the value of each attribute,
- **Property Explorer Panel**
  - Displays all the attributes that a specified UI object can have (including the ones that do not appear in the selector).



The fields in UI Explorer are:

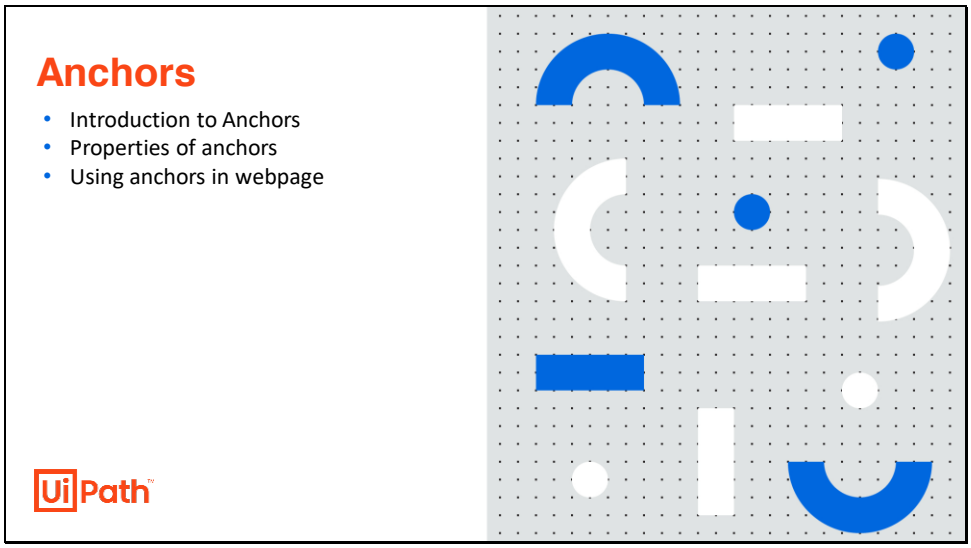
- **Validate**: The button shows the status of the selector by checking the validity of the selector definition and the visibility of the target element on the screen. The Validate button has following states:
    - O (buffer): Selector is being validated
    - √: Valid selector
    - ×: Invalid selector
    - ?: Modified selector, revalidate
- **Indicate Element:** Indicates a new UI element to replace the previous one.
- **Indicate Anchor:** Enables the user to choose an anchor relative to the target UI element.
- **Repair**: Enables the user to re-indicate the same target UI element and repair the selector. The button is available only when the selector is invalid.
- **Highlight**: Brings the target element in the foreground. The highlight stays on until it's switched off. The button is enabled only if the selector is valid.
- **UI Frameworks**: Changes the technology used to determine UI elements and their selectors.



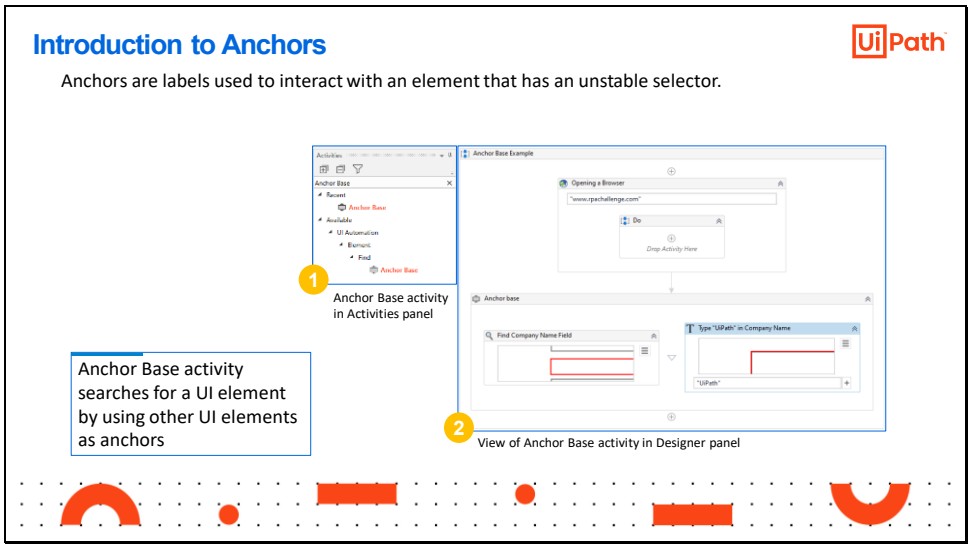
UI frameworks define the technology that determines UI elements and their selectors. The user can switch between three UI Frameworks from the UI Explorer designated button:

- **Default**: This is the proprietary method which usually works correctly with all types of user interfaces.
- **Active Accessibility**: This represents an earlier solution from Microsoft that makes apps accessible. It is recommended when using legacy software, if the default framework does not work as expected.
- **UI Automation**: This is the improved accessibility model from Microsoft, which is recommended when using newer applications in case the default framework does not work as expected.

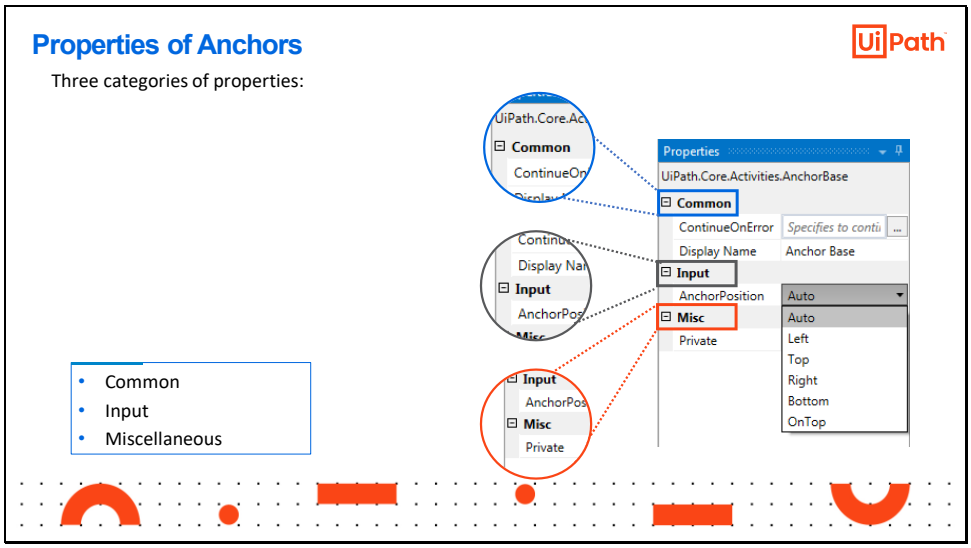
The selectors generated may be very different from one framework to another.



This section explains anchors and its properties.



Anchors are used when a reliable selector is not available in the DataTable. If a selector keeps changing after every iteration, then the user defines an anchor (or label) that can locate selector every time it changes. Anchor base activity in UiPath searches for a UI element by using other UI elements as anchors. So, the activity locates anchor and its relative UI element on a screen.

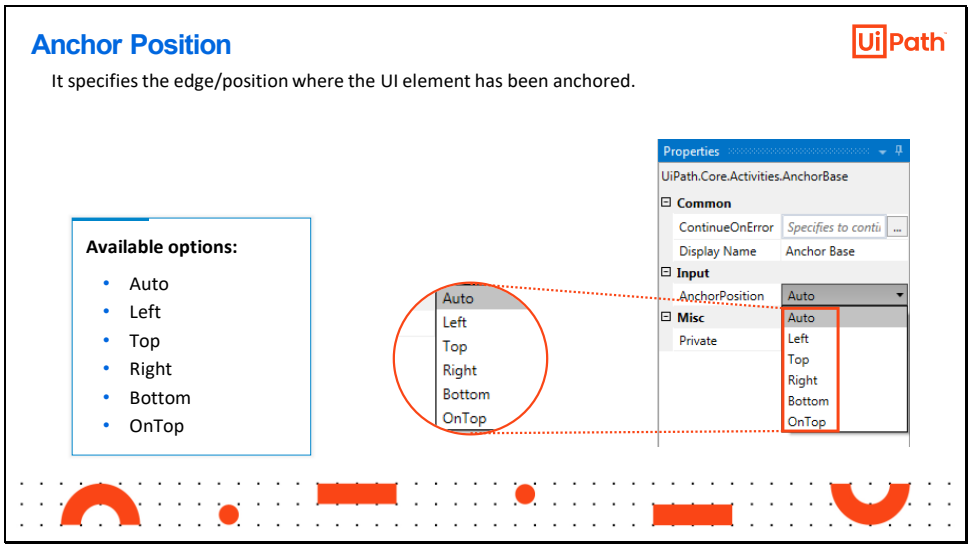


Properties of anchors can be categorized into three parameters.

- **Common**: Under "Common" category, there are two properties:
  - **ContinueOnError:** Used to specify whether automation should continue executing the rest of the activities if the current activity failed. **True** or **False** is used to input the decision for this property. If True or False is not specified for this property, then automation will use False as a default value.
  - **Display Name**. Displays the name of the activity.

- **Input**: Under "Input" category, there is one property:
  - **AnchorPosition:** This is one of the key properties of Anchor. (discussed in detail on the next slide)

- **Miscellaneous**: Under "Miscellaneous" category, there is one property
  - **Private**. If this property is selected, then variables and arguments will not be logged at Verbose level. Verbose level logs the values of variables and arguments. It also makes log entries every time a process starts or ends.



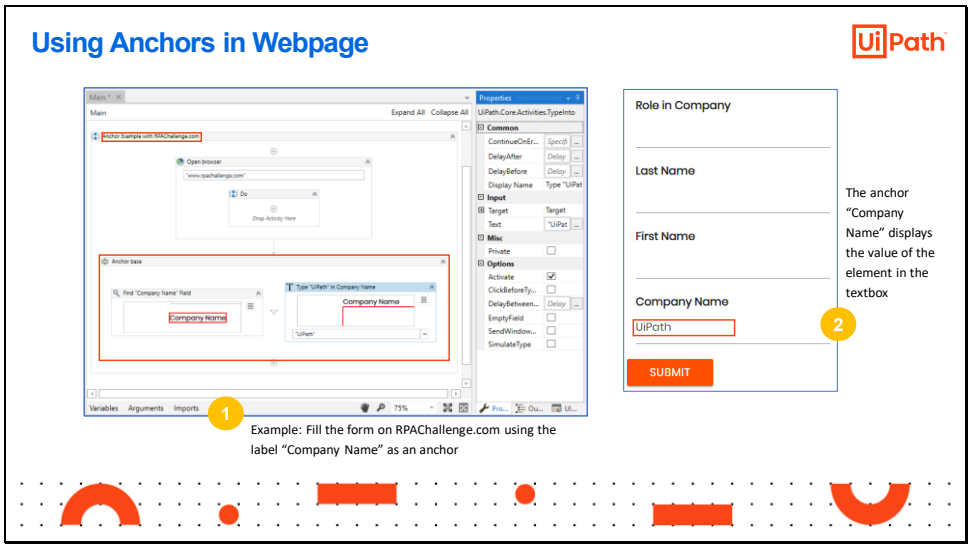
Anchor Position is the property under Input category of Anchor base and is one of the key properties of anchors. It refers to the edge/position where the UI element has been anchored.

There are six options that can be used by Anchor Position property:

- **Auto:** UI element closest to the container is searched. When there are UI elements in all directions and at equal distance, then the Anchor base activity will search UI element in the order – Right, Left, Bottom.
- **Left:** UI element on the left of the container is searched.
- **Top:** UI element at the bottom of the container is searched.
- **Right:** UI element on the right of the container is searched.
- **Bottom:** UI element at the top of the container is searched.
- **OnTop:** UI element with a label is searched.

It is good to use anchor position Auto, and the default direction for the anchor position is first left, right and then bottom. Right, Left and then Bottom directions are generally preferred for an anchor position and then the closest one is selected.

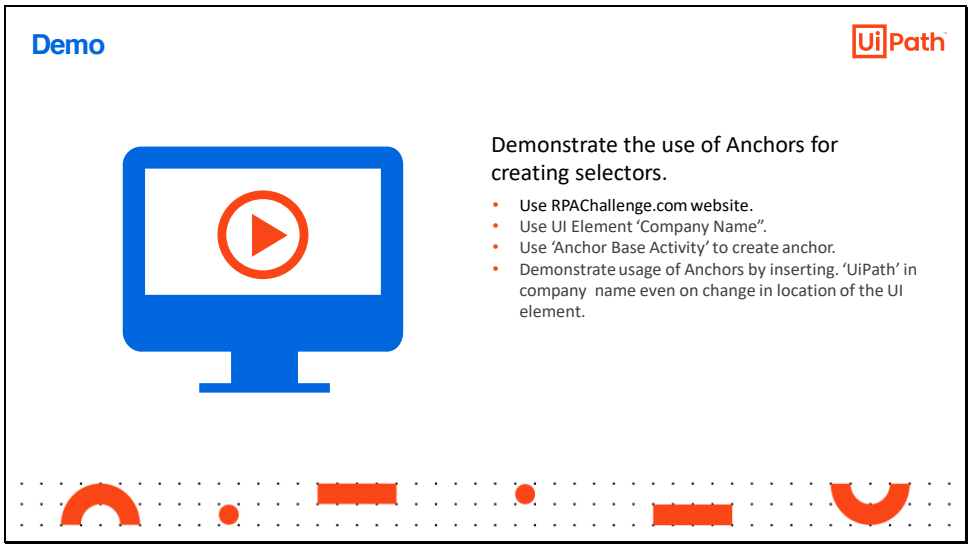
Note: The Top and Bottom options differ from the usual nomenclature. Top searches for UI element in bottom, whereas Bottom searches for UI element in top.



Anchor base is a container that searches for a UI element by using other UI elements as anchors and it is used when a reliable selector is not available. The Anchor base activity is used to locate an anchor on a webpage.

Example: Fill a form on **RPAChallenge.com**.

When a user opens RPAChallenge.com website, a form appears. The workflow writes a particular text string in a field (say Company Name) on this page which changes its position every time the webpage loads. This label **Company Name** can be used as anchor to type into the value in its respective text box.



Demonstrate the use of Anchors in creating selectors.

How to use anchors to locate another UI element on the screen:

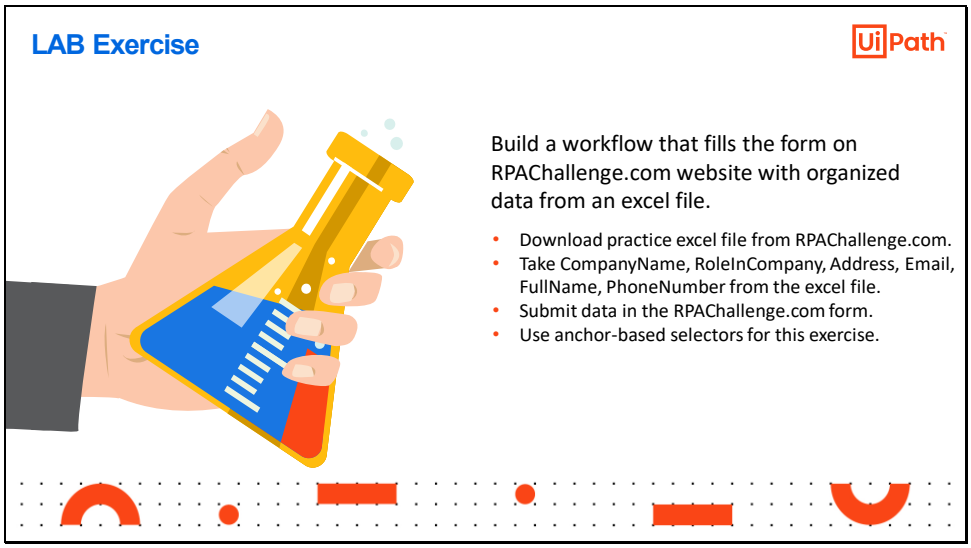
- In UiPath Studio, drag and drop **Open Browser** activity into the designer panel.
- Enter website URL "www.rpachallenge.com".
- Run the file…Notice the **location** of Company Name field in the form. Its location changes every time this webpage is refreshed.
- Insert **Type Into** activity within the **Do** container of **Open Browser** activity.
- Click "Indicate element inside browser" link and indicate input field of **Company Name** label on the website.
- Insert text "UiPath" in the input field of **Type Into** activity.
- Run the file…We can see that the location of the Company Name field has changed this time.
- Also, the execution will take longer than usual because workflow faces difficulty in finding the Company Name. Wait for few seconds to get an execution error.

To solve this problem, use Anchor Base activity and use an anchor.

- Delete the Type Into activity and insert **Anchor Base** activity in the **Do** container.
- Insert **Find Element** activity in the *Anchor* area of Anchor Base.
- Click "Indicate element inside browser" link and indicate the label Company Name as the anchor.

- Insert **Type Into** activity in the *Action Activity* area. Click "Indicate element inside browser" link and indicate the input field of Company Name.
- Run the file

Position of the Company Name field changed again, but the process succeeds in inserting the required text in the field. This is the benefit of using anchors.



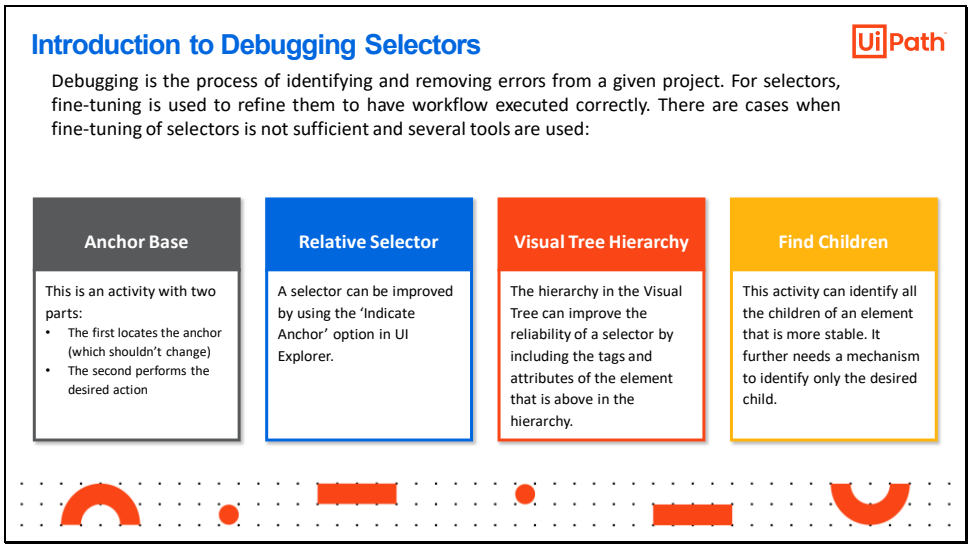
Build a workflow that fills the form on RPAChallenge.com website with organized data from an excel file.

- Download practice excel file from RPAChallenge.com.
- Take CompanyName, RoleInCompany, Address, Email, FullName, PhoneNumber from the excel file.
- Submit data in the RPAChallenge.com form.
- Use anchor-based selectors for this exercise.

Algorithm

- START
- Use a Read Range activity within an Excel Application Scope to read data from Excel file challenge.xlsx and store in a data table
- Use Open Browser activity to open URL – "www.rpachallenge.com"
- Use Click activity to click on the Start button on the website
- Use For Each Row activity to loop through rows in data table
- Use Anchor Base activities within For Each Row activity to locate anchors for each input area of the RPAChallenge.com submission form.
- Use Type Into activities with Anchor Base activities to input data from the data table.
- Use string manipulation methods called Substring and IndexOf to manipulate name of users.
- Use Click activity to submit the form.
- STOP

This section explains how to debug selectors.



Debugging is the process of identifying and removing errors from a given project. Coupled with logging, it becomes a powerful functionality that offers the users information about the project and step-by-step highlighting, to ensure that it is error-free.

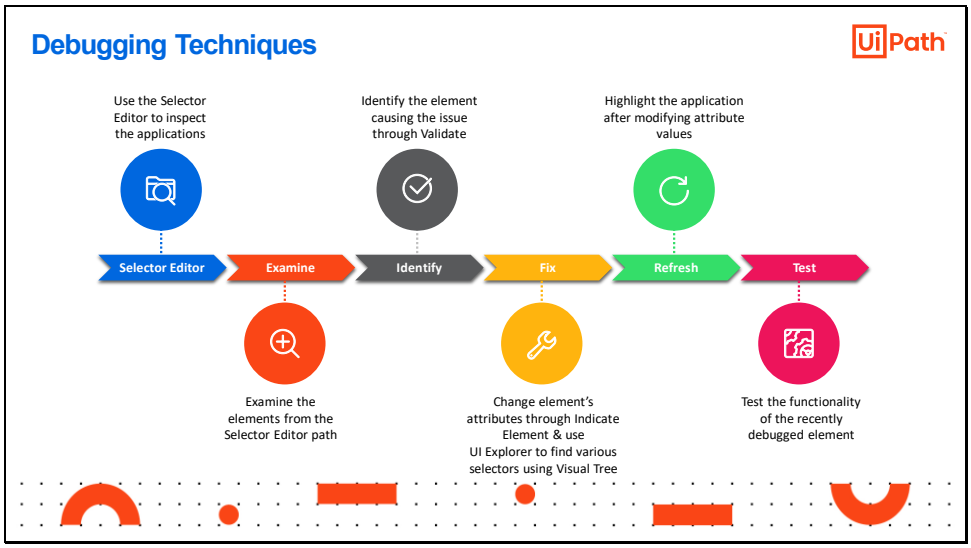
In case of selectors, fine-tuning is used which is the process of refining selectors in order to have the workflow correctly executed in situations in which the generated selector is unreliable, too specific or too sensitive with regards to system changes. It mainly consists of small simple changes that have a larger impact on the overall process, such as adding wildcards, using the repair function or using variables in selectors.

In most of the cases in which the selectors automatically generated are not reliable enough, fine-tuning will solve the issue. However, there are some difficult situations for which there are other approaches:

- **Anchor Base:** Useful in cases in which the attribute values are not reliable (are generated at each execution, for example), but there is a UI element that is stable and is linked to the target UI element. The Anchor Base activity has two parts, one to locate the anchor UI element (like 'Find Element'), and the second to perform the desired activity.
- **Relative Selector**: It incorporates the information about the anchor's selector in the selector of the target UI element. However, the new selector will probably need additional editing, as the part that made the first selector will still be in the new selector. So that part (like a dynamic ID) needs to be removed, and the selector will stabilize using the anchor's selector.
- **Visual Tree Hierarchy**: Useful when the target UI element's selector is not reliable, but the selector of the UI element right above in the hierarchy is. However, again, the selector needs

further editing and validation, as the dynamic part needs to be removed and, at the same time, the user needs to make sure that the target element can be identified with a unique attribute.

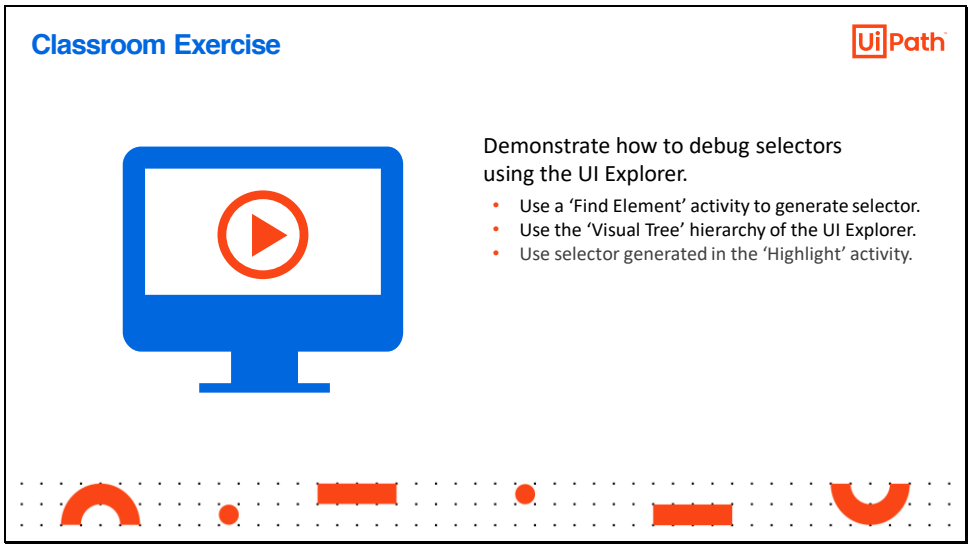
- **Find Children**: It identifies the children of a certain UI element. The output of the activity is the collection of children. So, in order to make it work, the user needs to come up with a mechanism to identify only the target UI element (using one of its attributes, that makes is unique between the children, but wouldn't be enough to identify it universally).



Debugging is the method to identify the error and help in finding the correct selectors until the desired action is achieved.

Selector Editor can be used for checking selectors and UI Explorer for debugging selectors. UI Explorer enables the user to inspect all the attributes that could be used in identifying the element causing the issue. Once the element has been identified, the debugging process starts. Debugging a selector involves changing element's attributes, either adding or removing them and using wildcards where specific attributes have variable values inside them.

- Inspect the applications
- Examine the elements from the Selector Editor path
- Identify the element causing the issue through Validate
- Change the element's attributes through Indicate Element or Repair and use UI Explorer to find various selectors using Visual Tree
- Highlight the application after modifying attribute values
- Test the functionality of the recently debugged element



Demonstrate how to debug selectors using the UI Explorer

How to debug a process using UI Explorer?

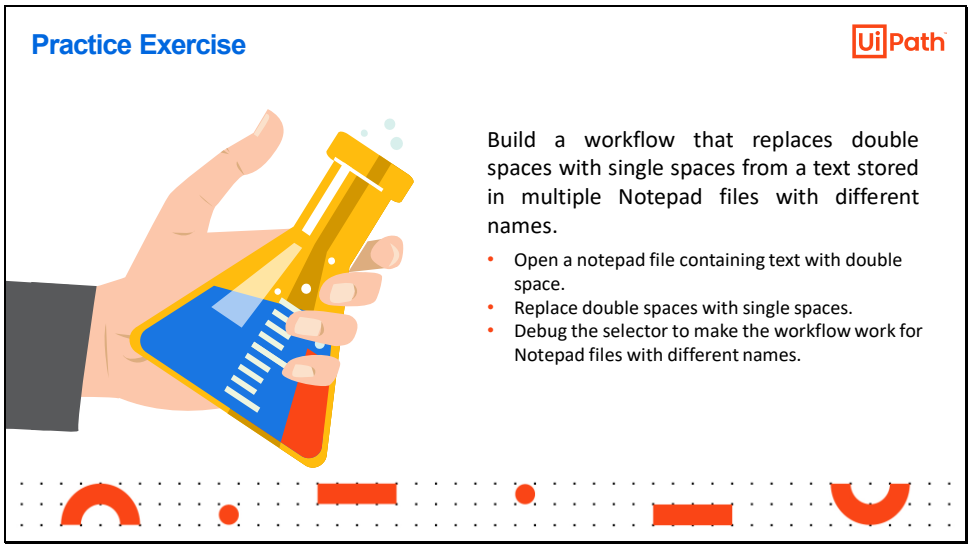
- Open a Notepad window. Notice its default name, which is **Untitled**.
- In UiPath Studio, insert **Type Into** activity in the designer panel.
- Click "Indicate on screen" link and indicate the text field in the Notepad.
- In the text area insert an example text, say "Test".
- Go to the Notepad window and save it as the name **test.txt**.
- Go back to UiPath Studio and run the process file.

Outcome: Process gives an error. It was unable to find the Notepad window due to renaming.

Follow below steps to debug:

- Click the hamburger button within **Type Into** activity and select **Open in UiExplorer** from the context menu.
- In the UI Explorer window, notice the red **Validate** button. It indicates error. Also, notice the title of the Notepad file in the bottom panel. It says 'Untitled - Notepad'.
- Click **Repair** and indicate the text field of the renamed Notepad window. Validate button has turned green. It means error is resolved. Also, the text 'Untitled' is replaced with an asterisk. It indicates wildcard, which means process will now use any open Notepad window if actual is missing.
- Click Save
- Run the file again

Outcome: Process runs without an error.



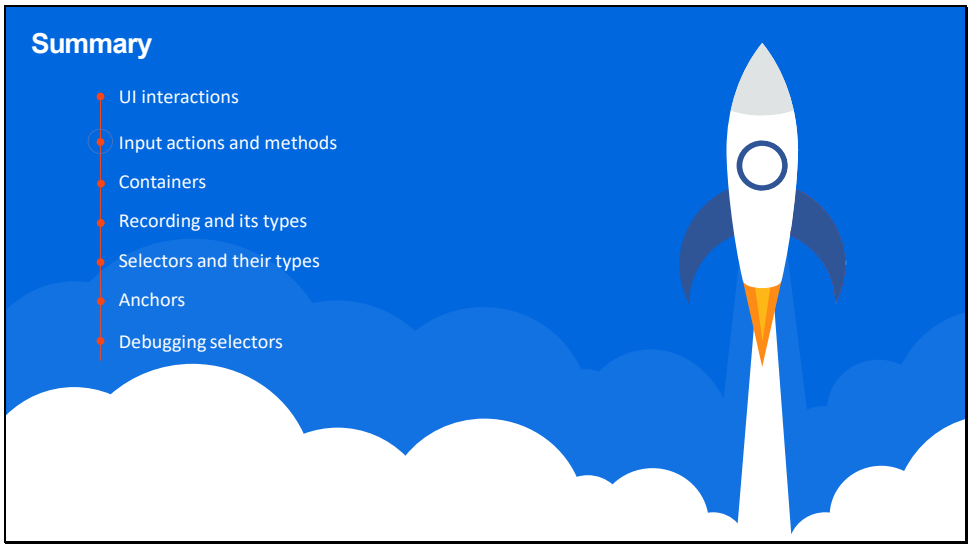
Build a workflow that replaces double spaces with single spaces from a text stored in multiple Notepad files with different names.

Notes:

- Open a notepad file containing text with double space.
- Replace double spaces with single spaces.
- Debug the selector to make the workflow work for Notepad files with different names.

Algorithm

- START
- Use a Attach window activity and indicate one Notepad file (File_123421.txt)
- Use Click activities and indicate Replace option within Edit option from the Notepad file's toolbar.
- Use Type Into activities to enter double space in *Find what* text box and single space in the *Replace with* text box of Replace popup window
- Use Click activities to click Replace All button and to lose the Replace pop up window
- Run the program to replace the double spaces of "File_123421.txt" file.
- Use wildcard (*) in the Notepad file name using Selector Editor to make the workflow work for Notepad with different name
- STOP



To summarize, this lesson explained:

- UI interactions
- Input Actions and Input Methods
- Containers
- Recording and its types
- Selectors and their types
- Anchors
- Debugging selectors