Most Complete WinAppDriver C# Cheat Sheet

Initialize

```
// NuGet: Appium.WebDriver
var appiumOptions = new AppiumOptions();
appiumOptions.AdditionalCapability("app", "Microsoft.WindowsCalculator_8wekyb3d8bbwe!App"); // for Universal Windows Platform apps
appiumOptions.AdditionalCapability("app", "C:\Windows\System32\notepad.exe"); // for classic Windows apps
appiumOptions.AdditionalCapability("deviceName", "WindowsPC");
_driver = new WindowsDriver<WindowsElement>(new Uri("http://127.0.0.1:4723"), options);
_driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(5);
```

Locators

```
_driver.FindElementById("42.333896.3.1");
_driver.FindElementByAccessibilityId("AppNameTitle");
_driver.FindElementByClassName("TextBlock");
_driver.FindElementByWindowsUIAutomation("Views");
_driver.FindElementByName("Calculator");
_driver.FindElementByTagName("Text");
// Find multiple elements
_driver.FindElementsByClassName("TextBlock");
// Search for an element inside another element
_driver.FindElementByName("System")
       .FindElementByName("Minimize");
```

Basic Interactions & Actions

```
// Simple actions
element.Click();
element.SendKeys("textToType");
element.Clear();
// Chained actions
Actions actions = new Actions(_driver);
actions.MoveToElement(element);
actions.MoveByOffset(x, y);
actions.Click();
actions.ClickAndHold();
actions.Release();
actions.ContextClick();
actions.DoubleClick();
actions.DragAndDrop(sourceElement, targetElement);
actions.DragAndDropToOffset(sourceElement, x, y);
actions.KeyDown(Keys.Shift);
actions.KeyUp();
actions.SendKeys("textToType");
actions.Build().Perform();
// Take screenshot
_driver.GetScreenshot(); // returns Screenshot
(base64 encoded PNG)
```

Touch & Pen Interactions

```
// Making a touch stroke
PointerInputDevice device = new PointerInputDevice(PointerKind.Touch);
ActionSequence sequence = new ActionSequence(penDevice, 0);
sequence.AddAction(device.CreatePointerMove(element, 0, 0, TimeSpan.Zero));
sequence.AddAction(device.CreatePointerDown(PointerButton.TouchContact));
sequence.AddAction(device.CreatePointerMove(element, 10, 10, TimeSpan.Zero));
sequence.AddAction(device.CreatePointerUp(PointerButton.TouchContact));
_driver.PerformActions(new List<ActionSequence> { sequence });
// Perform multitouch
PointerInputDevice touch1 = new PointerInputDevice(PointerKind.Touch);
ActionSequence touch1Sequence = new ActionSequence(touch1, 0);
touch1Sequence.AddAction(touch1.CreatePointerMove(element, 50, -50, TimeSpan.Zero));
touch1Sequence.AddAction(touch1.CreatePointerDown(PointerButton.TouchContact));
touch1Sequence.AddAction(touch1.CreatePointerMove(element, 80, -80,
TimeSpan.FromSeconds(1)));
touch1Sequence.AddAction(touch1.CreatePointerUp(PointerButton.TouchContact));
PointerInputDevice touch2 = new PointerInputDevice(PointerKind.Touch);
ActionSequence touch2Sequence = new ActionSequence(touch2, 0);
touch2Sequence.AddAction(touch2.CreatePointerMove(element, -50, 50, TimeSpan.Zero));
touch2Sequence.AddAction(touch2.CreatePointerDown(PointerButton.TouchContact));
touch2Sequence.AddAction(touch2.CreatePointerMove(element, -80, 80,
TimeSpan.FromSeconds(1)));
touch2Sequence.AddAction(touch2.CreatePointerUp(PointerButton.TouchContact));
_driver.PerformActions(new List<ActionSequence> { touch1Sequence, touch2Sequence });
// Pen actions and options
PointerInputDevice device = new PointerInputDevice(PointerKind.Pen);
ActionSequence sequence = new ActionSequence(device, 0);
PenInfo penExtraAttributes = new PenInfo { TiltX = 45, TiltY = 45, Twist = 45 };
sequence.AddAction(device.CreatePointerMove(element, 0, 0, TimeSpan.Zero));
sequence.AddAction(device.CreatePointerDown(PointerButton.PenContact,
 penExtraAttribute$);
sequence.AddAction(device.CreatePointerMove(element, 10, 10, TimeSpan.Zero, new
PenInfo { Pressure = 1f }));
sequence.AddAction(device.CreatePointerUp(PointerButton.PenContact));
_driver.PerformActions(new List<ActionSequence> { sequence });
```