How does JavaScript represent output parameters in the Windows Runtime?

devblogs.microsoft.com/oldnewthing/20221124-00

November 24, 2022

Raymond Chen

The Windows Runtime allows parameters to be declared as `out`, which means that the variable is passed by reference and will be written to, but not read from, by the method. (At the ABI layer, the variable is passed by address.)

```
runtimeclass MyClass
{
    Boolean TryGetCount(out Int32 count);
}
```

Many languages support passing variables by reference, and the projection aligns with those language features.

```
// C#
int count;
if (c.TryGetCount(out count)) ...

// C# 7.0
if (c.TryGetCount(out int count)) ...

// Visual Basic
Dim count as Integer
If c.TryGetCount(count) Then
    ...
End If

// C++/WinRT
int count;
if (c.TryGetCount(count)) ...

// Rust/WinRT
let mut count = 0;
if (c.TryGetCount(&mut count)) ...
```

JavaScript, on the other hand, does not support passing variables by reference. To work around this, any method that has an `out` parameter is rewritten by returning a JavaScript object with a property called `result` which contains the original return value, and with

additional properties, one for each `out` parameter. The original `out` parameters disappear from the formal parameter list.

```
var retVal = c.tryGetCount();
if (retVal.result) {
    console.log(retVal.count);
}
```

The return value of the original `TryGetCount` method is recorded as the `result` property, and the `count` that was returned by the original method becomes a `count` property.

The name of the formal parameter is not just a documentation nicety in JavaScript. The name of the formal parameter is part of the programming interface because it becomes the name of the property![1]

Python also doesn't support passing variables by reference. It performs a transform similar to JavaScript, but instead of returning an Object, it returns a tuple.

```
// Python/WinRT
result, count = c.try_get_count();
```

This awkwardness with output parameters in JavaScript and Python makes output parameters slightly less attractive in the Windows Runtime.

[1] In the Windows Runtime, parameter names are considered part of the interface, and changing parameter names is a breaking change. We saw how JavaScript can be affected by changing the name of a formal parameter. The name is also programmatically significant in C#, since C# lets you pass parameters by name, which is particularly handy for parameters of numeric or Boolean type.

```
var trigger = new TimeTrigger(freshnessTime: 60, oneShot: true);

var inkPoint = new InkPoint(position,
                    pressure: 0.5, tiltX: 0.0, tiltY: 0.0, timestamp: 0);
```

Raymond Chen

Follow