A Drip of JavaScript

JavaScript's void Operator

Originally published in the A Drip of JavaScript newsletter.

If you've been doing web development for any length of time, you've probably seen links that look something like this:

```
<a href="javascript:void 0;" onclick="doSomething()">Do something</a>
```

While this is a rather terrible line of HTML, there is something interesting going on in it: namely, the `void 0` that is executed when a user clicks the link. What exactly does `void 0` do? Lets ask our trusty console.

```
// Outputs: undefined
console.log(void 0);
```

That's interesting. If `void 0` is equivalent to `undefined`, what about using `void` with other values?

```
// Outputs: undefined
console.log(void "test");

// Outputs: undefined
console.log(void {});
```

It turns out that `void` 's one and only purpose is to make an expression evaluate to `undefined`. While this isn't useful in most situations, it can come in handy occasionally.

As we've talked about before, `undefined` isn't a JavaScript keyword, but a property of the global object. That means that `undefined` can be "shadowed" within a function's scope. Here is an example:

```
function shadowLog () {
    var undefined = "shadowed";
    console.log(undefined);
}

// Outputs: "shadowed"
shadowLog();
```

Under ordinary circumstances one can expect the `undefined` variable to refer to the `undefined` primitive. However, if you are dealing with variable shadowing or redefining undefined, then it can be helpful to use `void` as a guarantee that you are working with the `undefined` primitive, like so:

```
(function shadowFunc () {
    var undefined = "shadowed";
    var undefinedVar;

    // Outputs: false
    console.log(undefined === void 0);

    // Outputs: false
    console.log(undefinedVar === undefined);

    // Outputs: true
    console.log(undefinedVar === void 0);
})();
```

When using `void`, it is conventional to use `0` as the operand. Using `void 0` here allows us to easily check whether a variable is `undefined`, even though the name `undefined` was shadowed in the function's scope.

That's it for this week. Thanks for reading!

Josh Clanton
