A Drip of JavaScript

# Using Dispatch Tables to Avoid Conditionals in JavaScript

*Originally published in the A Drip of JavaScript newsletter.*

When writing code, one of the surest ways to keep things simple and straightforward is to avoid conditionals when possible. Unfortunately, it is fairly common to see code with a lot of `if`, `switch`, and `case` statements like the following:

```
function processUserInput(command) {
    switch (command) {
        case "north":
            movePlayer("north");
            break;
        case "east":
            movePlayer("east");
            break;
        case "south":
            movePlayer("south");
            break;
        case "west":
            movePlayer("west");
            break;
        case "look":
            describeLocation();
            break;
        case "backpack":
            showBackpack();
            break;
```

```
    }
}
```

Above we have a function for processing user input from a text adventure game. While it isn't terribly difficult to understand, it is more complicated than necessary. And as the number of commands grow, the function can quickly become unwieldy. So what can we do to simplify it?

```
var commandTable = {
    north:    function() { movePlayer("north"); },
    east:     function() { movePlayer("east");  },
    south:    function() { movePlayer("south"); },
    west:     function() { movePlayer("west");  },
    look:     describeLocation,
    backpack: showBackpack
}

function processUserInput(command) {
    commandTable[command]();
}
```

In this refactored version we are using a dispatch table to hold all the possible commands a user can give, and the functions the program should call. That changes the `processUserInput` function into a single line, and eliminates all conditionals.

If you are unfamiliar with bracket notation, it is just an alternate way of accessing a function's properties, with the advantage that you can use variables for the property's name. For example, if `command` is "north", then `commandTable[command]` is equivalent to `commandTable.north`.

The fundamental change we made is transforming the conditionals into a data structure. And data is much easier to manipulate than conditionals. In the future, if we want to add a new command, all we have to do is add it to `commandTable`. No messing around with `case` or `break` statements required.

Thanks for reading!

Josh Clanton

---