Assignment Operators

Another set of operators that you need to know about are the assignment operators, and these change the value of a variable.

Assignment for Primitive Types

Assignment operators let you set the value of a number, a string, or a Boolean variable. The assignment operator also lets us change the value of an existing variable.

```
var smallNumber = 2;

console.log(1 + 1);
// 2

console.log(1 + smallNumber + 3);
// 6
```

So assignment is very straightforward for primitive types.

Assignment for Reference Types

Assignment is actually pretty straightforward for reference types as well. You assign a variable to an array or an object or set values for the subelements of arrays and objects.

```
var myArray = [1, "Hello"];
var newArray = myArray;
console.log(myArray);
// [1, "Hello"]
console.log(newArray);
// [1, "Hello"]
newArray[0] = 2;
console.log(newArray);
// [2, "Hello"]
console.log(myArray);
// [2, "Hello"]

var myObject = {"size":5, "color":"red"};
var newObject = myObject;
console.log(newObject.color);
// "red"
console.log(myObject.color);
// "red"
newObject.color = "blue";
console.log(newObject.color);
// "blue"
console.log(myObject.color);
// "blue"
```

Add and Subtract Assignment

There are add and subtract assignment operators in JavaScript. Add and subtract assignment operators can change a primitive variable's value by adding or subtracting a value. We've seen something similar to this before with the increment anddecrement operators, but this lets you add any value, not just one. Remember, just because you can add strings to numbers doesn't mean you can subtract strings from numbers.

```
var firstNumber = 4;
var secondNumber = 1000;
var word = "Hello";

firstNumber += secondNumber;
console.log(firstNumber);
// 1004

firstNumber -= secondNumber;
console.log(firstNumber);
// 4

word += firstNumber;
console.log(word);
// Hello4

word -= firstNumber;
console.log(word);
// NaN
```

So `+=` is it's going to add the value of `secondNumber` to `firstNumber` changing `firstNumber` at the same time. The same applies to `-=`.

Multiply, Divide, Modulus Assignment

Now let's see an example on multiply, divide and modulus assignment.

```
var firstNumber = 3;
var secondNumber = 1000;
var word = "Hello";

firstNumber *= secondNumber;
console.log(firstNumber);
// 3000

firstNumber /= secondNumber;
console.log(firstNumber);
// 3

secondNumber %= firstNumber;
console.log(secondNumber);
// 1

word *= firstNumber;
console.log(word);
// NaN
```

So for primitive variables and particularly for numbers, the assignment operators are very convenient.

Not for Reference Variables

But it's not really a good idea to use them forour reference variables that we've seen. There are some surprising results that may come out.

```
var aNumber = 3;
var arr = [1, 2];
var obj = {"age":25};

console.log(arr);
```

```
// [1,2]
console.log(typeof(arr));
// "object"
console.log(obj.age);
// 25
console.log(typeof(obj));
// "object"

arr += aNumber;
console.log(arr);
// "1,23"
console.log(typeof(arr));
// String

obj += aNumber;
console.log(obj.age);
// undefined
console.log(obj);
// "[object Object]3"
console.log(typeof(obj));
// String
```

Note what happens when you do `arr += aNumber;`. `arr` is being converted to a string because we used `+=`.

For `obj += aNumber;` the `obj.age` is no longer even defined. It doesn't exist because `obj` has been converted to a string.

The results may not be what you expect. Limit your use of this modification assignment operators to primitive variables.