Comparison Operators

There are a number of operators that can be used to compare variables and generate a boolean value based on the relationship between variables.

Equality for Primitive Variables

We've seen a couple of examples where we've tested the equality of primitive variables to see if they were the same. The question when dealing with primitive variables is whether the value of the variables is the same.

Let me show you some examples of basic values comparisons for primitive variables.

```
var firstNumber = 5;
var secondNumber = 10;
var firstWord = "Hello";
var secondWord = "World";

console.log(firstNumber == 5);
// true
console.log(firstNumber == firstNumber);
// true
console.log(firstNumber == secondNumber);
// false

console.log(firstWord == "Hello");
// true
console.log(firstWord == firstWord);
// true
console.log(firstWord == secondWord);
// false
```

Remember, we're using the comparison operator with two equal signs to say that the two values on either side of this operator have the same value.

Equality for Reference Variables

Reference variables are not considered equal if they don't both reference the same object in memory. As we've discussed primitive variables such as strings and numbers are compared by value while reference variables such as objects and arrays are compared by their location in memory.

```
var arr = [1, 2];
var arr2 = [1, 2];
var obj = {"color":"red"};
var obj2 = {"color":"red"};

console.log(arr == arr);
// true
console.log(arr == [1, 2]);
// false
console.log(arr == arr2);
// false
```

```
arr2 = arr;
console.log(arr == arr2);
// true

console.log(obj == obj);
// true
console.log(obj == {"color":"red"});
// false
console.log(obj == obj2);
// false
obj2 = obj;
console.log(obj == obj2);
// true
```

Comparing Variables of Different Types

You can also compare variable of different types. In most cases, variables of different types will be converted to a common type first, before the comparison is done. So for example, strings might be converted to numbers, booleans will be converted to numbers and objects will use the `toString` method of the object to convert them to strings before doing a comparison.

```
var aNumber = 5;
var aString = "5";
var truth = true;
var falsehood = false;
var anObject = {"color":"red"};

console.log(aNumber == aString);
// true

console.log(truth == 1);
// true

console.log(faslehood == 0);
// true

console.log("[object Object]" == anObject);
// true
```

Because `aNumber` contains the number 5, and `aString` contains the string value 5, the two were coerced to be the same variable type. And because the string could be converted to a number, it was possible to say that the values actually matched.

When you convert the object to a string, you don't get the contents of the object, you get object.

Once you know what to expect, you can figure out what the comparison is going toyield when comparing variables of different types.

Strict Equality

You can also force JavaScript not to convert types. We've introduced this strict equality operator before. It uses three equal signs to determine whether both the value, and the type of two variables are the same.

```
var firstNumber = 5;
var secondNumber = 5;
var firstWord = "Hello";
var secondWord = "Hello";

console.log(firstNumber == secondNumber);
// true
console.log(firstNumber === secondNumber);
// true
```

As a JavaScript programmer, you might find it's much more convenient to use the three equal signs, rather than the two equal signs in most cases because that way you'll know that the type, and the value are both being compared. You can trust the result that you're getting shows that the two

variables orvalues really are the same.

Inequality

You can also test inequality with JavaScript. And again, when you're doing inequality comparisons, the values will be converted before the comparison is done. JavaScript supports both loose and strict inequality, and I'll show you the difference.

```
var firstNumber = 5;
var secondNumber = 10;
var numberString = "5";
var firstWord = "Hello";
var secondWord = "World";

console.log(firstNumber != secondNumber);
// true

console.log(firstNumber != numberString);
// false

console.log(firstNumber != 5);
// false

console.log(firstNumber !== numberString);
// true

console.log(firstWord != secondWord);
// true

console.log(firstWord != "Hello");
// false
```

We're using the `!=` operator, which says these two are not equal. This operator converts `firstNumber` and `numberString` to the same type, compares them, sees that they actually are the same. So saying that they're not the same is `false`.

However using `!==` preserves the type when doing the comparison.

Once again, using the exact comparison operator with the two equal signs is probably going to be more effective most cases, so that you know forsure what you're comparing.

Relation for Number Variables

Frequently, you're going to want to compare number variables to see which number is larger or smaller than another. JavaScript supports greater than, greater than or equals to, less than, and less than equals to operators, and I'll show you quickly how those work.

```
var firstWord = "Hello";
var secondWord = "World";
var thirdWord = "hello";
var fourthWord = "help";

console.log(firstWord > secondWord);
// false

console.log(firstWord < secondWord);
// true

console.log(firstWord >= "Hello");
// true

console.log(firstWord > thirdWord);
// false

console.log(firstWord < thirdWord);
// true
```

```
console.log(fourthWord > thirdWord);
// true
```

Strings are compared, letter by letter, in order by the position of each letter in the alphabet. In the case of capital and lowercase letters, capital letters actually have a lower value than lower case letters, and that's because of the ASCII numbering system that's used to calculate the position in the alphabet of each character.

The `firstWord` is not greater than `secondWord`, according to JavaScript. If we go letter by letter,we can see that the letter H is lower in the alphabet than the letter W. And since H and W are the first letters of these strings, the comparison between them determines the `firstWord` is not greater than `secondWord`.

The formula for calculating relations with string variables is to go letter by letter through the string until you get to letters that are different, and as soon as you do, that's where you do the comparison. And if the letter is capital, it's considered lower than the exact same letter as a lower case.