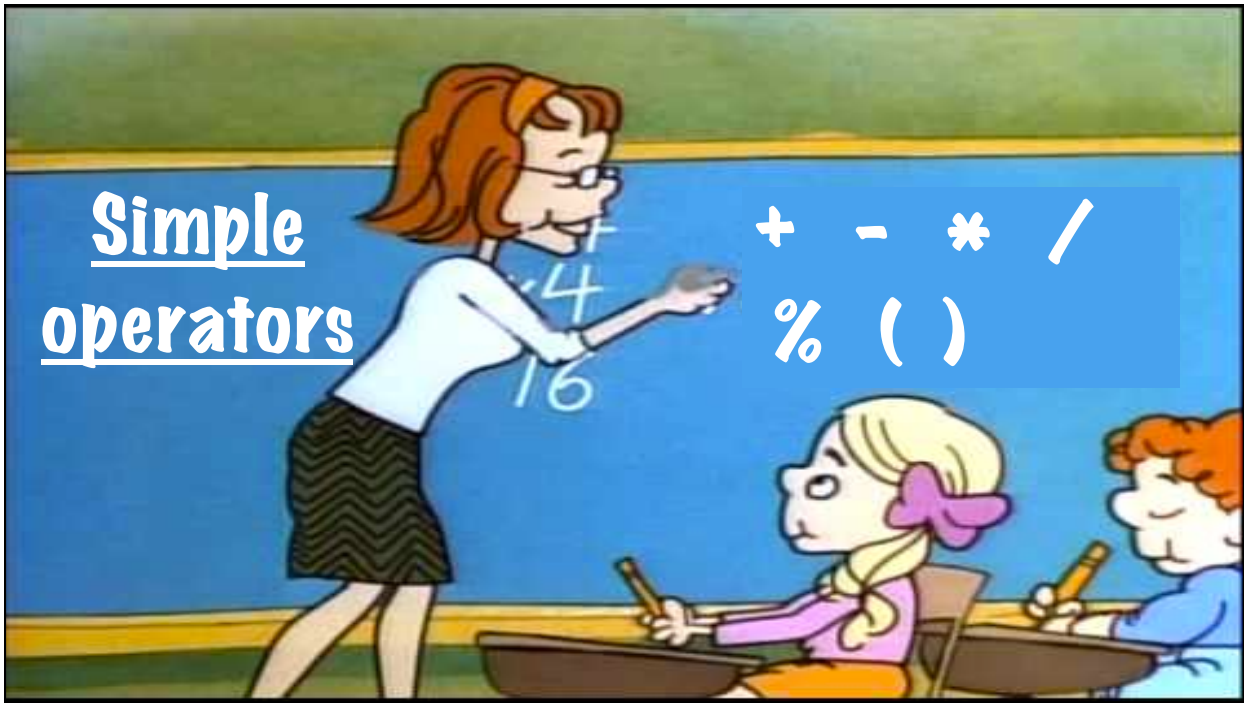
Operators

tl;dr

- Mathematical operators like +, -, *, /, ()
- Auto operators like +=, -=, *=, ++, --
- Comparison operators like ==, ===, !=, >, <, >=
- Logical operators like &&, !, and ||
- Ternary operator
- String templates
- Destructuring



Combining numbers and strings

```
var n = "3";
var x = "The magic number is " + n;
// implicitly coerces numbers to strings
x = 5 + n;   // 53
x = 5 + Number(n);  // 8
x = 5 + +n;   // 8
```

Auto-assignment operators let you write quicker

```
x += 5; same as x = x + 5;
x -= 5; same as x = x - 5;
x *= 5; same as x = x * 5;
x /= 5; same as x = x / 5;
x++; same as x = x + 1;
x--; same as x = x - 1;
```

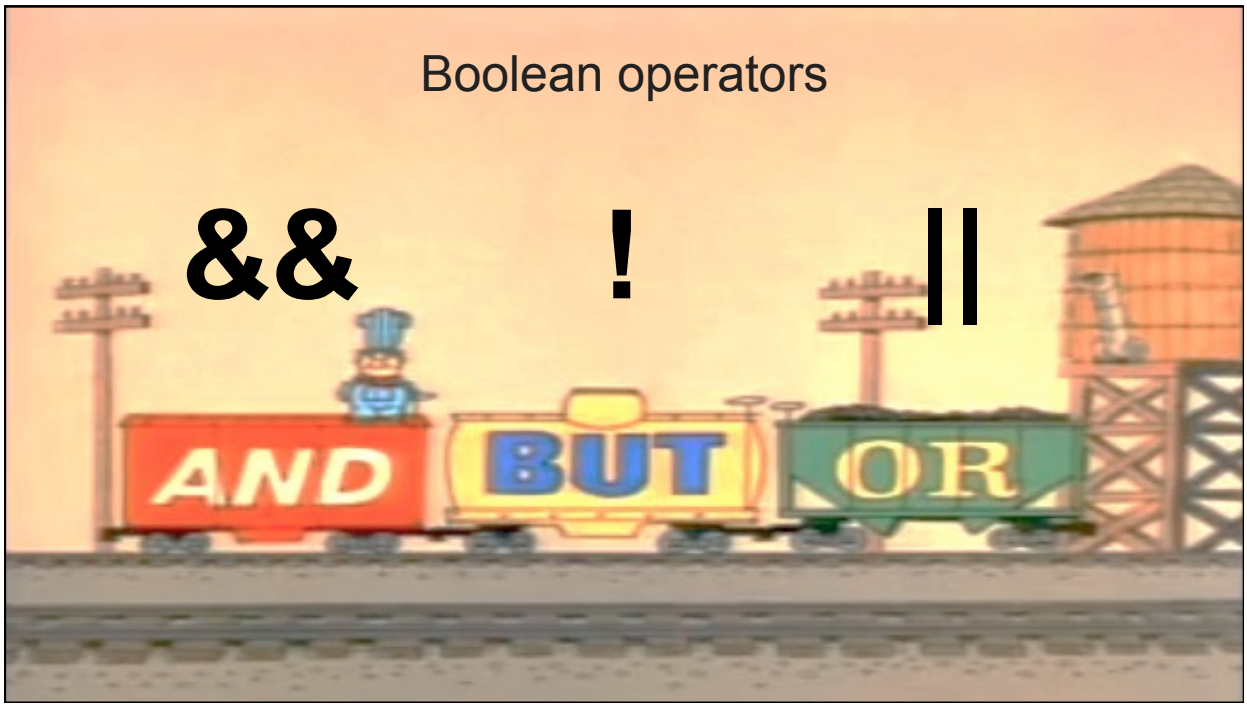
Comparison Operators

==
!=
===
!==
>
<
>=
<=

Identical objects are never equal

```
const o1 = { foo: "bar" };
const o2 = { foo: "bar" };
o1 == o2  // false
o1 === o2  // false
const o3 = o1;
o1 == o3  // true
o1 === o3 // true
```

Boolean operators

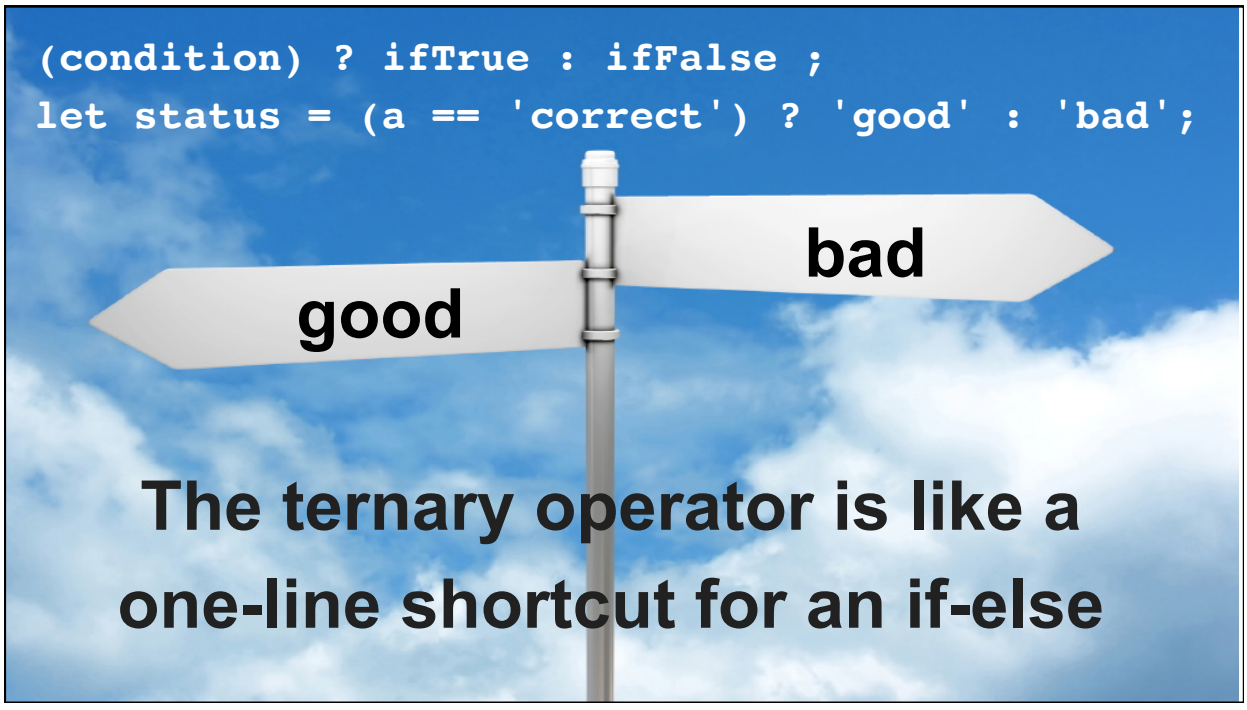


Boolean operators do short-circuit

```
if ( iNeedIt(prod) && priceIsReasonable(prod) )
  buy(prod);
if ( iHaveCoupon(prod) || isOnSale(prod) )
  buy(prod);
```

- So things like this work:

```
const employer = company || "no company";
```



String templates

Borrowing from mustaches/handlebars

Traditional way

```
var name = 'Your name is ' + first + ' ' + last
+ '.';
var url = 'http://us.com/api/messages/' + id;
```

New way

```
var name = `Your name is ${first} ${last}.`
var url = `http://us.com/api/messages/${id}`
```

Multiline strings

```
var roadPoem =
`Then took the other, as just as fair,
 And having perhaps the better claim
 Because it was grassy and wanted wear,
 Though as for that the passing there
 Had worn them really about the same,`;
```

Destructuring

Syntax - Object form

- Just put the object on the left side of the =

```
let {prop1:v1, prop2:v2} = someObject;
```

- Where prop1 & prop2 are properties of someObject
- And v1 & v2 will become the new variables.

- Note: You're not creating an object. You're matching based on the shape of the object.

An example of object destructuring

```
let {username: user, password: pass} = req.body;
// same as
// let user = req.body.username;
// let pass = req.body.password;
```

- or more directly ...

```
let {username, password} = req.body;
// same as
// let username = req.body.username;
// let password = req.body.password;
```

Syntax - Array form

- Just put the array on the left side of the =

```
let [x, y] = someArray;
```

- x and y will be populated with the first and second elements in someArray
- Note: You're not creating an array. You're matching positionally

An example of array destructuring

```
let [ln1, ln2, ln3, , ln5] = allLines.split('\n');
console.log(ln1);  // I'm line one
console.log(ln5);  // I'm the 5th line
```

- Note that we're ignoring lines four, six, and up.

tl;dr

- Mathematical operators like +, -, *, /, ()
- Auto operators like +=, -=, *=, ++, --
- Comparison operators like ==, ===, !=, >, <, >=
- Logical operators like &&, !, and ||
- Ternary operator
- String templates
- Destructuring