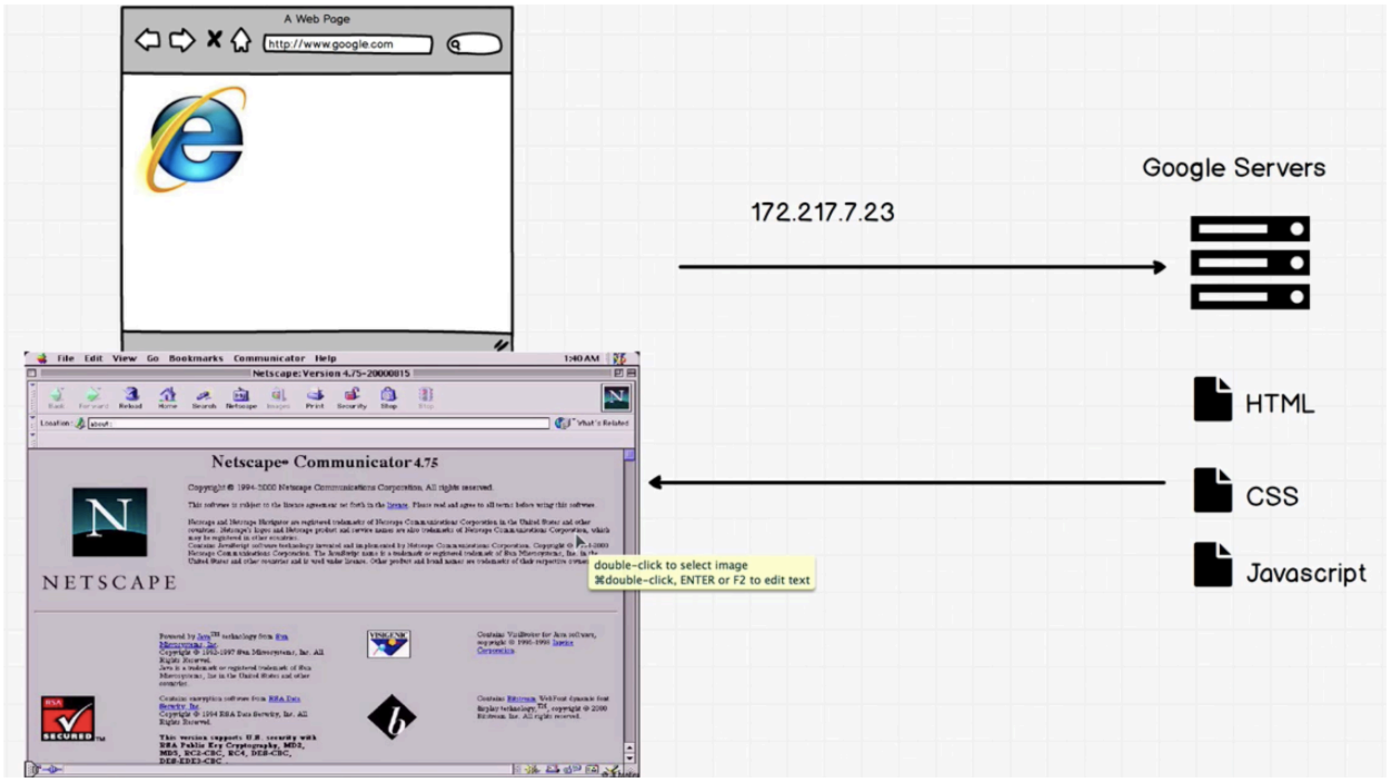
JavaScript: Types & Comparisons

The Complete Web Developer in 2019

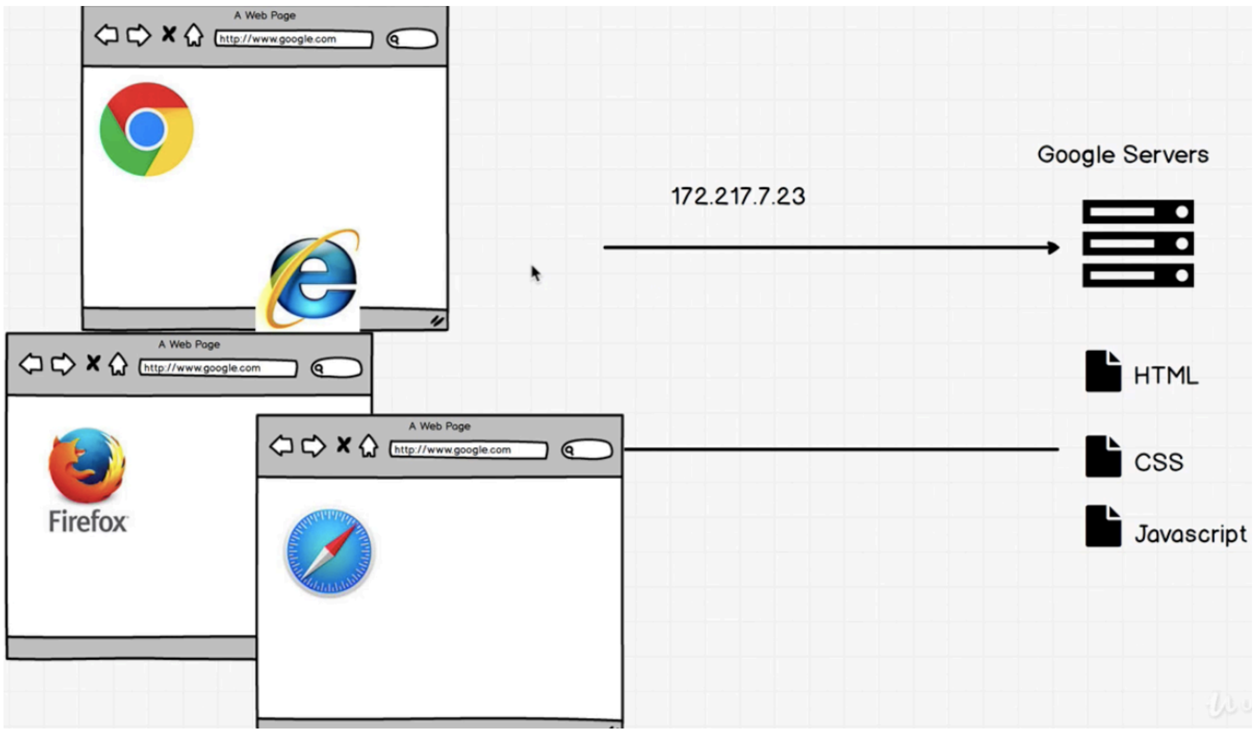
The Complete Web Developer in 2019
Zero to Mastery
Andrei Neagoie
Lecture Notes by Stephanie

Javascript

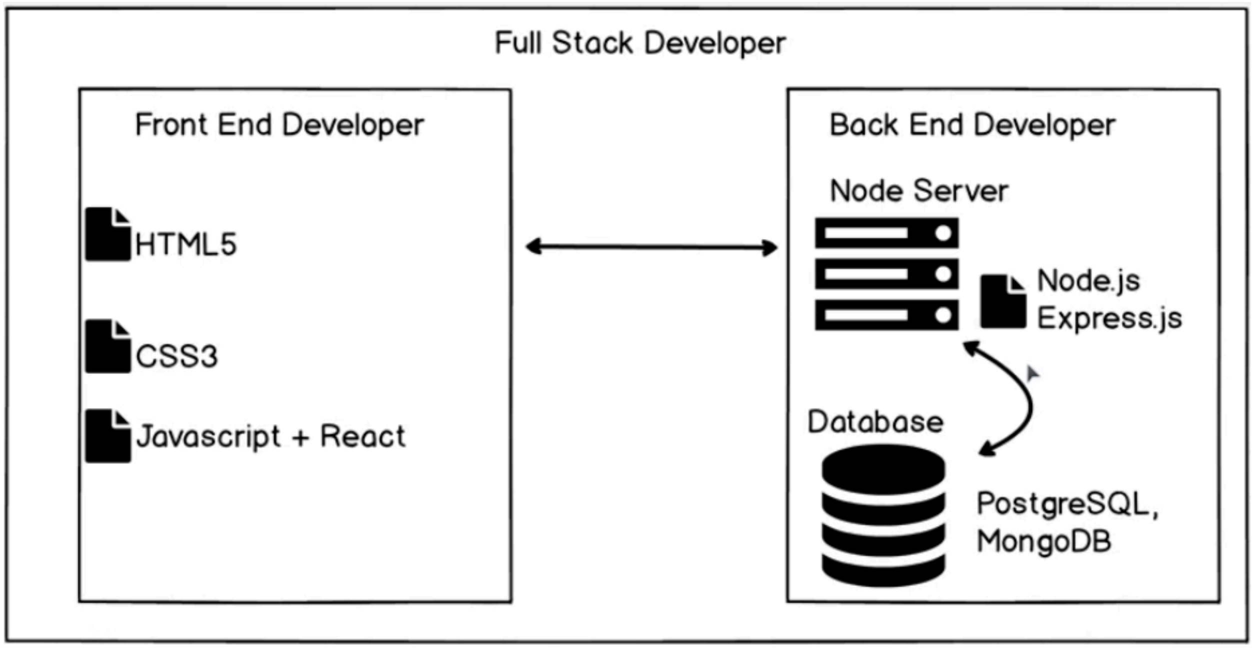
originally Netscape - Javascript performs ***actions***


A Web Page
http://www.google.com
Google Servers
172.217.7.23
HTML
CSS
Javascript
Netscape Communicator 4.75
NETSCAPE
double-click to select image
⌘double-click, ENTER or F2 to edit text

Now- web, apps, VR, drones, robotics



Javascript used in Node.js, Express.js, react (backend)

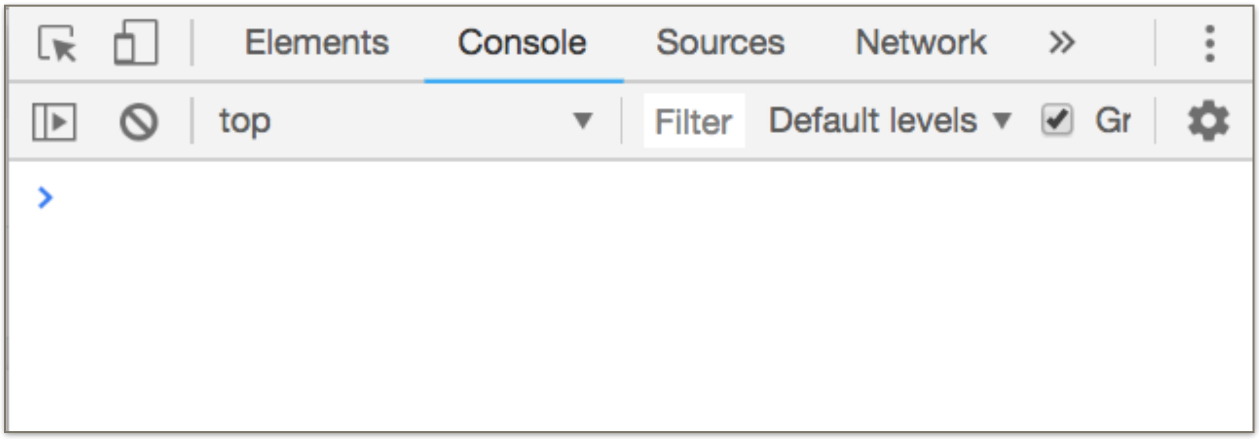


Javascript is **file** to write instructions to computer

In web - performs actions

Javascript

Chrome Developer Tools > Console



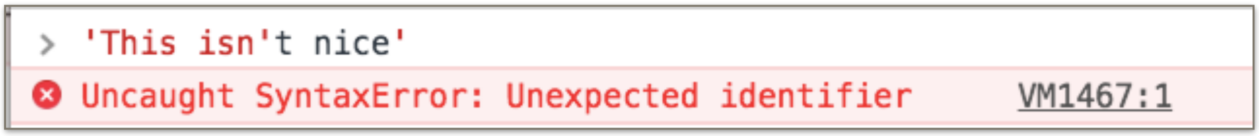
To clear console: clear( ) or ⃠

Javascript Types

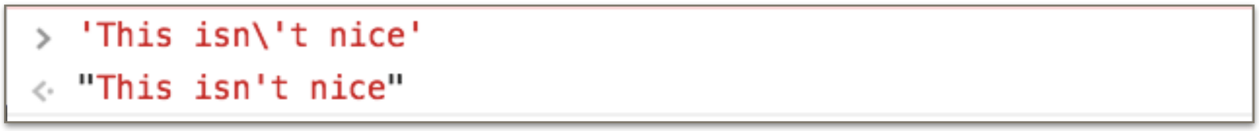
1. Number
2. String
3. Boolean
4. Undefined
5. Null
6. Symbol (new in ECMAScript 6)
7. Object

JavaScript Escape Character: \

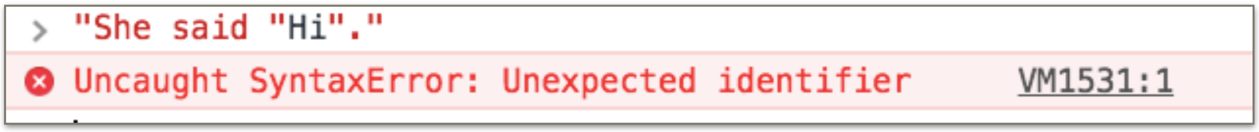
Use \ in front of quote to prevent error



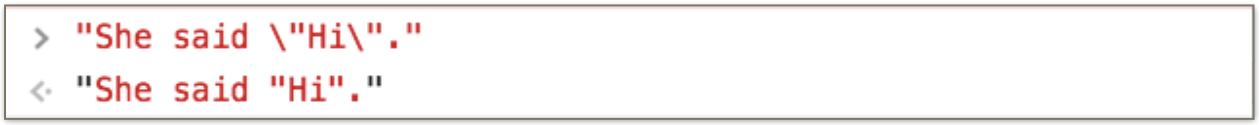
```
> 'This isn't nice'
⊗ Uncaught SyntaxError: Unexpected identifier        VM1467:1
```



```
> 'This isn\'t nice'
<· "This isn't nice"
```

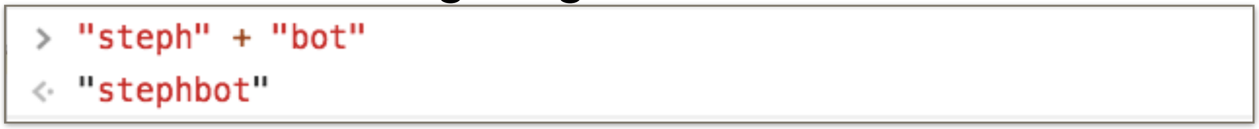


```
> "She said "Hi"."
⊗ Uncaught SyntaxError: Unexpected identifier        VM1531:1
```



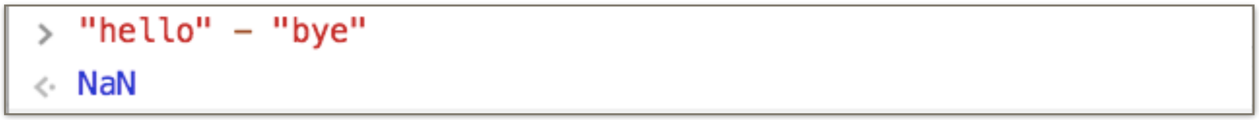
```
> "She said \"Hi\"."
<· "She said "Hi"."
```

add strings together to concatenate



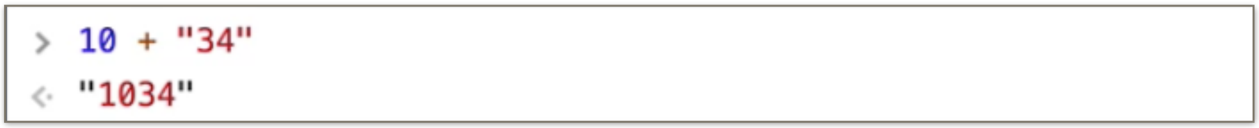
```
> "steph" + "bot"
<· "stephbot"
```

string - string = NaN



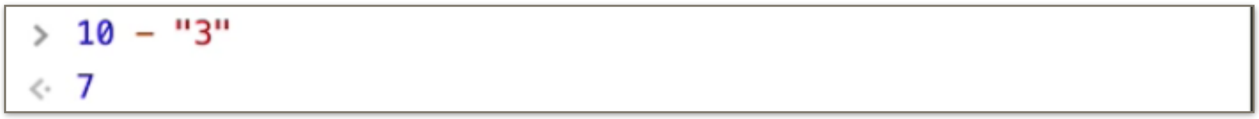
```
> "hello" - "bye"
<· NaN
```

string + number = string



```
> 10 + "34"
<· "1034"
```

number - string = number

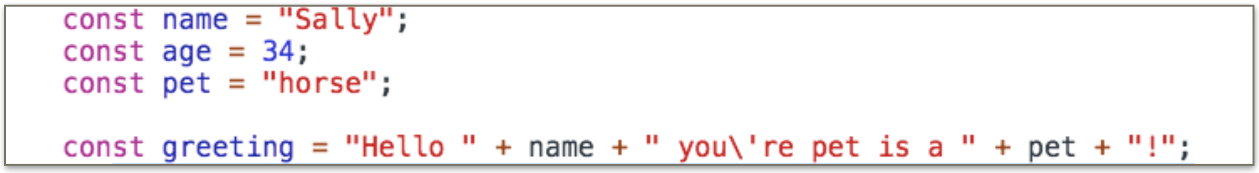


```
> 10 - "3"
<· 7
```

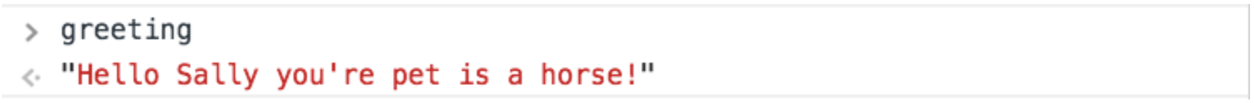
Template Strings (ES5/ES6 only)

- Skip escape character for quotes
- Denote variables within strings more conveniently
- Dynamic variables allowed

Normally, must be careful using quotes within a string…
denote that they are okay using the JavaScript escape character, backwards backslash \

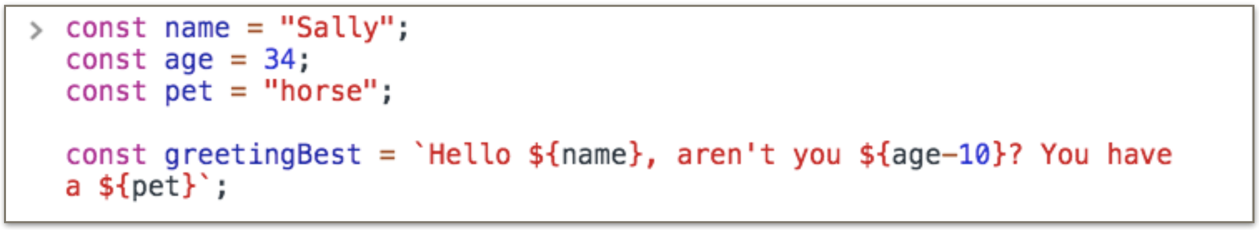


```
const name = "Sally";
const age = 34;
const pet = "horse";

const greeting = "Hello " + name + " you\'re pet is a " + pet + "!";
```

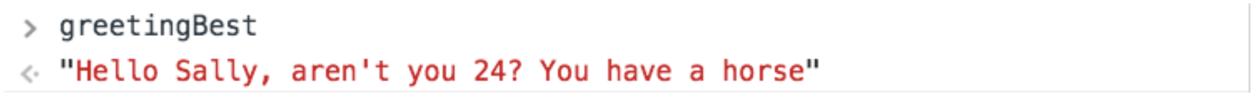


```
> greeting
<· "Hello Sally you're pet is a horse!"
```

Use back ticks `` (above tab key) so that we can use double and single quotes within a string while skipping the escape character, and denote our variables more conveniently (can be dynamic!)…



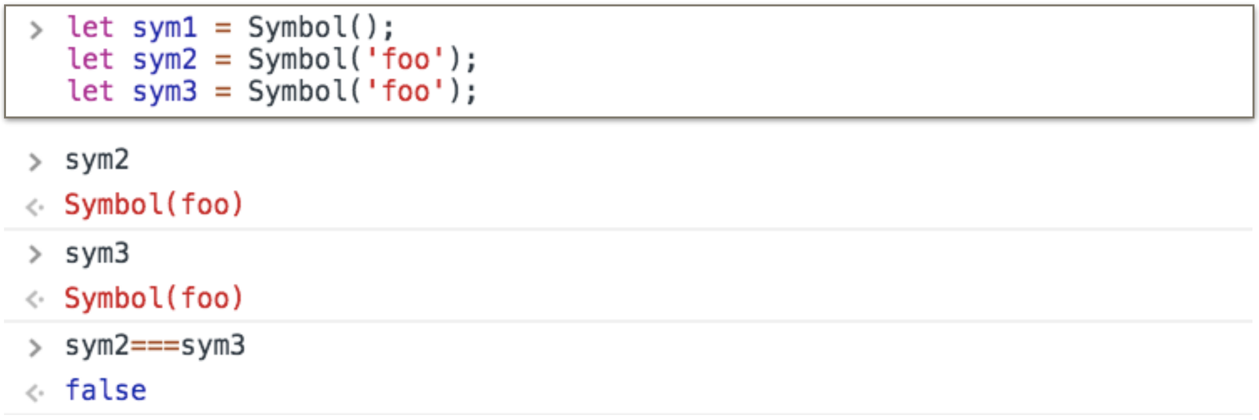
```
> const name = "Sally";
  const age = 34;
  const pet = "horse";

  const greetingBest = `Hello ${name}, aren't you ${age-10}? You have
  a ${pet}`;
```



```
> greetingBest
<· "Hello Sally, aren't you 24? You have a horse"
```

Advanced JavaScript Types: ***Symbol*** (ES5/ES6 only)

Symbols create a completely unique type, so that you can make sure there's never going to be any conflict.

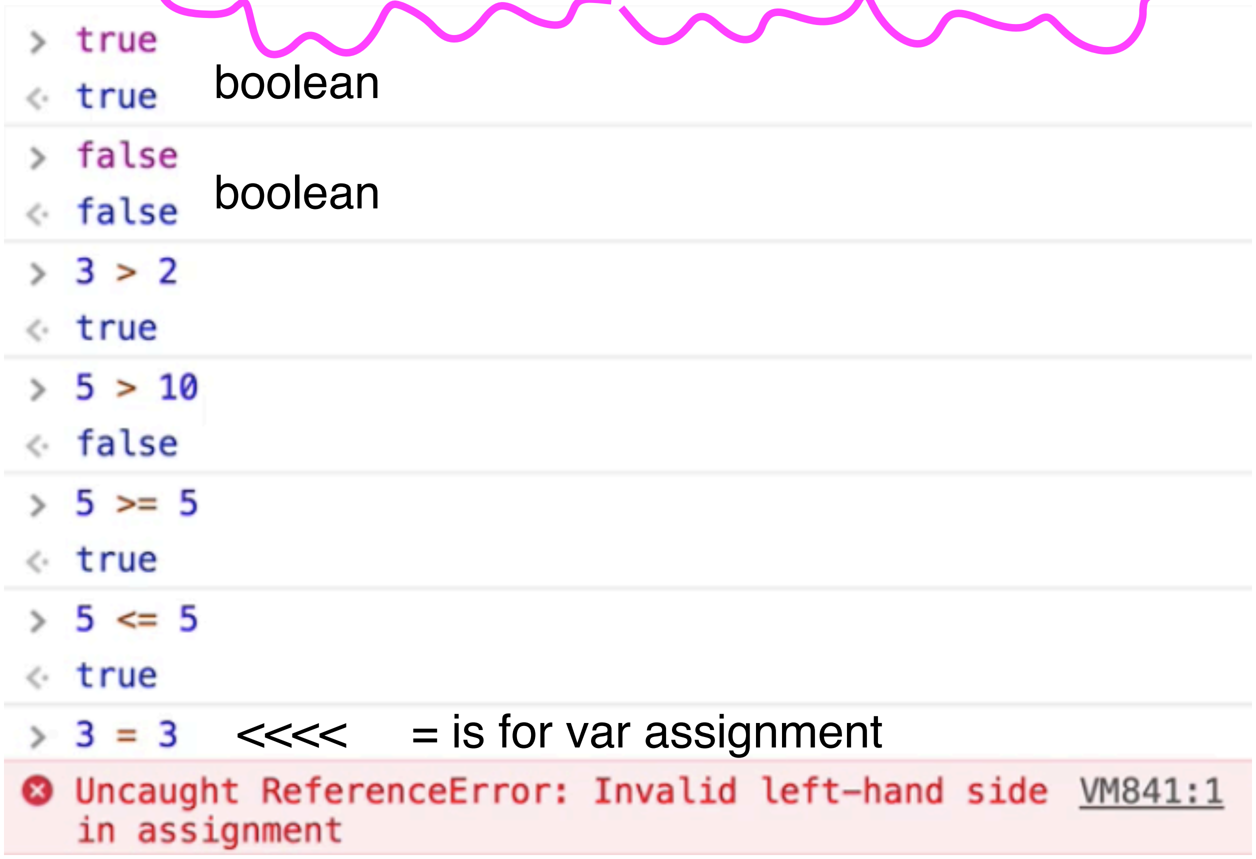


```
> let sym1 = Symbol();
  let sym2 = Symbol('foo');
  let sym3 = Symbol('foo');
> sym2
<· Symbol(foo)
> sym3
<· Symbol(foo)
> sym2===sym3
<· false
```

The symbol value is used as an indentifier mostly for object properties… because sometimes you don't want object properties (if you have thousands of them) to collide and be the same ones because then they'll get bugs.

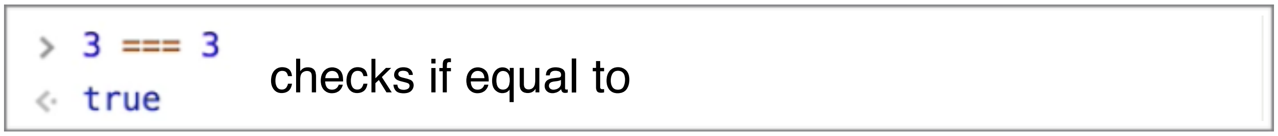
Javascript Comparisons (Boolean)

!==	not equal to	<=
===	equal to	>
>=		<

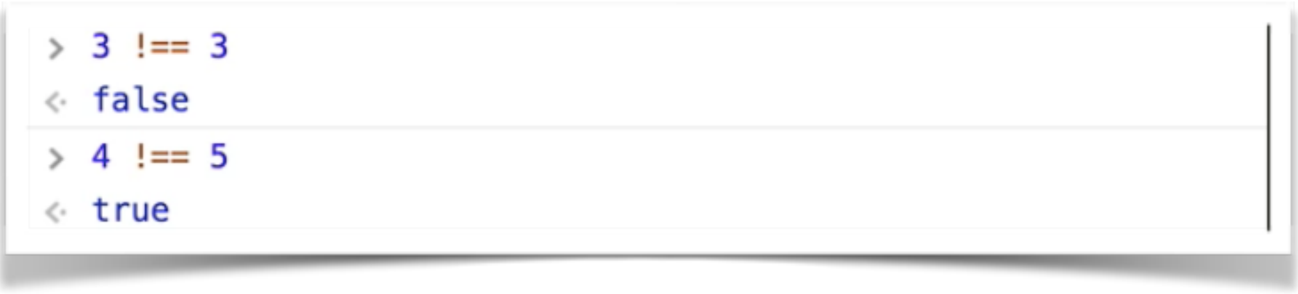


```
> true
<· true        boolean
> false
<· false       boolean
> 3 > 2
<· true
> 5 > 10
<· false
> 5 >= 5
<· true
> 5 <= 5
<· true
> 3 = 3     <<<<     = is for var assignment
✖ Uncaught ReferenceError: Invalid left-hand side    VM841:1
  in assignment
```

Use === to check if they are equal, = gives error



```
> 3 === 3
<· true        checks if equal to
```



```
> 3 !== 3
<· false
> 4 !== 5
<· true
```

includes( )

includes( ) checks to see if something is included, returns true/false

```
'Hellooooo'.includes('o');
```

```
<· true
```

```
var pets = ['cat', 'dog', 'fish'];
pets.includes('cat');
```

```
<· true
```

```
pets.includes('bird');
```

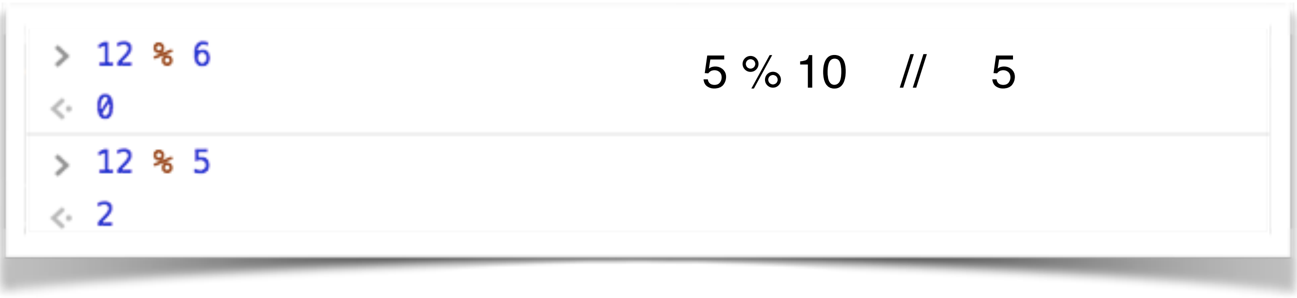
```
<· false
```

Exercise: ES7 Section 13, Lecture 149

```
// #1) Check if this array includes the name "John".
const dragons = ['Tim', 'Johnathan', 'Sandy', 'Sarah'];

dragons.includes('John');
// false
```

```
// #2) Check if this array includes any name that has "John"
inside of it. If it does, return that
// name or names in an array.
const dragons2 = ['Tim', 'Johnathan', 'Sandy', 'Sarah'];
dragons2.includes('John');

dragons2.filter(function (name) {
    return name.includes('John');
})

// ["Johnathan"]
```

Module Operator % returns remainder (division)



```
> 12 % 6
<· 0
> 12 % 5
<· 2
```

5 % 10 // 5

— exercise1.txt —

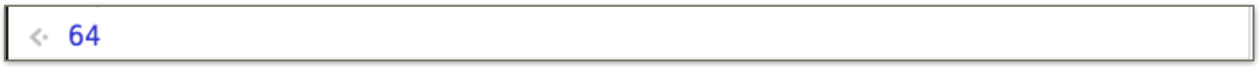
// Guess what answers you would get if you ran this in the
// Javascript Console in Google Chrome. Once you have an answer to the questions then
// check them by copying them and running it in the console yourself line by line

```
//Evaluate the below:
5 + "34"
5 - "4"
10 % 5
5 % 10
"Java" + "Script"
" " + " "
" " + 0
true + true
true + false
false + true
false - true
3 - 4
"Bob" - "bill"
```

*Exponential ***

```
var squareFunc = function squareFunc(x) {
  return x ** 2;
};

squareFunc(8);
```

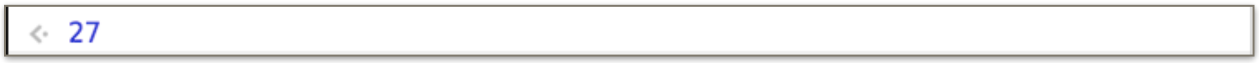
```
const squareFunc = (x) => x**2;          // ES5/6 ONLY!
squareFunc(8);
```



```
<· 64
```

```
var cubeFunc = function cubeFunc(y) {
  return y ** 3;
};

cubeFunc(3);
```

```
const cubeFunc = (y) => y**3;            // ES5/6 ONLY!
cubeFunc(3);
```



```
<· 27
```

Exercise: ES7 Section 13, Lecture 149

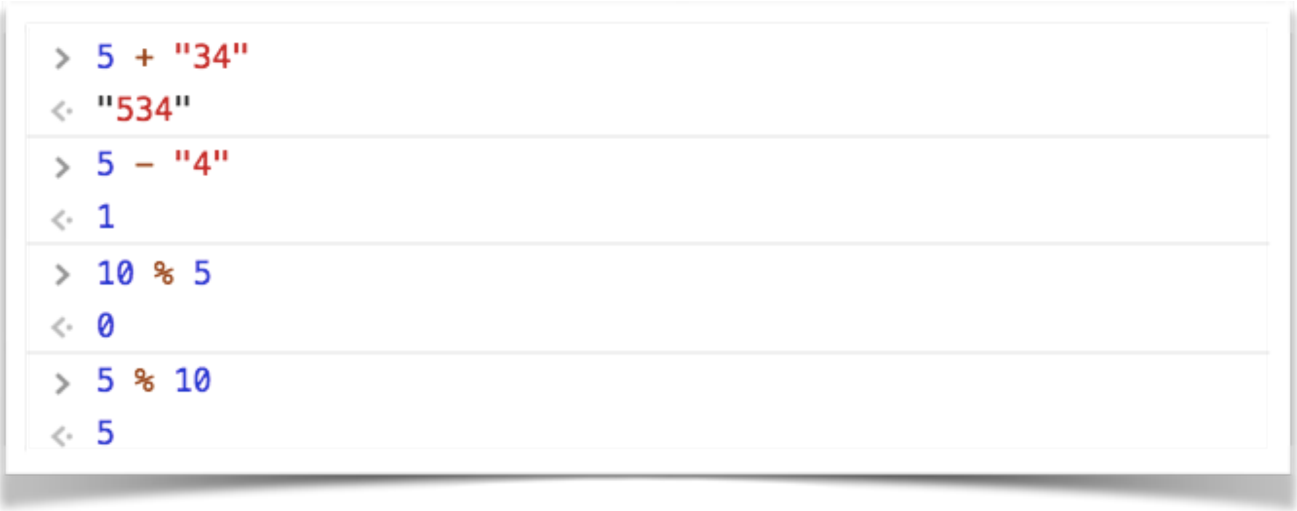
```
// #3) Create a function that calulates the power of 100 of a
number entered as a parameter
var exp100 = function (x) {
  return x ** 100;
};

// #4) Useing your function from #3, put in the paramter 10000.
What is the result?
// Research for yourself why you get this result
exp100(10000);
//Infinity
```

```
//Evaluate the below comparisons:
5 >= 1
0 === 1
4 <= 1
1 != 1
"A" > "B"
"B" < "C"
"a" > "A"
"b" < "A"
true === false
true != true
```

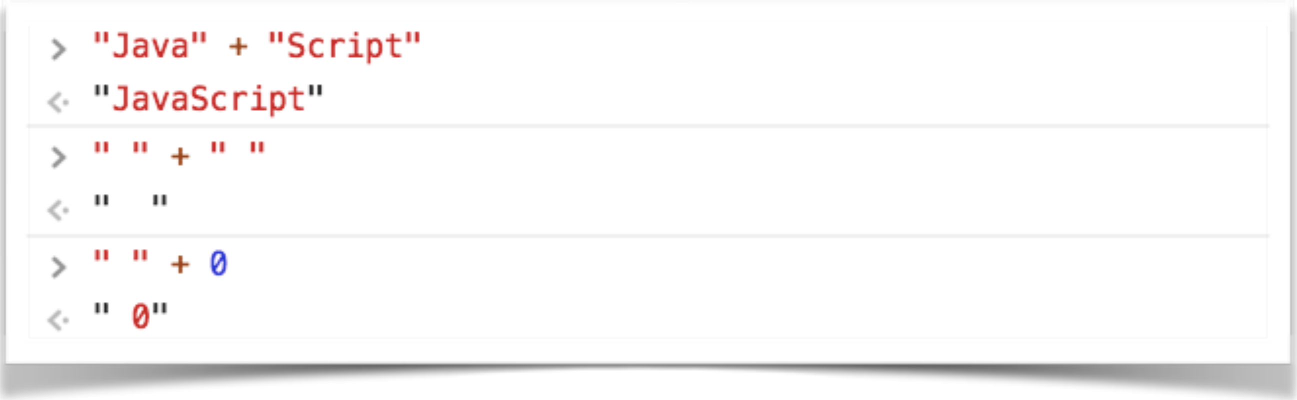
```
// Make the string: "Hi There! It's "sunny" out" by using the +
sign:
```

e x p r e s s i o n - code that produces a value
**** needs to have a semicolon at the end ****

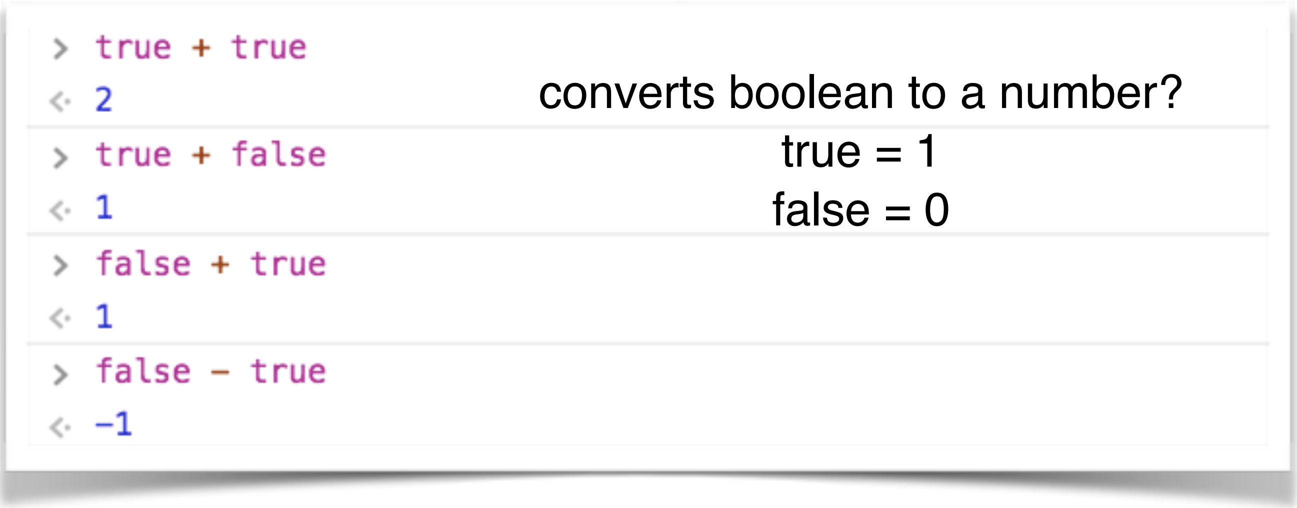
Exercise 1 solutions



```
> 5 + "34"
<· "534"
> 5 - "4"
<· 1
> 10 % 5
<· 0
> 5 % 10
<· 5
```

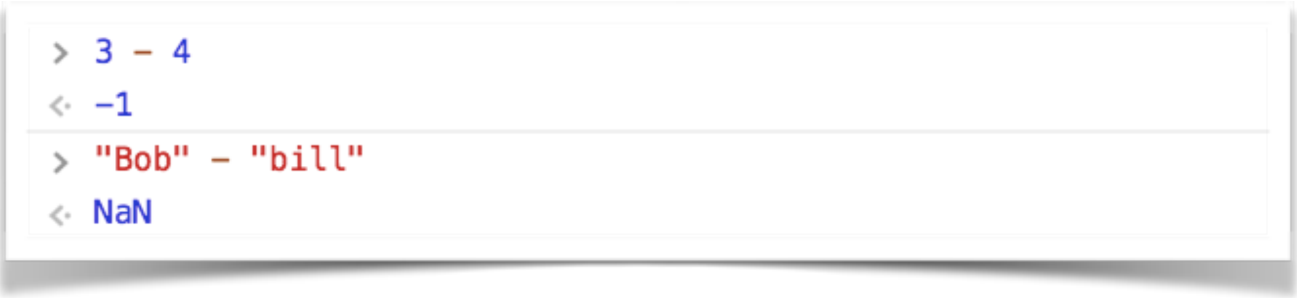


```
> "Java" + "Script"
<· "JavaScript"
> " " + " "
<· "  "
> " " + 0
<· " 0"
```

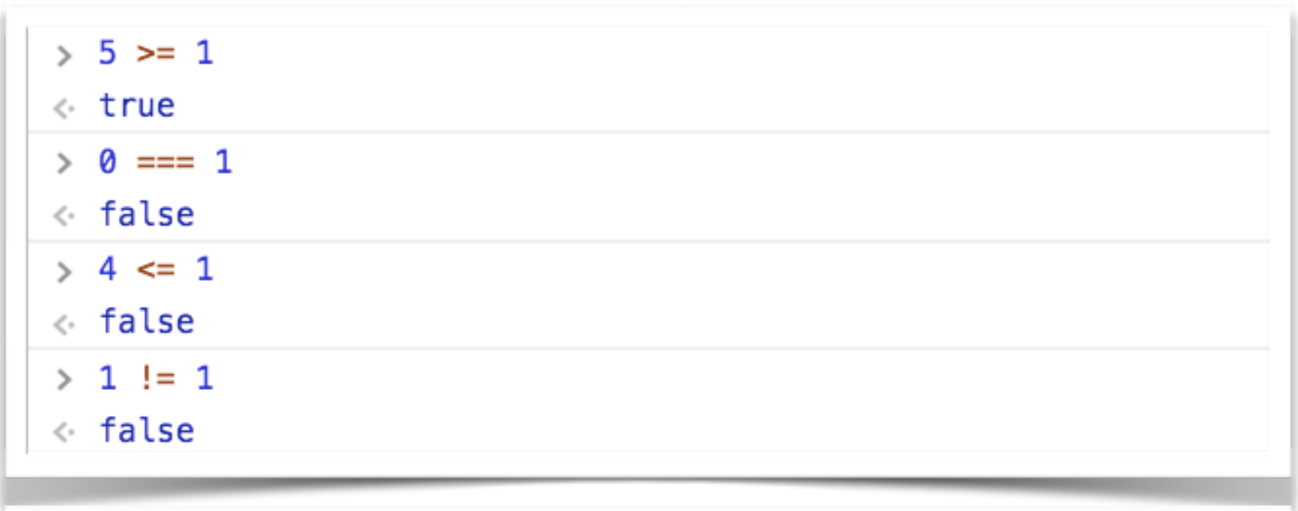


```
> true + true
<· 2
> true + false
<· 1
> false + true
<· 1
> false - true
<· -1
```

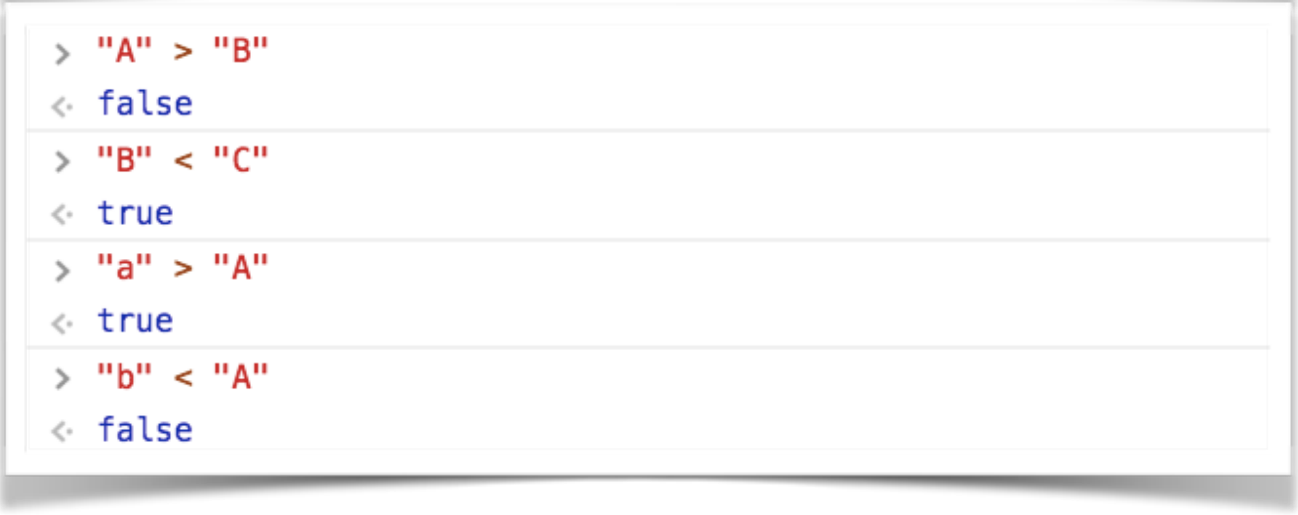
converts boolean to a number?
true = 1
false = 0



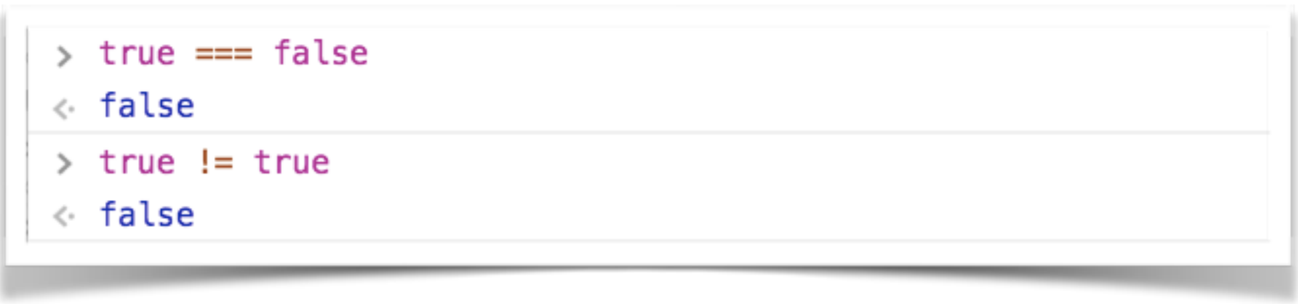
```
> 3 - 4
<· -1
> "Bob" - "bill"
<· NaN
```



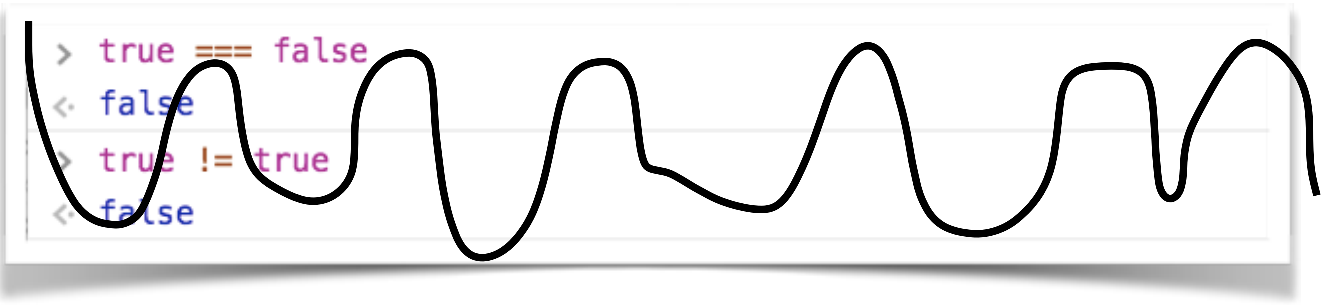
```
> 5 >= 1
<· true
> 0 === 1
<· false
> 4 <= 1
<· false
> 1 != 1
<· false
```



```
> "A" > "B"
<· false
> "B" < "C"
<· true
> "a" > "A"
<· true
> "b" < "A"
<· false
```

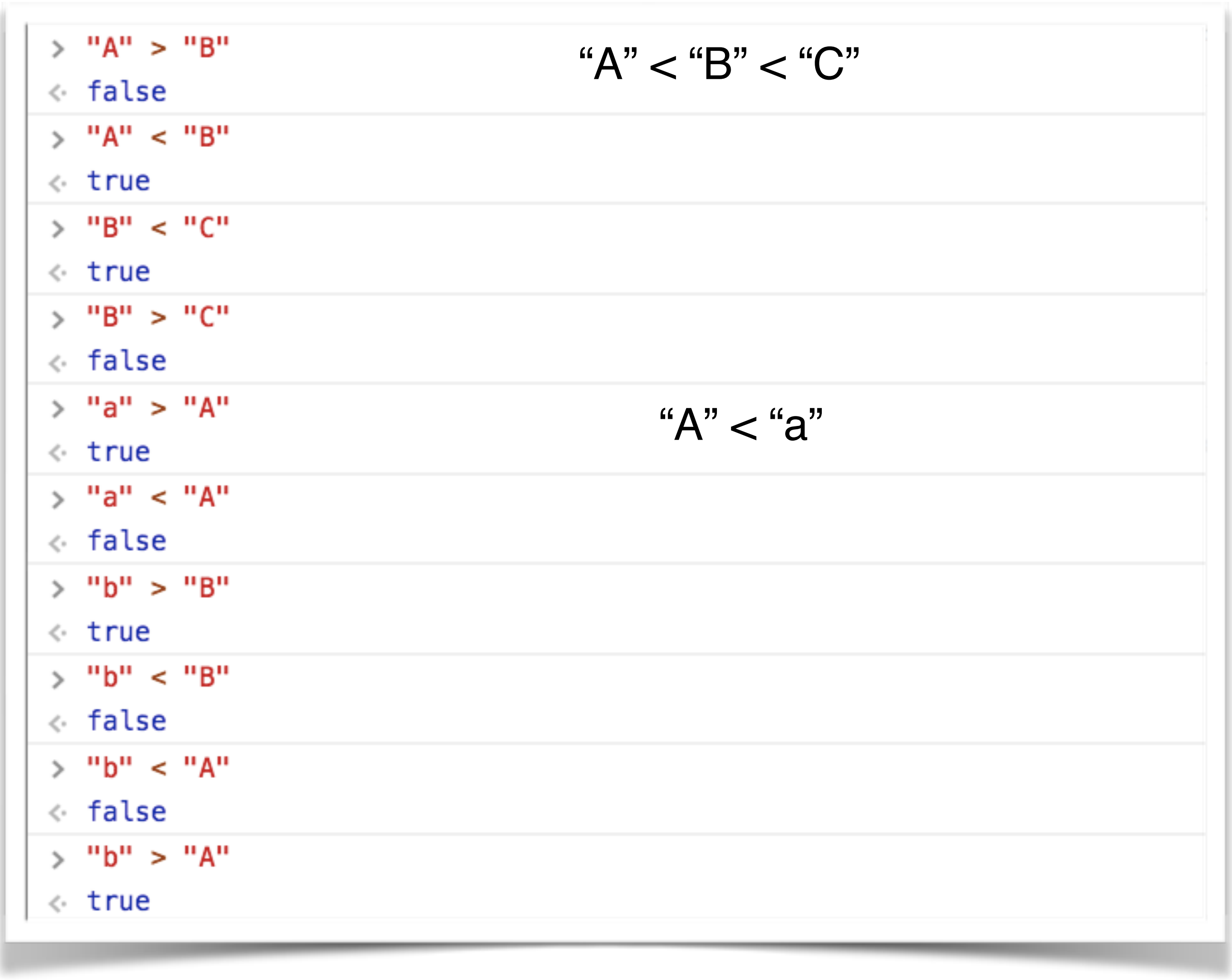


```
> true === false
<· false
> true != true
<· false
```



```
> true === false
<· false
> true != true
<· false
```

So how does this work….?



```
> "A" > "B"
<· false
> "A" < "B"
<· true
> "B" < "C"
<· true
> "B" > "C"
<· false
> "a" > "A"
<· true
> "a" < "A"
<· false
> "b" > "B"
<· true
> "b" < "B"
<· false
> "b" < "A"
<· false
> "b" > "A"
<· true
```

“A” < “B” < “C”

“A” < “a”

“ “ < “A” < “B” < “C” < “D” < … < “Z” < “a” < “b” < …

ABCD….Zabcd….z

smallest^ ^largest