Web Programming

Woche 15

HS 2019

Prof. D. König

Rückschau

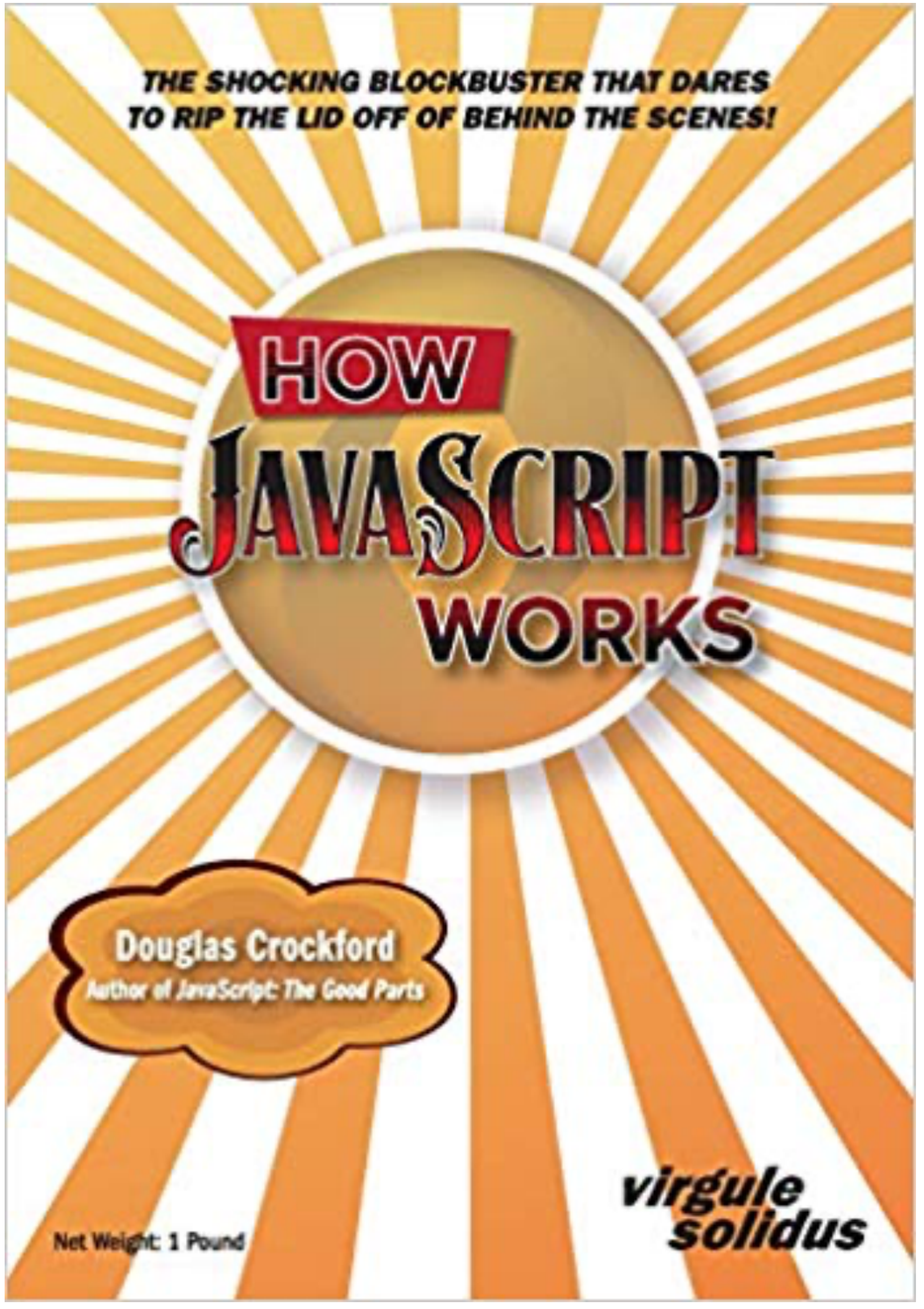
Fragen zum letzten Quiz (transpiler)

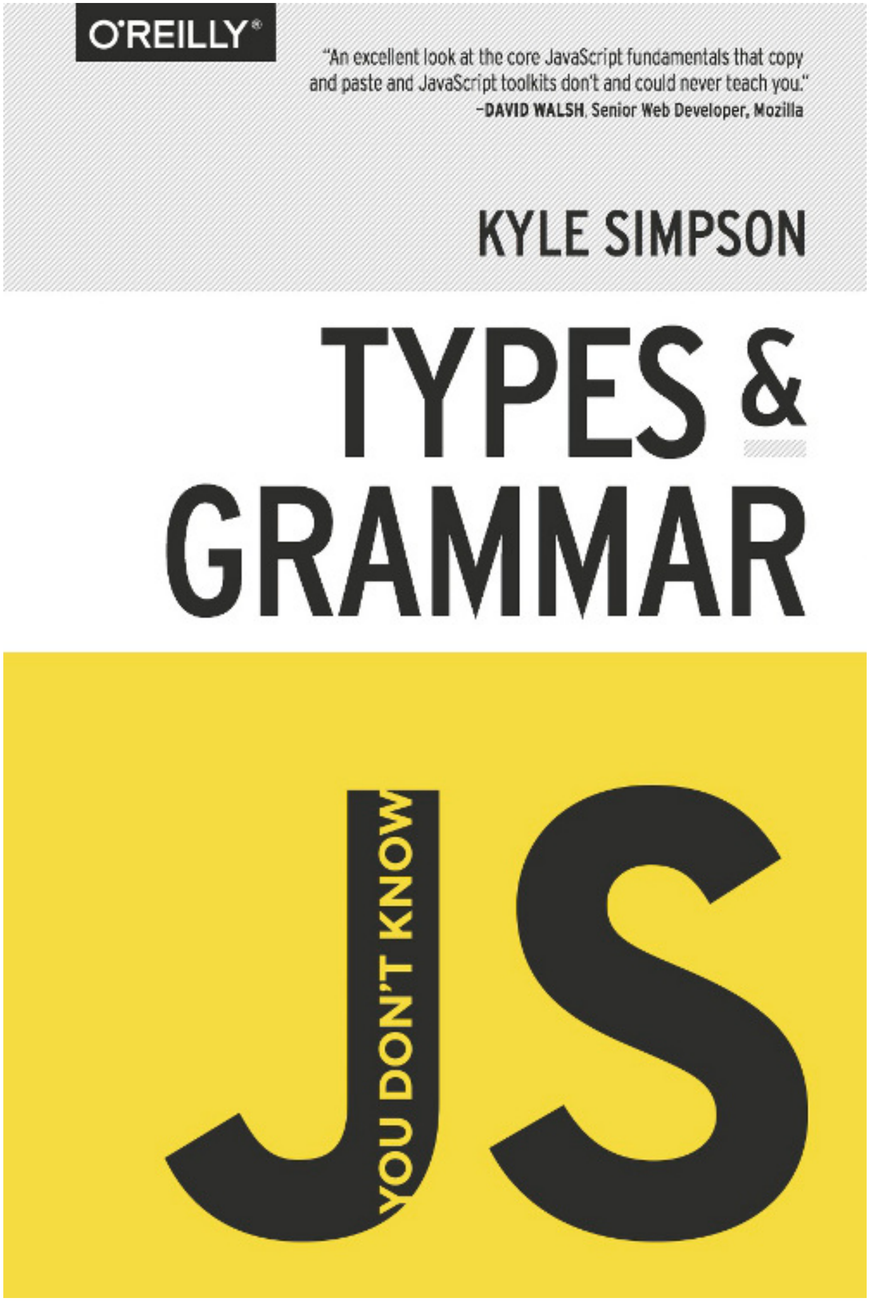
Sonstige Fragen

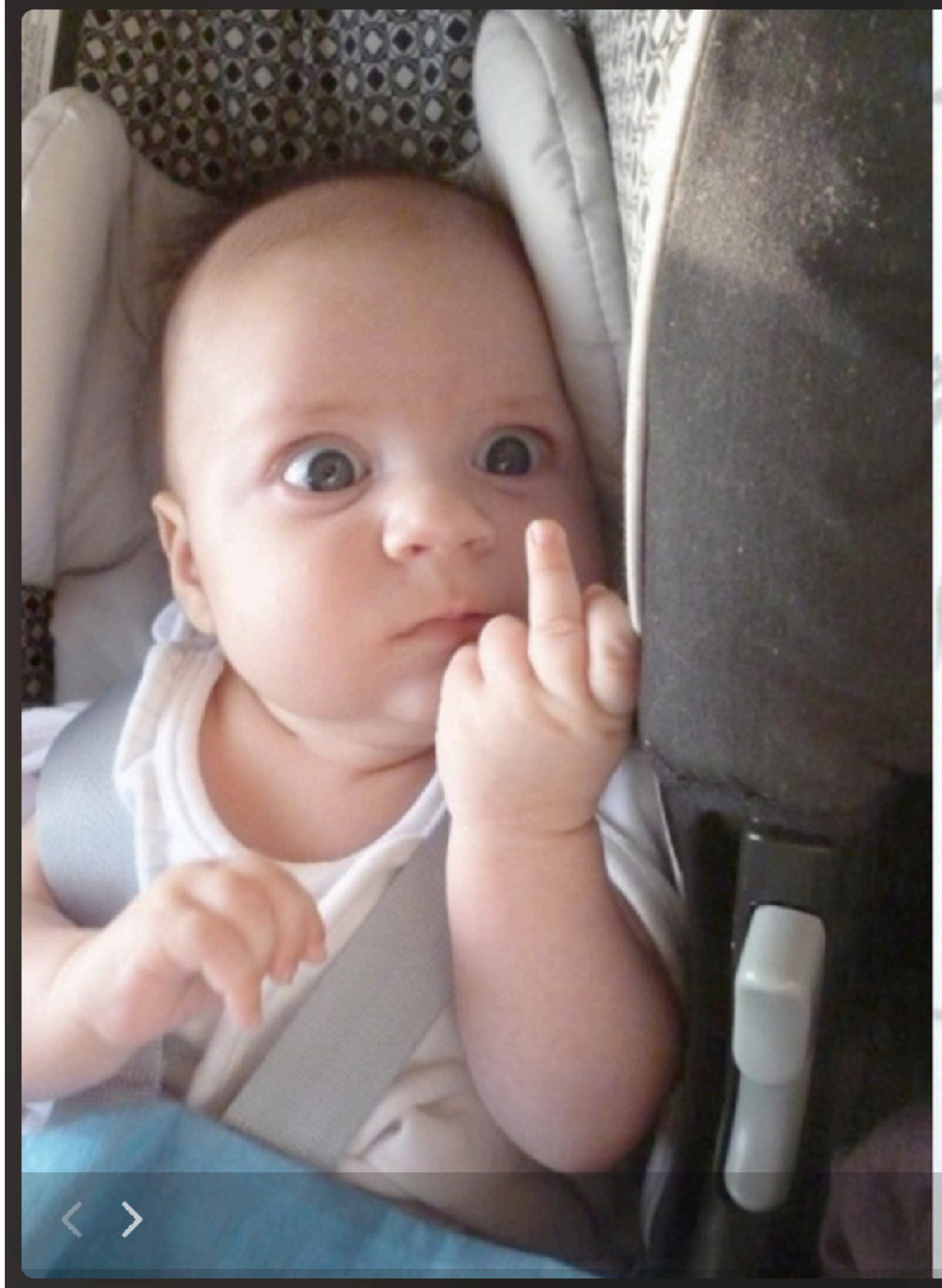
Agenda

Spass-Vorlesung Crazy JavaScript

Quiz


THE SHOCKING BLOCKBUSTER THAT DARES
TO RIP THE LID OFF OF BEHIND THE SCENES!
HOW
JAVASCRIPT
WORKS
Douglas Crockford
Author of JavaScript: The Good Parts
virgule
solidus
Net Weight: 1 Pound


O'REILLY
"An excellent look at the core JavaScript fundamentals that copy and paste and JavaScript toolkits don't and could never teach you."
–DAVID WALSH, Senior Web Developer, Mozilla
KYLE SIMPSON
TYPES & GRAMMAR
YOU DON'T KNOW
JS



what the... JavaScript?

kyle simpson
http://getify.me
@getify

http://www.waxoil.com/art/alternative/strange-art-leaf.gif

Ressources

https://speakerdeck.com/getify/what-the-javascript

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Comparison_Operators

Daniel Kurka: https://www.youtube.com/watch?v=xE8tL8NdHaY

Anette Bergo: https://www.youtube.com/watch?v=wOMLlEAqgRk

Themen

Truthiness, conditionals

Coercion, == vs ===, ordering

Deconstructors, min, max

Elvis and guards, [generators]

Anwärmen

Transitivität der Äquivalenz

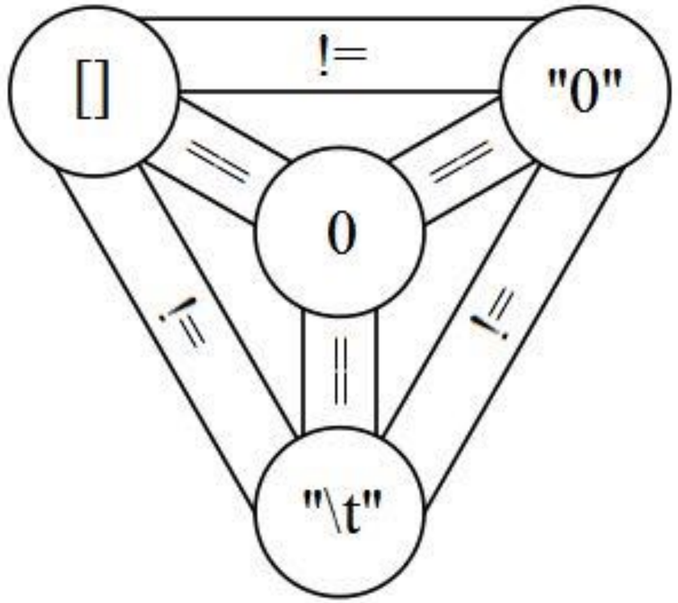
a == b == c => a == c

0 == "0"

0 == [ ]

0 == "\t"

Trinity



JavaScript @hsjoihs

False

false

null, undefined

""

0

NaN

Falsy

falsy = what coerces to false

true = everything that isn't false

truthy = what coerces to true

Bool'scher Kontext

if (_____) { "true" } else { "false" }

0, "0", u.s.w. unmodified

Coercion

implicit: "0" == 0, +"0" , !"0", !!"0"

explicit: Number("0") == 0

Daumenregel (nicht verlässlich)
object -> string -> number -> boolean

new Craziness

```
if ( ______ ) { "true" } else { "false" }

"0"                         // string

Number("0")                 // coercion number

new Number("0")             // ?
```

== vs ===

== coerces the operands

=== does not coerce

contrary to popular belief...

Crazy ==

if(x)
if(x == true)

should be the same as
for all x

Crazy ==

```
if(x)                    should be the same as
if(x == true)             for all x, but

if("0")                  // no coercion

if("0" == true)          // coercion
```

Takeaway

prefer === over ==

remaining case:
(a == null) better than
(a === null || a === undefined)

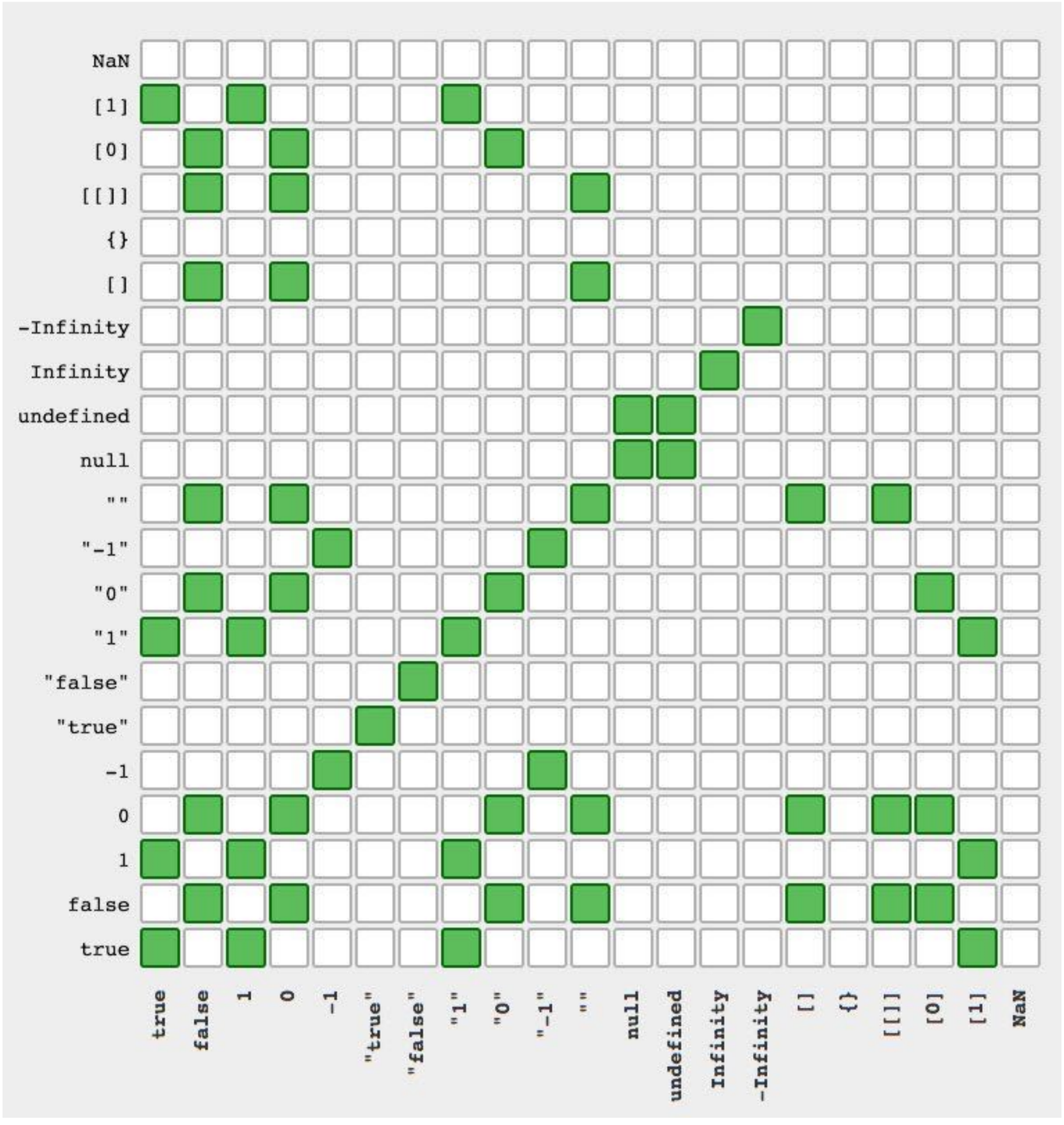
More coercion effects

```
"2" + 1,  "2" - 1,  "2" - - 1,  1 + 2 + "3"

+true,  +false,  true + true

[] == [],  [] == ![],  +[],  2 == [2]

[] + {},   {} + []
```


NaN
[1]
[0]
[[]]
{}
[]
-Infinity
Infinity
undefined
null
""
"-1"
"0"
"1"
"false"
"true"
-1
0
1
false
true
true
false
1
0
-1
"true"
"false"
"1"
"0"
"-1"
""
null
undefined
Infinity
-Infinity
[]
{}
[[]]
[0]
[1]
NaN

Crazy Numbers

```
Number("-0")       // -0
JSON.parse("-0") // -0  but JSON.stringify(-0) // "0"
String(-0) // "0"
typeof null, null instanceof Object,
typeof NaN, NaN == NaN, NaN == !NaN
```

Crazy Numbers

```
typeof (1/0)

0.1 + 0.2 === 0.3

9999999999999999

Number.MAX_VALUE > 0

Number.MIN_VALUE < 0  // ?
```

Crazy Numbers

```
Math.min(1, 2, 3) < Math.max(1, 2, 3)

Math.min() < Math.max()    // ?
```

Crazy Comparison

Comparison coerces to numbers

1 < 2 < 3 === true

3 > 2 > 1 === false // why?

Crazy Comparison

{} == {}

{} > {}

{} >= {}

Cool: Generators

```
function *foo() {
    console.log( "a:", yield 1 );
    console.log( "b:", yield 2 );
}

var it = foo();

it.next();          // { value:1, done:false }

it.next(5);         // a: 5
                    // { value:2, done:false }

it.next(10);        // b: 10
                    // { value:undefined, done:true }
```

Cool: Deconstructor

```
function foo() {
    var x = 2, y = 3;

    return { x: x, y: y };
}

var o = foo(),
    x = o.x, y = o.y;

console.log(x,y);          // 2 3
```

```
function foo() {
    var x = 2, y = 3;

    return { x: x, y: y };
}

var { x, y } = foo();    // <-- awesome!

console.log(x,y);          // 2 3
```

```
function foo({x, y}) {        // <-- awesome!
    console.log(x,y);
}

foo( { y: 10, x: 25 } );     // 25 10

foo( { y: 5 } );                  // undefined 5
```

Cool: Elvis and Guard

```
let a = null;
let b = a ? a : "default"
let b = a || "default"

let b = a ? guardedOperation() : a
let b = a && guardedOperation()
```

Crazy but not covered

finally edge cases,

generator edge cases with return, for-of

unreal arrays/lists, e.g. "arguments"

class, new, this, valueOf

Farewell