Tem um *tipo* no meu Javascript

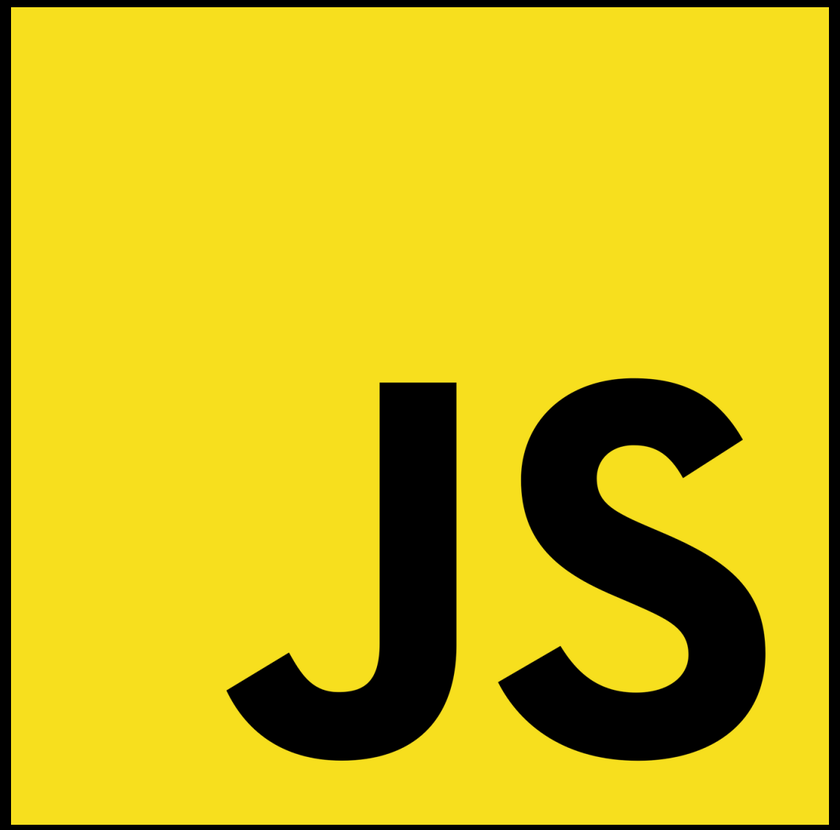
Matias H. Leidemer

Software Dev. @
Stickermule

http://leidemer.com

@matiasleidemer

Disclaimer


JS

Cart

DESCRIPTION	QUANTITY	TOTAL
 Die cut stickers 2" x 2" js.png	50	$57
 Circle stickers 1" x 1"	200	$65
Discount for multiple designs		-$23.50
Store credit		-$5.56

ADMIN Clear cart

Subtotal: $92.94

Checkout

or continue shopping

```
function sum(a, b) {
  return a + b
}

sum(1, 1)
```

```
function sum(a, b) {
  return a + b
}

sum(1, 1) == 2
```

```
function sum(a, b) {
  return a + b
}

sum('1', 1)
```

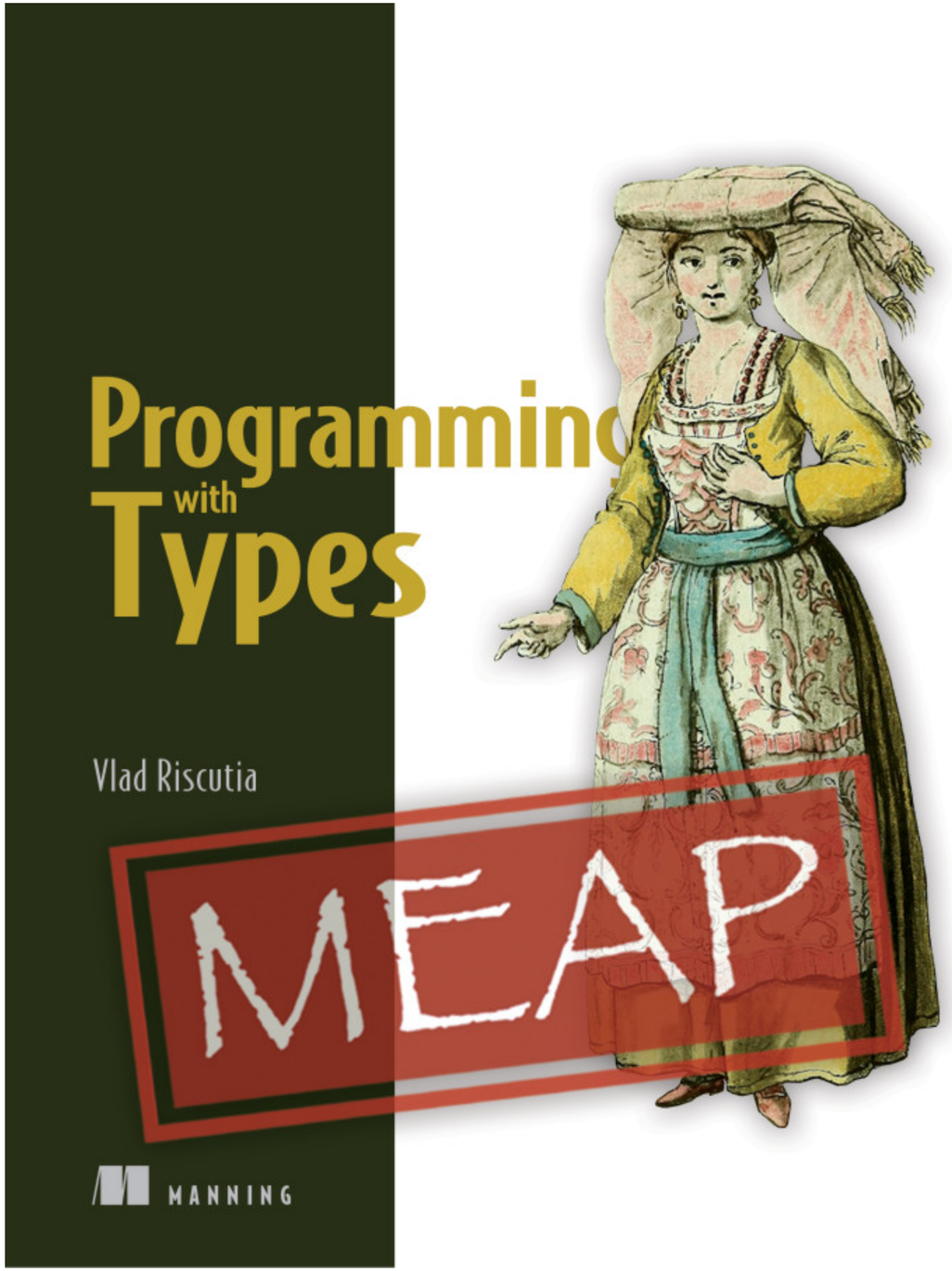
```
function sum(a, b) {
  return a + b
}

sum('1', 1) == '11'
```

```
function sum(a, b) {
  return a + b
}

sum({}, [])
```

```
function sum(a, b) {
  return a + b
}

sum({}, []) == '[object Object]'
```

Types

"The Mars Climate Orbiter disintegrated into the planet's atmosphere because a component developed by Lockheed produced momentum measurements in pound-force seconds (US units) while another component developed by NASA expected momentum to be measured in Newton seconds (metric units).
***That was a type mismatch**."*



https://www.manning.com/books/programming-with-types

Static Typing

"In a statically typed language, variables, parameters and members of objects have types that the compiler knows at compile time. The compiler can use that information to perform type checks and to optimize the compiled code."

http://2ality.com/2013/09/types.html

```
//go

package main

func main() {
    sum("a", "b")
}

func sum(a int, b int) int {
    return a + b
}
```

```
go build sum.go
./sum.go:4:6: cannot use "a" (type string)
              as type int in argument to sum
./sum.go:4:11: cannot use "b" (type string)
               as type int in argument to sum
```

Dynamic Typing

"Dynamic type checking is the process of verifying the type safety of a program at runtime. (...) By definition, dynamic type checking may cause a program to fail at runtime. In some programming languages, it is possible to anticipate and recover from these failures. In others, type-checking errors are considered fatal."

https://en.wikipedia.org/wiki/Type_system

```ruby
# Ruby

def sum(a, b)
  puts a + b
end

sum(1, '1')

# => TypeError: String can't be coerced into Fixnum
#    from (irb):3:in `+'
#    from (irb):3:in `sum'
#    from (irb):5
```

```
// JS

'3' * '4'
12

Number(true)
1

Number('123')
123

String(true)
('true')
```

https://medium.freecodecamp.org/js-type-coercion-explained-27ba3d9a2839

Conceito

x

Ferramenta

flow

FLOW IS A STATIC TYPE CHECKER FOR JAVASCRIPT.

```
// @flow

function sum(a: number, b: number): number {
  return a + b
}

sum('1', 1)
```

```
$ yarn run flow
~/Code/flowtype-talk/node_modules/.bin/flow

Error ---------------------------------  src/01_getting_started.js:7:5
Cannot call sum with '1' bound to a because string [1]
is incompatible with number [2].

 [2]  3| function sum(a: number, b: number): number {
      4|   return a + b
      5| }
      6|
 [1]  7| sum('1', 1)
      8|

Found 1 error
error Command failed with exit code 2.
```

```
// @flow

function sum(a: number, b: number): number {
  return a + b
}

sum(1, 1)
```

```
$ yarn run flow
~/Code/flowtype-talk/node_modules/.bin/flow

No errors!
✨  Done in 0.27s.
```

Setup

```
$ yarn add -D flow-bin
```

```
$ flow init
```

```
// .flowconfig

[ignore]
<PROJECT_ROOT>/lib/.*

[include]

[libs]

[lints]

[options]

[strict]
```

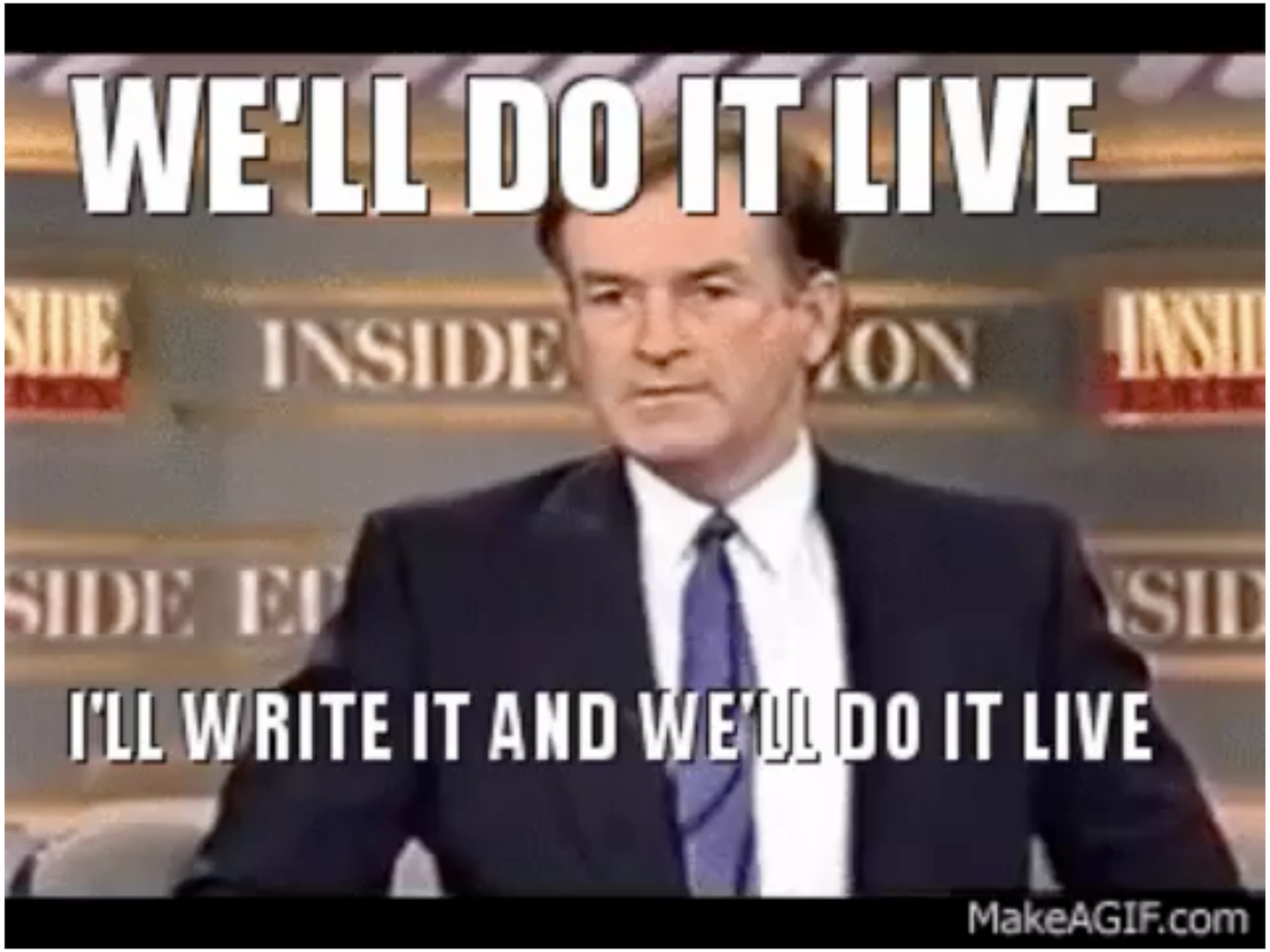
```
$ yarn run flow
~/Code/flowtype-talk/node_modules/.bin/flow

No errors!
✨  Done in 0.27s.
```


BABEL

```
$ yarn add -D \
    @babel/core \
    @babel/cli \
    @babel/preset-flow
```

```
//.babelrc

{
   "presets": ["@babel/preset-flow"]
}
```

Primitive Types - Literal Types - Mixed Types - Any Types - Maybe Types - Variable Types - Function Types - Object Types - Array Types - Tuple Types - Class Types - Type Aliases - Opaque Type Aliases - Interface Types - Generic Types - Union Types - Intersection Types - Typeof Types - Type Casting Expressions - Utility Types - Module Types - Comment Types


WE'LL DO IT LIVE
I'LL WRITE IT AND WE'LL DO IT LIVE
MakeAGIF.com

Primitives

primitives.js

Arrays

arrays.js

Functions

Aliases

aliases.js

Modules

modules.js types.js

Business Logic

business_logic.js

React

react.js

Benefits

→ Simple syntax;

→ Relatively easy to learn;

→ Helps reducing bugs;

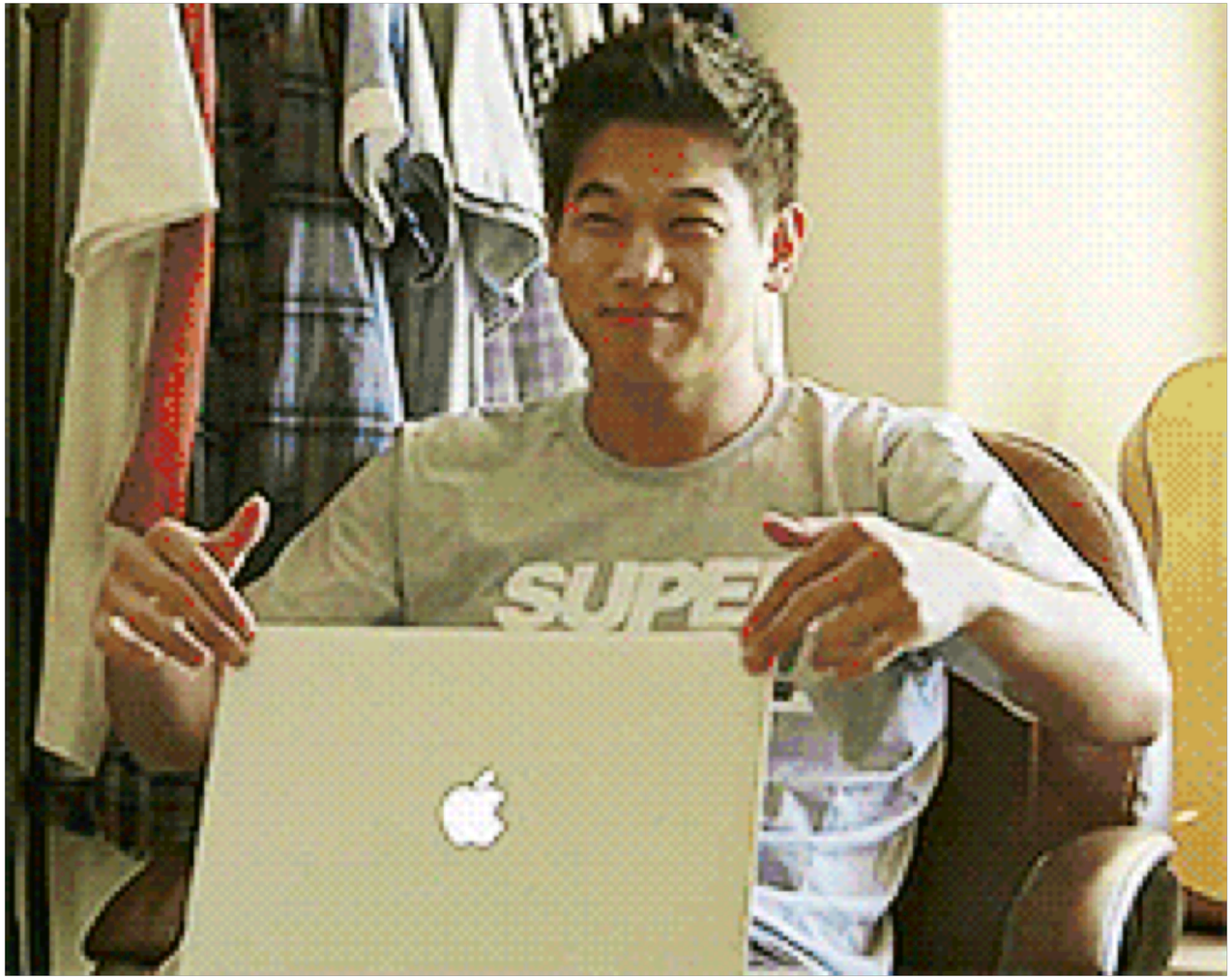
→ Helps with business logic;

→ Easy to integrate, even in big projects;

→ Integrates nice with text editors;

→ It's self documenting;

→ Makes maintaining code safer;



https://flow.org/
https://www.saltycrane.com/flow-type-cheat-sheet
https://www.destroyallsoftware.com/talks/wat
http://2ality.com/2013/09/types.html
https://en.wikipedia.org/wiki/Type_system
http://www.adamsolove.com/js/flow/type/2016/04/13/modeling-with-adts.html
http://thejameskyle.com/adopting-flow-and-typescript.html
https://github.com/rpl/flow-coverage-report
https://www.manning.com/books/programming-with-types