YES, WE CAN
HASKELL!

Who is this person?

Android Developer at Detroit Labs
Haskell enthusiast / hobby hacker
@JasonStolaruk
https://github.com/jasonstolaruk

DETROIT
LABS

Let's write a function!

```
public class QuickSort {

    public static final int NUMBERS_TO_SORT = 25;

    public QuickSort() {
    }

    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<Integer>();
        Random rand = new Random();
        for (int i = 0; i < NUMBERS_TO_SORT; i++)
            numbers.add(rand.nextInt(NUMBERS_TO_SORT + 1));
        for (int number : numbers)
            System.out.print(number + " ");
        System.out.println("\nBefore quick sort\n\n");
        for (int number : quicksort(numbers))
            System.out.print(number + " ");
        System.out.println("\nAfter quick sort\n\n");
    }

    public static ArrayList<Integer> quicksort(ArrayList<Integer> numbers) {
        if (numbers.size() <= 1)
            return numbers;
        int pivot = numbers.size() / 2;
        ArrayList<Integer> lesser = new ArrayList<Integer>();
        ArrayList<Integer> greater = new ArrayList<Integer>();
        int sameAsPivot = 0;
        for (int number : numbers) {
            if (number > numbers.get(pivot))
                greater.add(number);
            else if (number < numbers.get(pivot))
                lesser.add(number);
            else
                sameAsPivot++;
        }
        lesser = quicksort(lesser);
        for (int i = 0; i < sameAsPivot; i++)
            lesser.add(numbers.get(pivot));
        greater = quicksort(greater);
        ArrayList<Integer> sorted = new ArrayList<Integer>();
        for (int number : lesser)
            sorted.add(number);
        for (int number: greater)
            sorted.add(number);
        return sorted;
    }
}
```

Make an array of random numbers.
Print out the numbers before and after sorting.

Pivot on the element at the center of the array.

Make “lesser” and “greater” arrays.

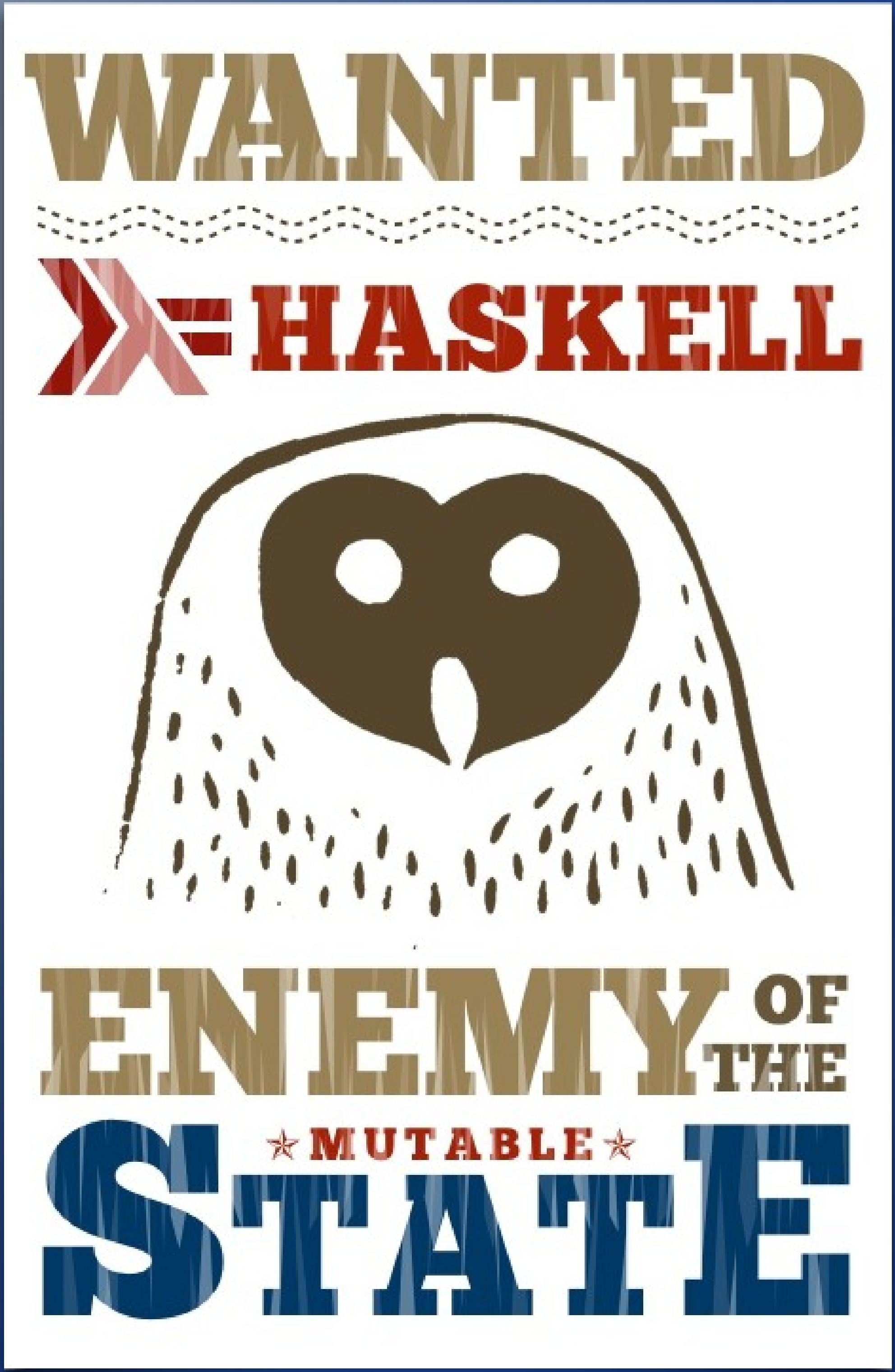
Recursively sort those arrays.

Finally, combine everything into a single sorted array.

What is this Haskell?

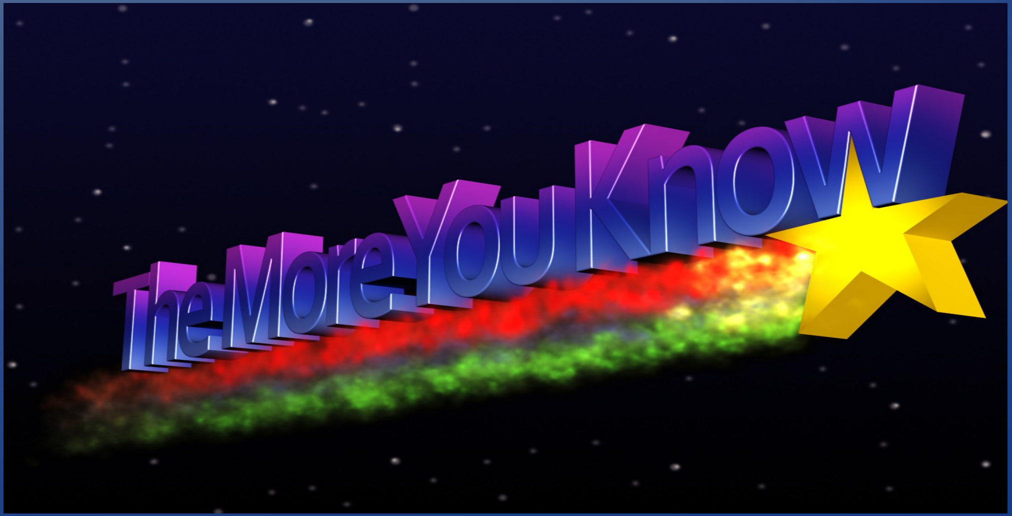
Is Haskell really so difficult?

very pure funkshunal


WANTED
HASKELL
ENEMY OF THE
MUTABLE
STATE

Why immutability?

- Mutability introduces a ton of complexity and un-certainty.
- In Haskell, there are no variables! Wow!
- Wherever you see the name of a binding, you can replace it with the value - or the expression - to which it refers.
- Referential transparency is awesome!



(More) very pure funkshunal

Hurray for higher order functions!

Functions are data, too! In fact, functions are data just like “x” is data in
`int x = 5;`
You can toss around functions just like you can toss around `x`.

such lazy eval

```java
public class Main {
    public static void main(String[] args) {
        int x = 11;
        System.out.print( foo(x, bar(x)) );
    }

    public static int foo(int a, int b) {
        if (a > 10)
            return a;
        else
            return b;
    }

    public static int bar(int x) {
        // Do some time-consuming operations on x...
        return x;
    }
}
```

(More) such lazy eval

It's good to be lazy?

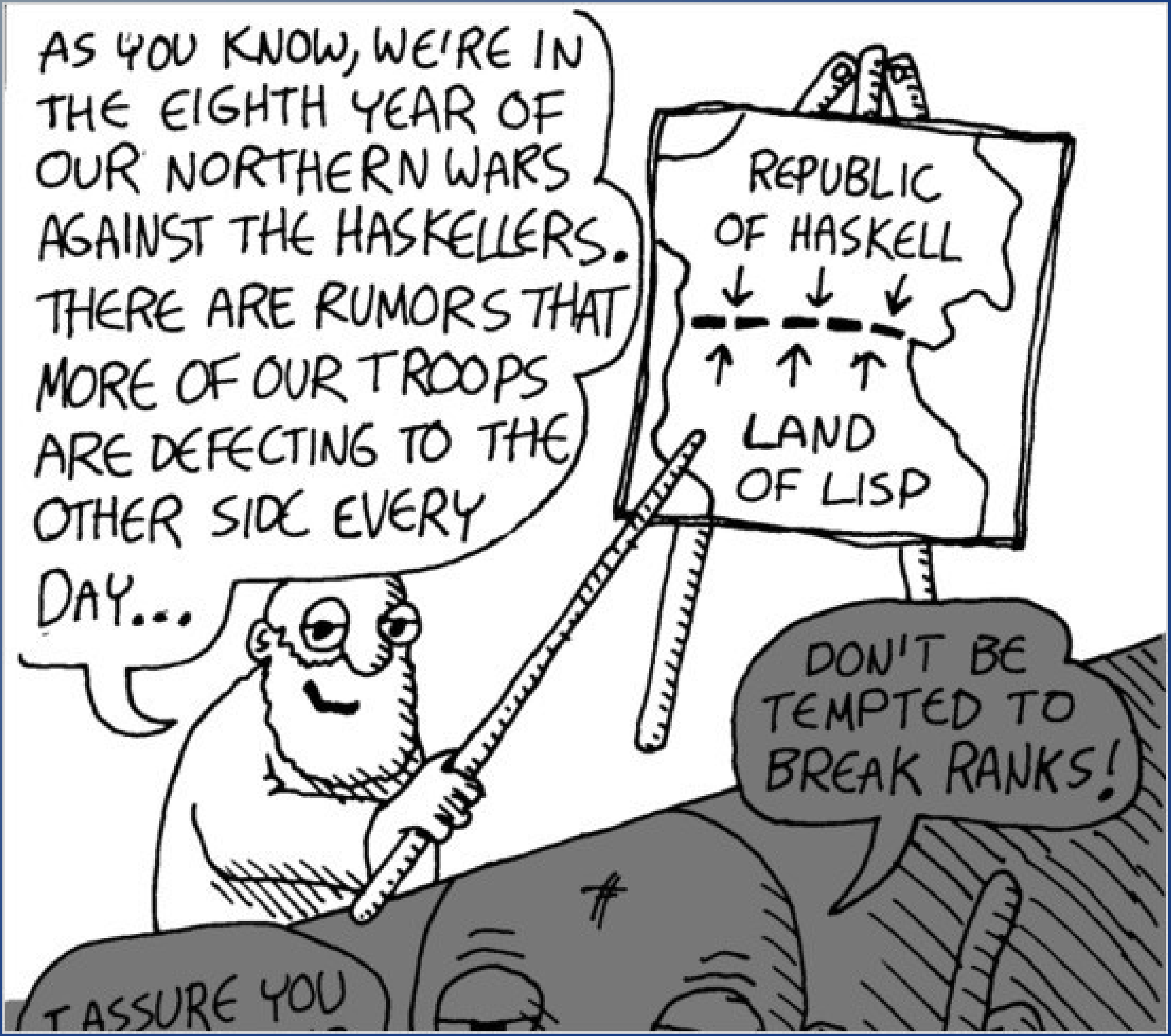
- Laziness can be more efficient.
- Infinite data structures are conceptually possible.
- Laziness allows for simpler, more elegant code.

Recap!

Haskell is pretty neat!

~~Some links:~~ A link:

- http://learnyouahaskell.com


AS YOU KNOW, WE'RE IN THE EIGHTH YEAR OF OUR NORTHERN WARS AGAINST THE HASKELLERS. THERE ARE RUMORS THAT MORE OF OUR TROOPS ARE DEFECTING TO THE OTHER SIDE EVERY DAY...
REPUBLIC OF HASKELL
LAND OF LISP
DON'T BE TEMPTED TO BREAK RANKS!
I ASSURE YOU