



# LIGHTNING TALKS

## ACCU2018

Thursday 12<sup>th</sup> April

electricity is, really just  
organized( lightning.?  
– George Carlin



# THE RULES

subjects are open!  
five minutes (max)  
have fun



**Peter Sommerlad - FOOL**  
**Michel Grootjans - Crafting Guitars**  
**Rob Smallshire - The Gender Equality Paradox**  
**Florian Gilcher - Trains**  
**Graham Haynes - On Automati**  
**Marshall Clow - Fuzzing Your Code**  
**Chris Oldwood - The Far Side**  
**Jon Kalb - This is Why We Can't Have Nice Things**  
**Phil Nash - East All The Things**  
**Jim Hague - A Brief of one-line abuses**  
**Mike Seymour - Sparsity Parsery**

The background of the image consists of rich red theater curtains with deep vertical pleats. The curtains are held back on both sides by gold-colored tassels. The lighting is dramatic, with the center of the stage area being brighter than the edges.

**SLASH  
OR  
DASH**

**DIR**

**/// SLASH ///**

**AND \**

**Peter Sommerlad - FOOL**

**Michel Grootjans - Crafting Guitars**

**Rob Smallshire - The Gender Equality Paradox**

**Florian Gilcher - Trains**

**Graham Haynes - On Automati**

**Marshall Clow - Fuzzing Your Code**

**Chris Oldwood - The Far Side**

**Jon Kalb - This is Why We Can't Have Nice Things**

**Phil Nash - East All The Things**

**Jim Hague - A Brief of one-line abuses**

**Mike Seymour - Sparsity Parsery**

# {The Problem

```
std::vector v{1,2,3,4,5,6};  
auto p=accumulate(begin(v),end(v),1, std::multiplies<>{});
```

looks very ugly:  $\langle \rangle \{ \}$

or:  $\langle \rangle ()$

or:  $\langle int \rangle \{ \}$



What else wrong in  
<functional> ?

- Many operators are missing:  
unary operators: `*`, `&`, `+`  
member access: `.*`, `->*`  
shifts, assignments, ternary, ...
- Arity is fixed: 1 or 2
- Must instantiate objects...

# A Better Way!

## P0984R0 - Functions Objects Obsoleted by Lambdas

Document Number:	P0984R0
Date:	2018-04-01
Project:	Programming Language C++
Audience:	EWG/LEWG
Target:	C++20



```
// unary operators
```

```
constexpr inline auto Deref = see below ; // unary *  
constexpr inline auto Address = see below ; // unary &  
constexpr inline auto Negate = see below ; // unary -  
constexpr inline auto Posate = see below ; // unary +  
constexpr inline auto Not = see below ; // ! not  
constexpr inline auto Cmpl = see below ; // ~ compl
```

```
// left associative binary operators
```

```
constexpr inline auto PtrMemb = see below ; // ->*  
constexpr inline auto RefMemb = see below ; // .*
```

```
constexpr inline auto Plus = see below ; // +  
constexpr inline auto Minus = see below ; // -  
constexpr inline auto Times = see below ; // *  
constexpr inline auto Divide = see below ; // /  
constexpr inline auto Remainder = see below ; // %
```

# More FOOL!



```
constexpr inline auto Equal = see below ; // ==
using equalTea = decltype(Equal); // to replace equal_to<>
constexpr inline auto Not_eq = see below ; // !=
constexpr inline auto Bigger = see below ; // >
using moreTea = decltype(Bigger); // to replace greater<>
constexpr inline auto Smaller = see below ; // <
using lessTea = decltype(Smaller); // to replace less<>
constexpr inline auto Maybe_bigger = see below ; // >=
constexpr inline auto Sometimes_smaller = see below ; // <=
constexpr inline auto Spaceship = see below ; // <=>
constexpr inline auto And = see below ; // && and
constexpr inline auto Or = see below ; // || or

constexpr inline auto Bitand = see below ; // & bitand
constexpr inline auto Bitor = see below ; // | bitor
constexpr inline auto Xor = see below ; // ^ xor
constexpr inline auto Lshift = see below ; // << left shift
constexpr inline auto Rshift = see below ; // >> right shift
constexpr inline auto Assign = see below ; // =
constexpr inline auto PlusAssign = see below ; // +=
constexpr inline auto MinusAssign = see below ; // -=
constexpr inline auto TimesAssign = see below ; // *=
constexpr inline auto DivideAssign = see below ; // /=
constexpr inline auto RemainderAssign = see below ; // %=
constexpr inline auto And_eq = see below ; // &=
constexpr inline auto Or_eq = see below ; // |=
constexpr inline auto Xor_eq = see below ; // ^=
constexpr inline auto LshiftAssign = see below ; // <<=
constexpr inline auto RshiftAssign = see below ; // >>=
```

# Further Usage

```
auto endl=static_cast<std::ostream&(*)>(std::ostream&)(std::endl);  
Lshift(std::cout, "Hello", ' ', "World!", endl, "The Answer is: ", 6*7, endl);
```

```
std::set<int, lessTea> s({3,1,4,1,5,9,2,6}, Smaller); // no C++20 compiler  
std::set<int, lessTea> s({3,1,4,1,5,9,2,6}); // C++20 allows creating from decltype(lambda)
```

# See below?

- Lambda constexpr variable
- Variadic Lambda
- Deduced Noexcept
- Deduced Return Type
- Fold Expression
- No specializations
- No overloads

# How?

```
std::vector v{1,2,3,4,5,6};  
auto res=accumulate(begin(v),end(v),1,Times);
```

# How?

```
std::vector v{1,2,3,4,5,6};  
auto res=accumulate(begin(v),end(v),1,Times);
```

variadic lambdas & folds

# How?

```
std::vector v{1,2,3,4,5,6};  
auto res=accumulate(begin(v),end(v),1,Times);
```

## variadic lambdas & folds

```
constexpr auto Times=[](auto&&... l) constexpr
```



# How?

```
std::vector v{1,2,3,4,5,6};  
auto res=accumulate(begin(v),end(v),1,Times);
```

## variadic lambdas & folds

```
constexpr auto Times=[](auto&&... l) constexpr  
noexcept(noexcept(... * std::declval<decltype(l)>()))
```

# How?

```
std::vector v{1,2,3,4,5,6};  
auto res=accumulate(begin(v),end(v),1,Times);
```

## variadic lambdas & folds

```
constexpr auto Times=[](auto&&... l) constexpr  
noexcept(noexcept(... * std::declval<decltype(l)>()))  
-> decltype(... * std::forward<decltype(l)>(l) )
```

# How?

```
std::vector v{1,2,3,4,5,6};  
auto res=accumulate(begin(v),end(v),1,Times);
```

## variadic lambdas & folds

```
constexpr auto Times=[](auto&&... l) constexpr  
noexcept(noexcept(... * std::declval<decltype(l)>()))  
-> decltype(... * std::forward<decltype(l)>(l) )  
{
```

# How?

```
std::vector v{1,2,3,4,5,6};  
auto res=accumulate(begin(v),end(v),1,Times);
```

## variadic lambdas & folds

```
constexpr auto Times=[](auto&&... l) constexpr  
noexcept(noexcept(... * std::declval<decltype(l)>()))  
-> decltype(... * std::forward<decltype(l)>(l) )  
{  
return (...*std::forward<decltype(l)>(l));
```

# How?

```
std::vector v{1,2,3,4,5,6};  
auto res=accumulate(begin(v),end(v),1,Times);
```

## variadic lambdas & folds

```
constexpr auto Times=[](auto&&... l) constexpr  
noexcept(noexcept(... * std::declval<decltype(l)>()))  
-> decltype(... * std::forward<decltype(l)>(l) )  
{  
return (...*std::forward<decltype(l)>(l));  
};
```

Unary Ops:

# Unary Ops:

```
constexpr inline auto Posate = [](auto&& l) constexpr
```

# Unary Ops:

```
constexpr inline auto Posate = [](auto&& l) constexpr  
noexcept(noexcept( + std::declval<decltype(l)>()))
```



# Unary Ops:

```
constexpr inline auto Posate = [](auto&& l) constexpr  
noexcept(noexcept( + std::declval<decltype(l)>()))  
-> decltype(+ std::forward<decltype(l)>(l))
```

# Unary Ops:

```
constexpr inline auto Posate = [](auto&& l) constexpr  
    noexcept(noexcept( + std::declval<decltype(l)>()))  
-> decltype(+ std::forward<decltype(l)>(l))  
{
```

# Unary Ops:

```
constexpr inline auto Posate = [](auto&& l) constexpr  
noexcept(noexcept( + std::declval<decltype(l)>()))  
-> decltype(+ std::forward<decltype(l)>(l))  
{  
    return + std::forward<decltype(l)>(l);  
};
```

# Unary Ops:

```
constexpr inline auto Posate = [](auto&& l) constexpr
noexcept(noexcept( + std::declval<decltype(l)>()))
-> decltype(+ std::forward<decltype(l)>(l))
{
    return + std::forward<decltype(l)>(l);
};
```

Ternary Op:

# Ternary Op:

```
constexpr auto Wtf=[](auto&& c, auto&& l, auto&& r) constexpr
```

# Ternary Op:

```
constexpr auto Wtf=[](auto&&c, auto&& l, auto&& r) constexpr  
noexcept(noexcept(std::declval<decltype(c)>()?std::declval<decltype(l)>():std::declval<decltype(r)>()))
```

# Ternary Op:

```
constexpr auto Wtf=[](auto&&c, auto&& l, auto&& r) constexpr  
noexcept(noexcept(std::declval<decltype(c)>()?std::declval<decltype(l)>():std::declval<decltype(r)>()))  
-> decltype(std::declval<decltype(c)>()?std::declval<decltype(l)>():std::declval<decltype(r)>())
```



# Ternary Op:

```
constexpr auto Wtf=[](auto&&c, auto&& l, auto&& r) constexpr  
noexcept(noexcept(std::declval<decltype(c)>()?std::declval<decltype(l)>():std::declval<decltype(r)>()))  
-> decltype(std::declval<decltype(c)>()?std::declval<decltype(l)>():std::declval<decltype(r)>())  
{
```

# Ternary Op:

```
constexpr auto Wtf=[](auto&& c, auto&& l, auto&& r) constexpr  
noexcept(noexcept(std::declval<decltype(c)>()?std::declval<decltype(l)>():std::declval<decltype(r)>()))  
-> decltype(std::declval<decltype(c)>()?std::declval<decltype(l)>():std::declval<decltype(r)>())  
{  
    return std::forward<decltype(c)>(c) ?
```

# Ternary Op:

```
constexpr auto Wtf=[](auto&&c, auto&& l, auto&& r) constexpr
noexcept(noexcept(std::declval<decltype(c)>()?std::declval<decltype(l)>():std::declval<decltype(r)>()))
-> decltype(std::declval<decltype(c)>()?std::declval<decltype(l)>():std::declval<decltype(r)>())
{
    return std::forward<decltype(c)>(c) ?
        std::forward<decltype(l)>(l) : std::forward<decltype(r)>(r);
}
```

# Ternary Op:

```
constexpr auto Wtf=[](auto&&c, auto&& l, auto&& r) constexpr
noexcept(noexcept(std::declval<decltype(c)>()?std::declval<decltype(l)>():std::declval<decltype(r)>()))
-> decltype(std::declval<decltype(c)>()?std::declval<decltype(l)>():std::declval<decltype(r)>())
{
    return std::forward<decltype(c)>(c) ?
           std::forward<decltype(l)>(l) : std::forward<decltype(r)>(r);
};
```

# Ternary Op:

```
constexpr auto Wtf=[](auto&&c, auto&& l, auto&& r) constexpr
noexcept(noexcept(std::declval<decltype(c)>()?std::declval<decltype(l)>():std::declval<decltype(r)>()))
-> decltype(std::declval<decltype(c)>()?std::declval<decltype(l)>():std::declval<decltype(r)>())
{
    return std::forward<decltype(c)>(c) ?
           std::forward<decltype(l)>(l) : std::forward<decltype(r)>(r);
};
```

Sum up

**Function Objects  
can/should be**

**Obsoleted by Lambdas**

The image features a pair of rich red theater curtains with vertical pleats and gold tassels on the sides. The curtains are drawn back, revealing a dark background. Centered on the curtains is the text "SLASH OR DASH" in a bold, white, sans-serif font, arranged in three lines.

**SLASH  
OR  
DASH**

**NET**



**/// SLASH ///**

**NETST**

**— — — DASH — — —**

**NETSTAT**

**— — — DASH — — —**

**Peter Sommerlad - FOOL**

**Michel Grootjans - Crafting Guitars**

**Rob Smallshire - The Gender Equality Paradox**

**Florian Gilcher - Trains**

**Graham Haynes - On Automati**

**Marshall Clow - Fuzzing Your Code**

**Chris Oldwood - The Far Side**

**Jon Kalb - This is Why We Can't Have Nice Things**

**Phil Nash - East All The Things**

**Jim Hague - A Brief of one-line abuses**

**Mike Seymour - Sparsity Parsery**

# The craft of building guitars

A luthier with glasses and a mustache is focused on his work in a workshop. He is using a hand plane to smooth a piece of wood on a workbench. A bright lamp illuminates his workspace. In the foreground, a finished acoustic guitar is visible on a stand. The background is filled with various tools and materials, creating a sense of a busy, traditional craft environment.

Random musings by @michelgrootjans























So... what is the difference between  
**work** and **craft**?

Care, attention to detail, functionality, durability



Continuous education, sharing, accepting criticism



# Experimenting





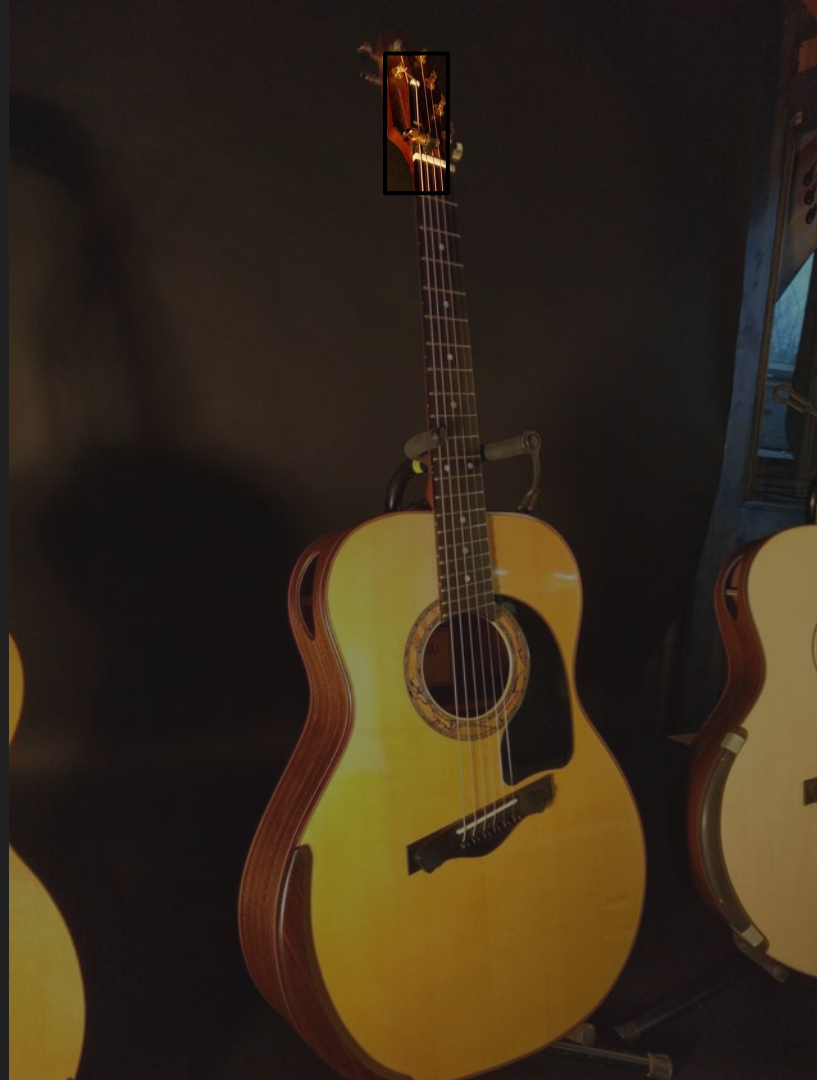














**Should every guitar be finely crafted?**

**Should every ~~guitar~~ <sup>codebase</sup> be finely crafted?**

Lush Tshirts  
www.lushshirts.co.uk

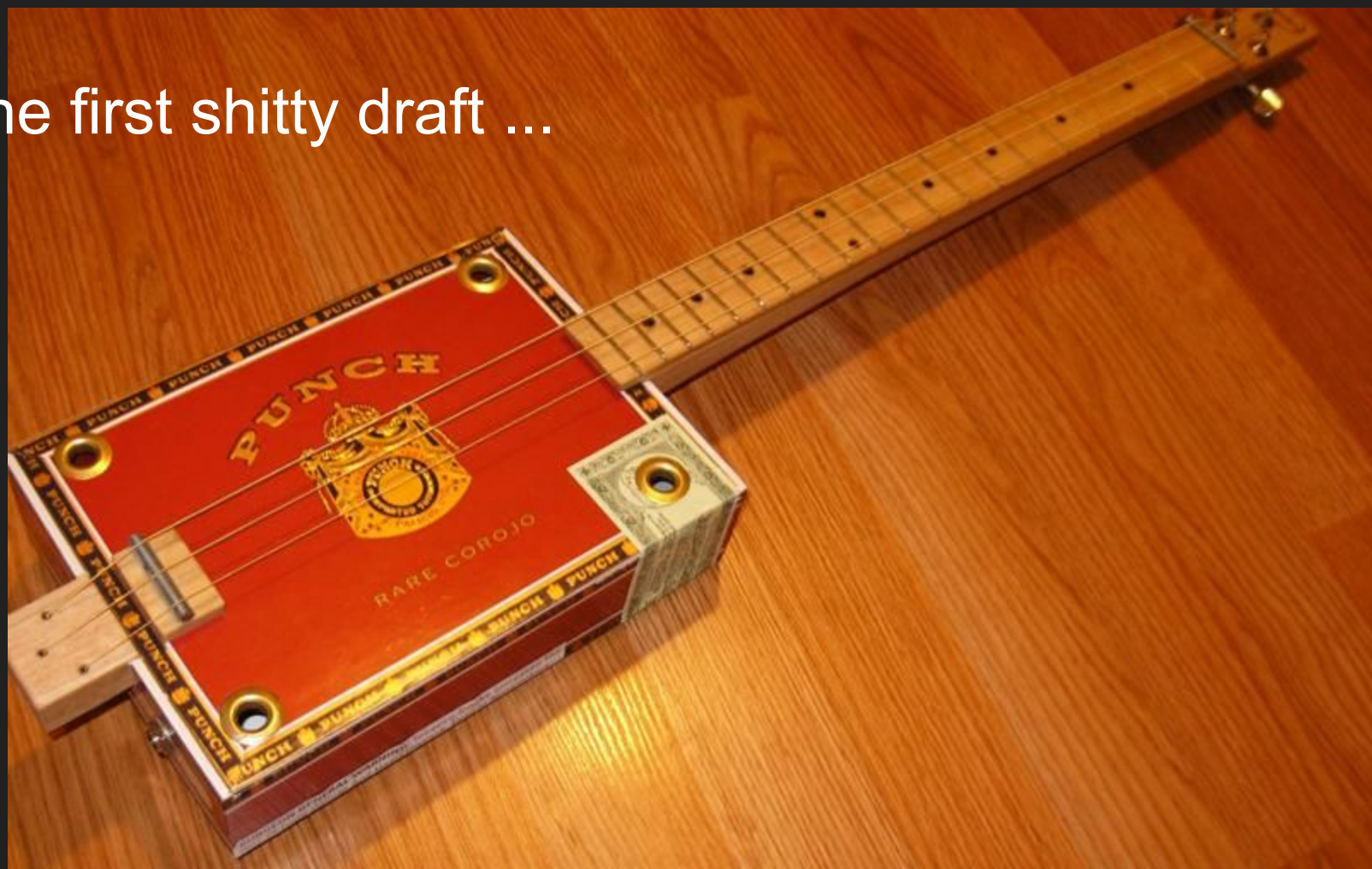
~~SINGLE~~ yes

~~TAKEN~~ no

**DEPENDS**  
**ON WHO'S ASKING**

It depends on what you plan to do with it

The first shitty draft ...





# The throw-away guitar





The image features a pair of rich red theater curtains with a heavy, draped top edge. The curtains are pulled back slightly on both sides, revealing gold tassels. The background behind the curtains is a dark, almost black color, which makes the red curtains stand out. Centered on the curtains is the text "SLASH OR DASH" in a large, bold, white, sans-serif font. The text is arranged in three lines: "SLASH" on the top line, "OR" on the middle line, and "DASH" on the bottom line.

**SLASH  
OR  
DASH**

**ATTRIB**

**/// BOTH ---**

**Peter Sommerlad - FOOL**

**Michel Grootjans - Crafting Guitars**

**Rob Smallshire - The Gender Equality Paradox**

**Florian Gilcher - Trains**

**Graham Haynes - On Automati**

**Marshall Clow - Fuzzing Your Code**

**Chris Oldwood - The Far Side**

**Jon Kalb - This is Why We Can't Have Nice Things**

**Phil Nash - East All The Things**

**Jim Hague - A Brief of one-line abuses**

**Mike Seymour - Sparsity Parsery**

# The Gender Equality Paradox

**Robert Smallshire**

 @robsmallshire

# The Gender-Equality Paradox in Science, Technology, Engineering, and Mathematics Education



**Gijsbert Stoet<sup>1</sup>**  and **David C. Geary<sup>2</sup>**

<sup>1</sup>School of Social Sciences, Leeds Beckett University, and <sup>2</sup>Department of Psychological Sciences, University of Missouri

Psychological Science

1–13

© The Author(s) 2018

Reprints and permissions:

[sagepub.com/journalsPermissions.nav](http://sagepub.com/journalsPermissions.nav)

DOI: 10.1177/0956797617741719

[www.psychologicalscience.org/PS](http://www.psychologicalscience.org/PS)

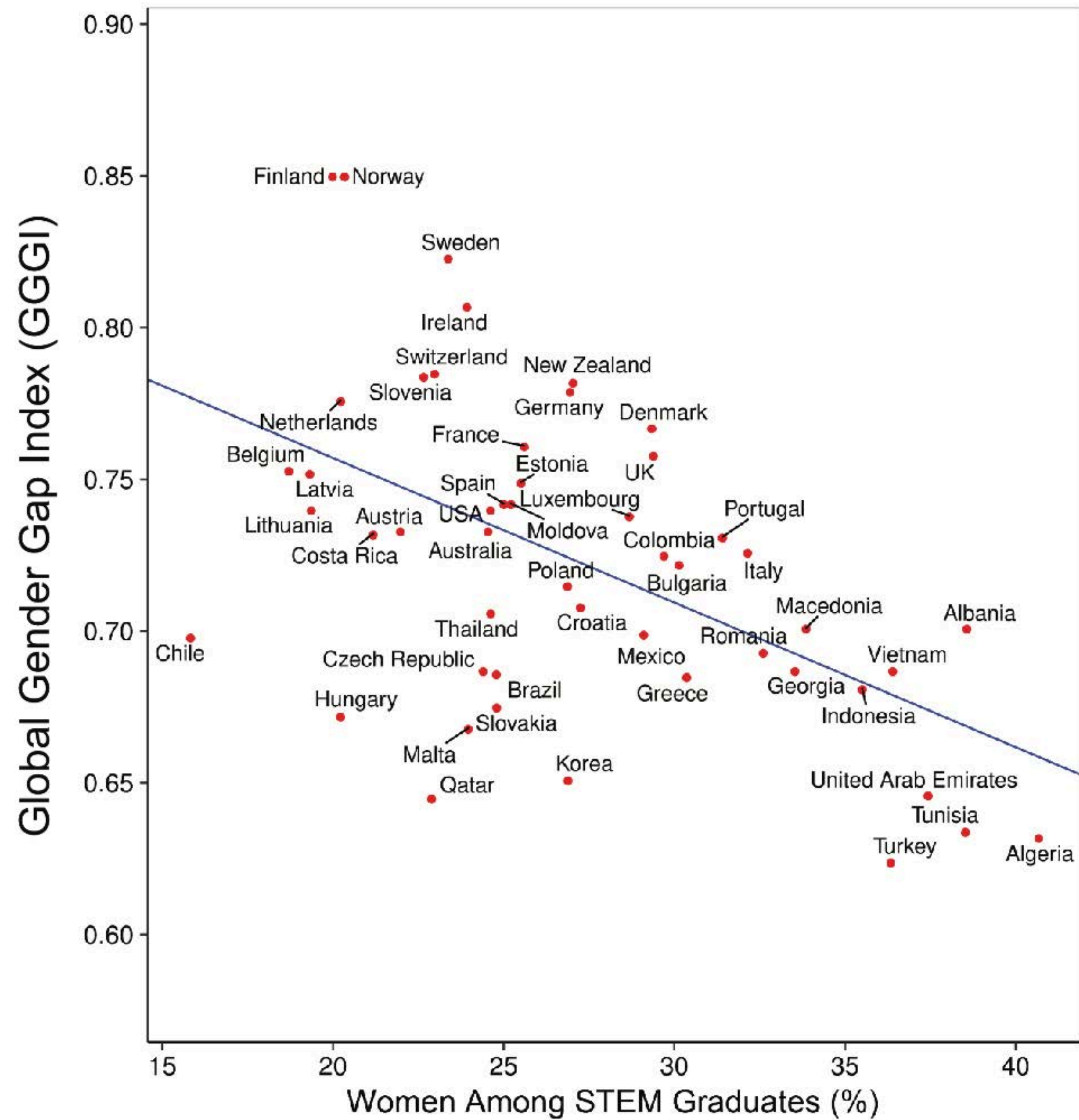


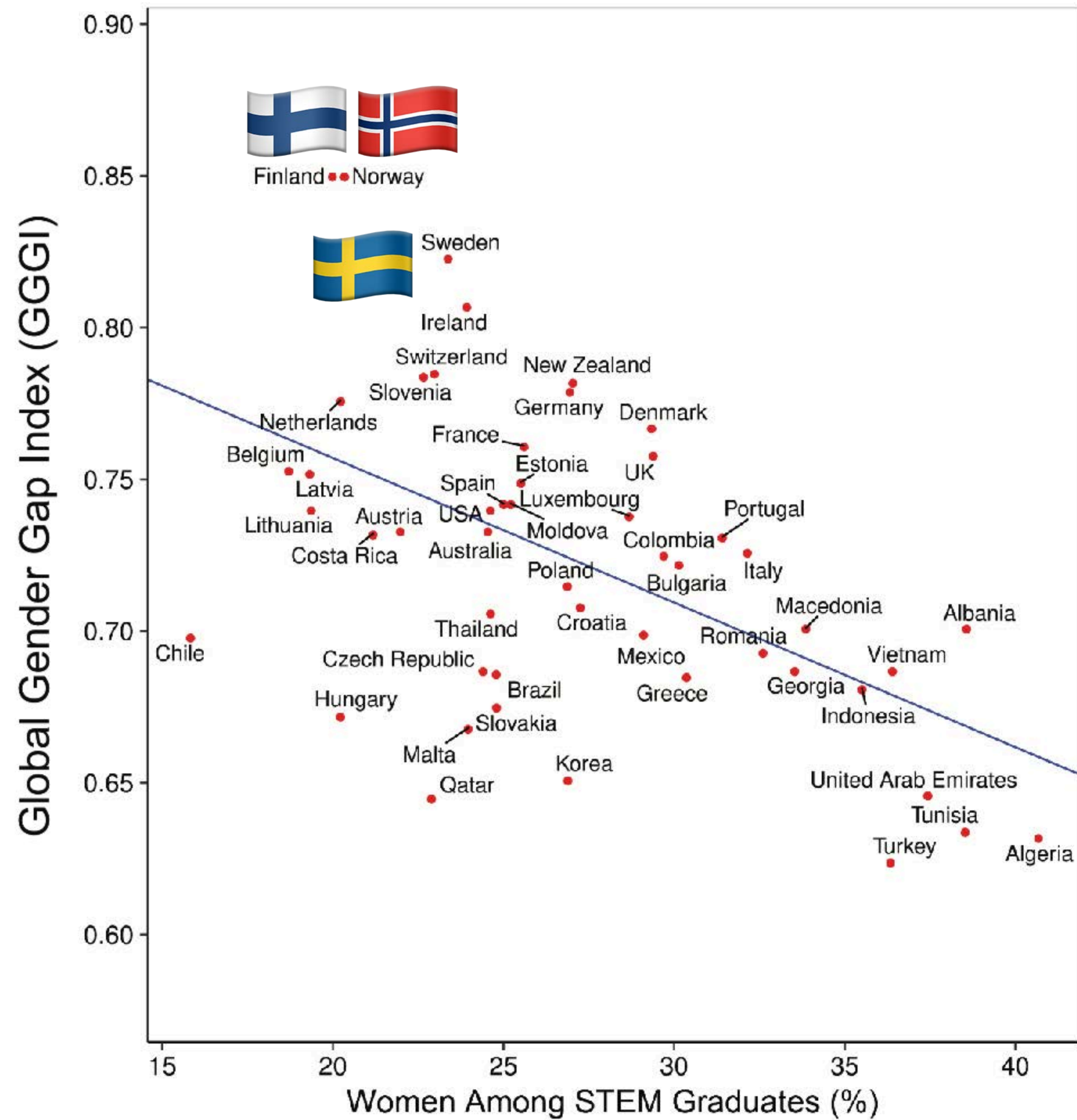
## Abstract

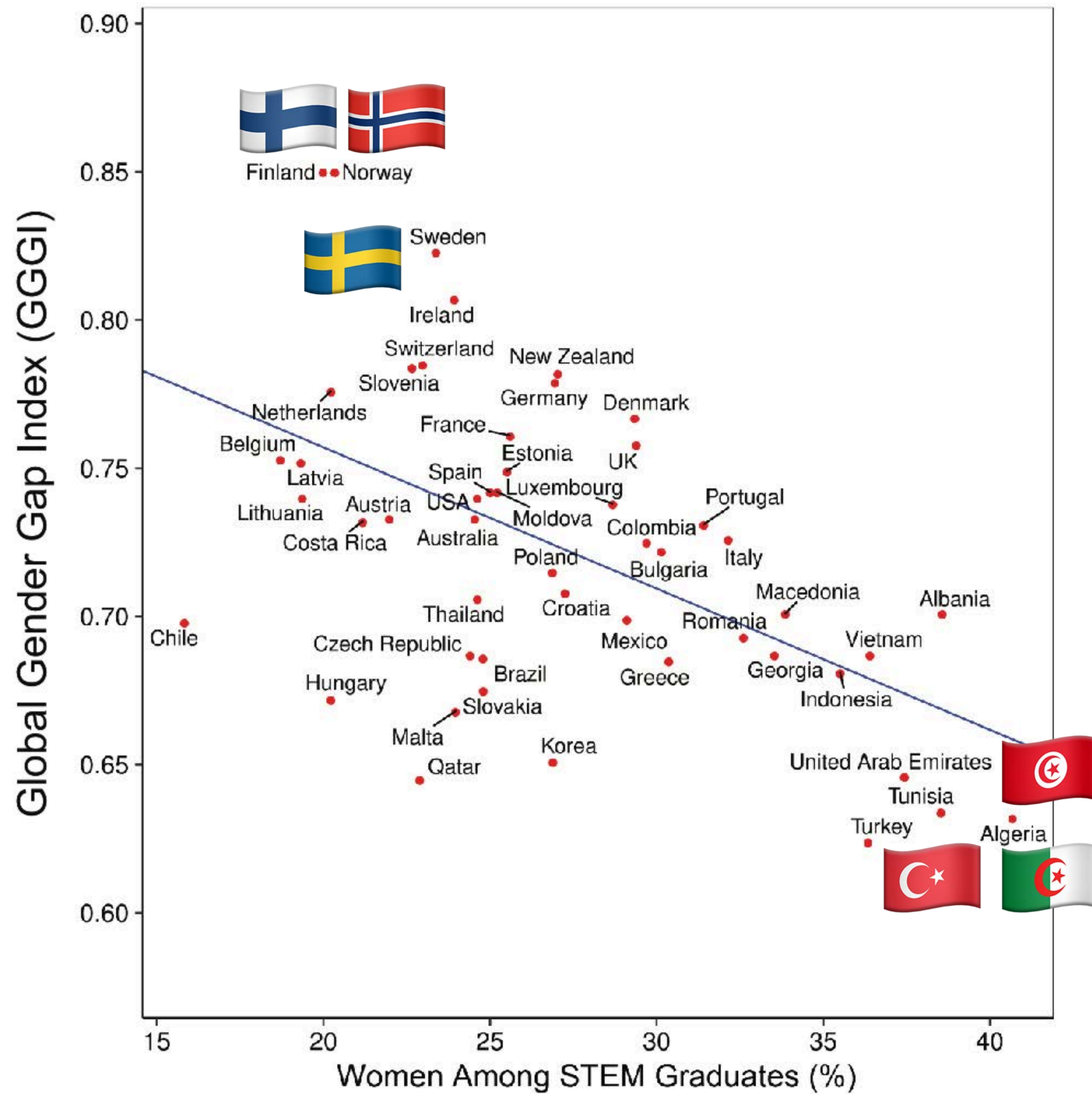
The underrepresentation of girls and women in science, technology, engineering, and mathematics (STEM) fields is a continual concern for social scientists and policymakers. Using an international database on adolescent achievement in science, mathematics, and reading ( $N = 472,242$ ), we showed that girls performed similarly to or better than boys in science in two of every three countries, and in nearly all countries, more girls appeared capable of college-level STEM study than had enrolled. Paradoxically, the sex differences in the magnitude of relative academic strengths and pursuit of STEM degrees rose with increases in national gender equality. The gap between boys' science achievement and girls' reading achievement relative to their mean academic performance was near universal. These sex differences in academic strengths and attitudes toward science correlated with the STEM graduation gap. A mediation analysis suggested that life-quality pressures in less gender-equal countries promote girls' and women's engagement with STEM subjects.

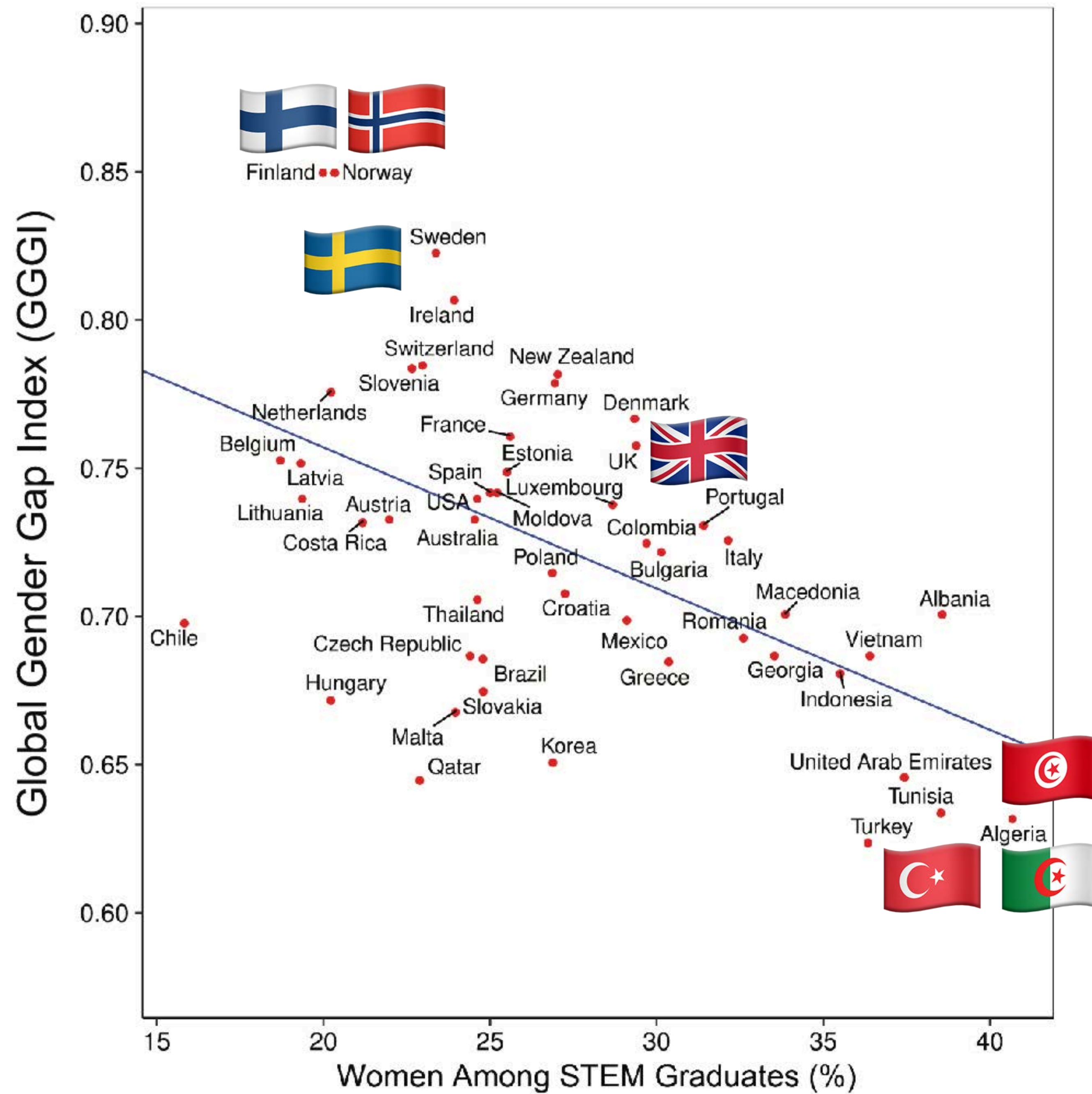


***N* = 472 242**









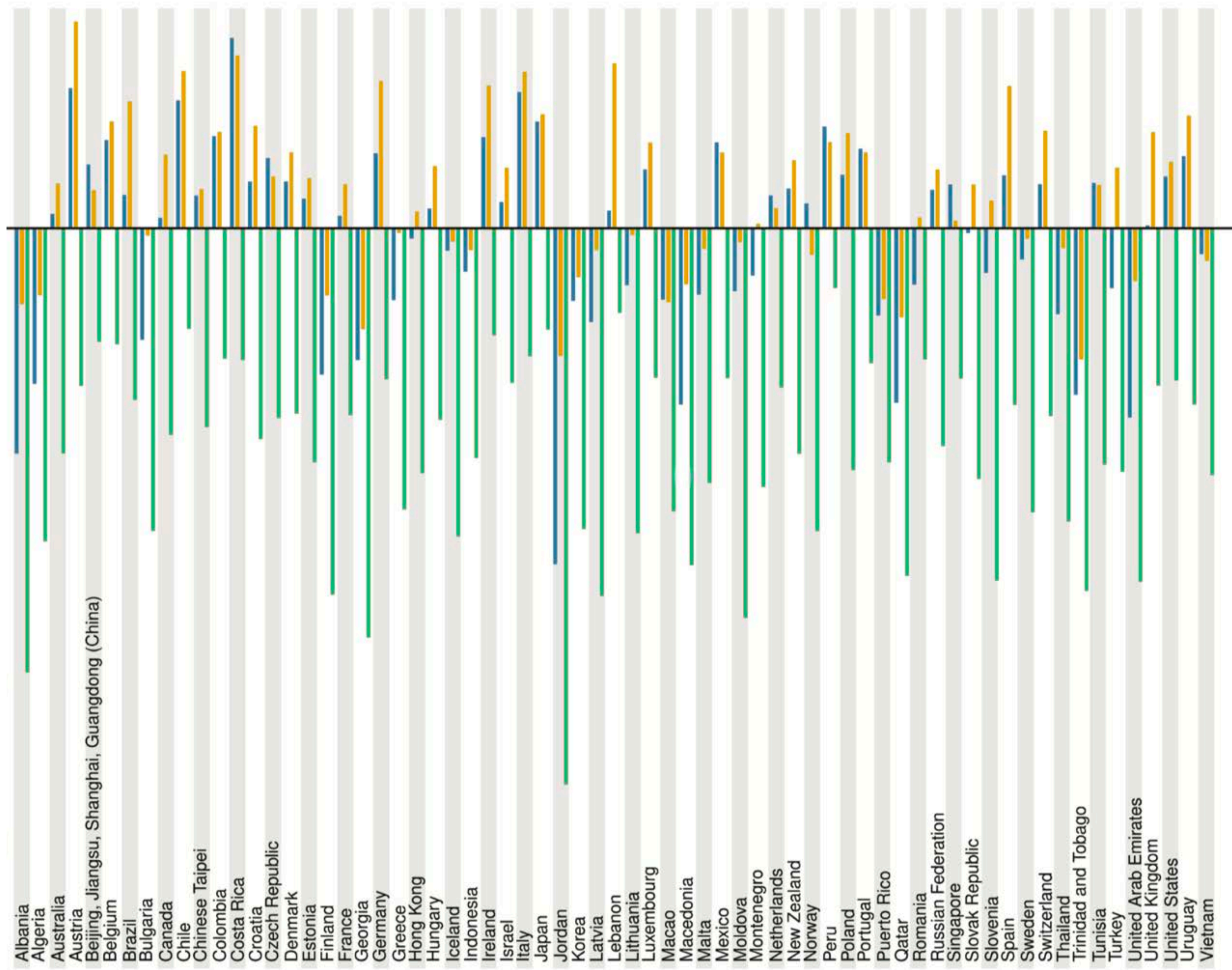
**boys  
better**

**girls  
better**

**reading**

**maths**

**science**



# intra-individual achievement



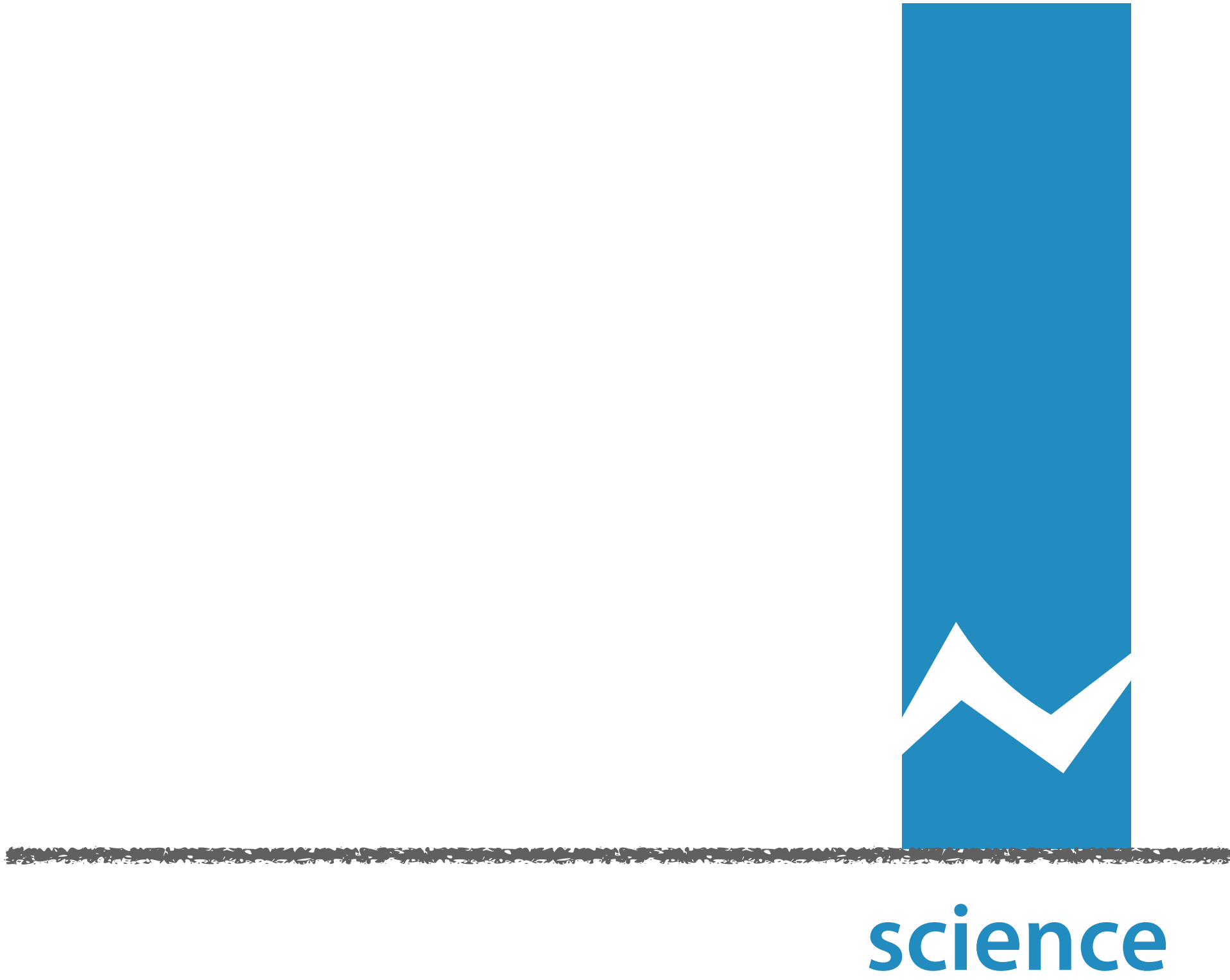
**Girls**



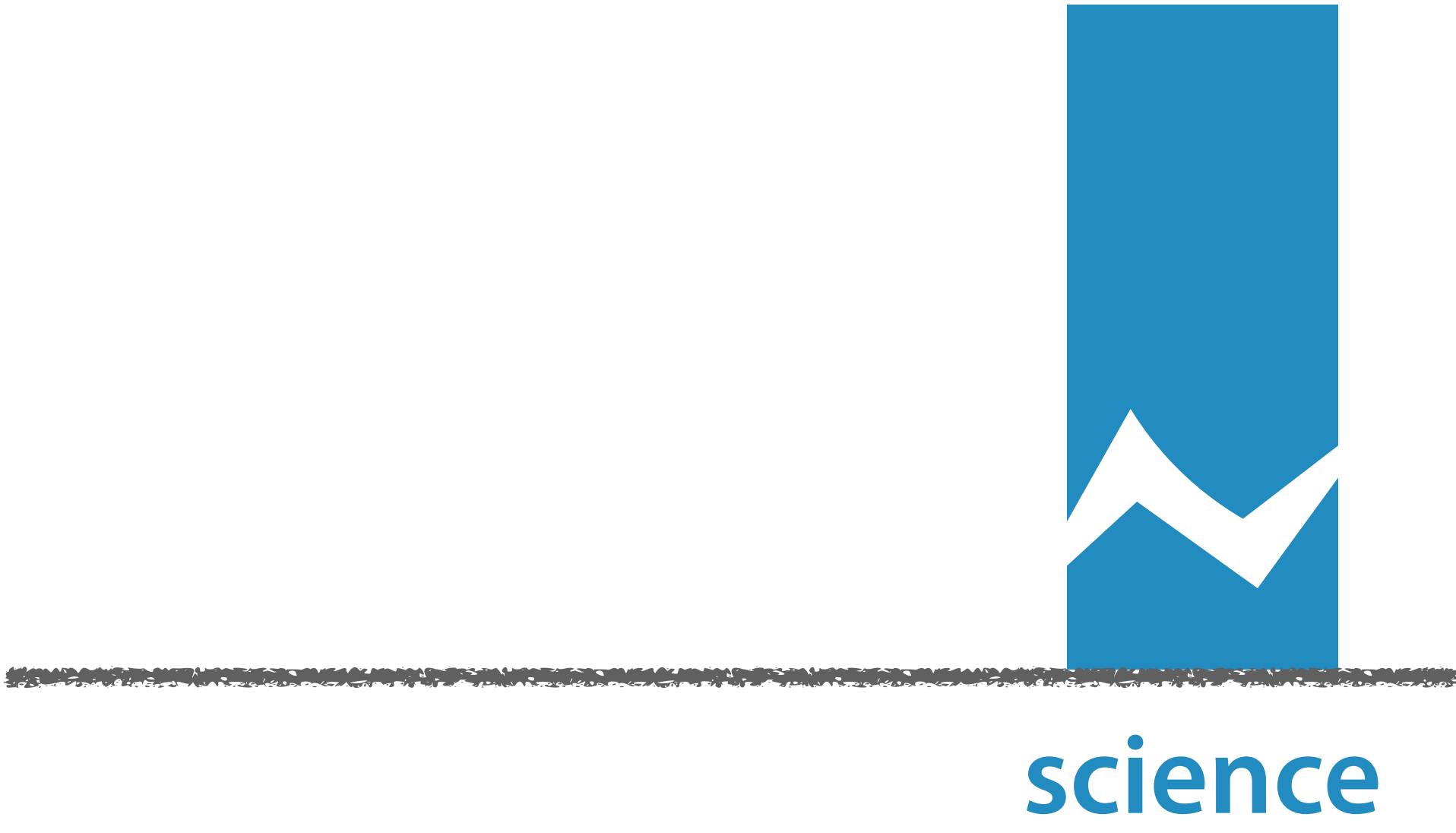
**science**

**Boys**

# intra-individual achievement



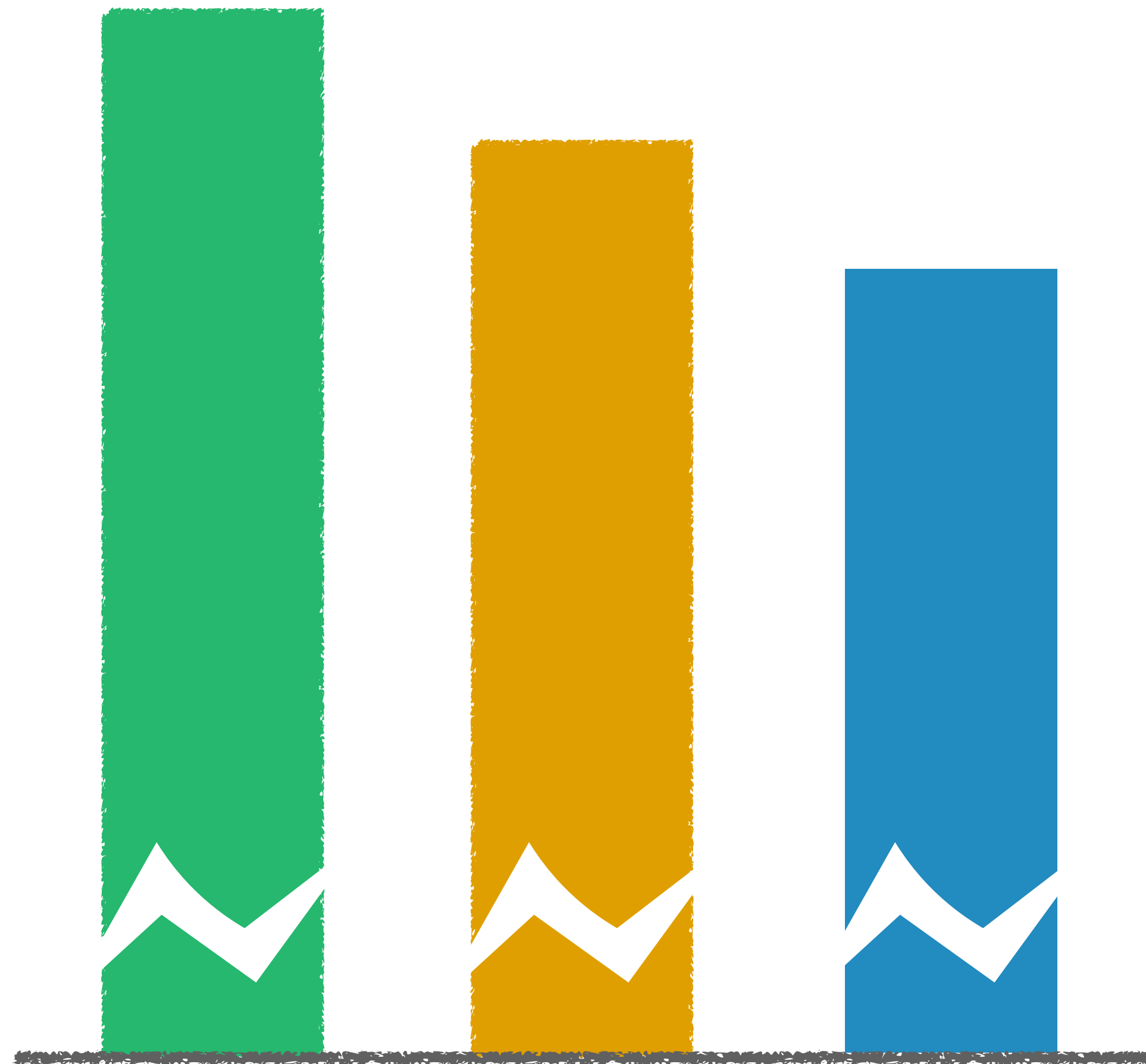
**Girls**



**Boys**



# intra-individual achievement

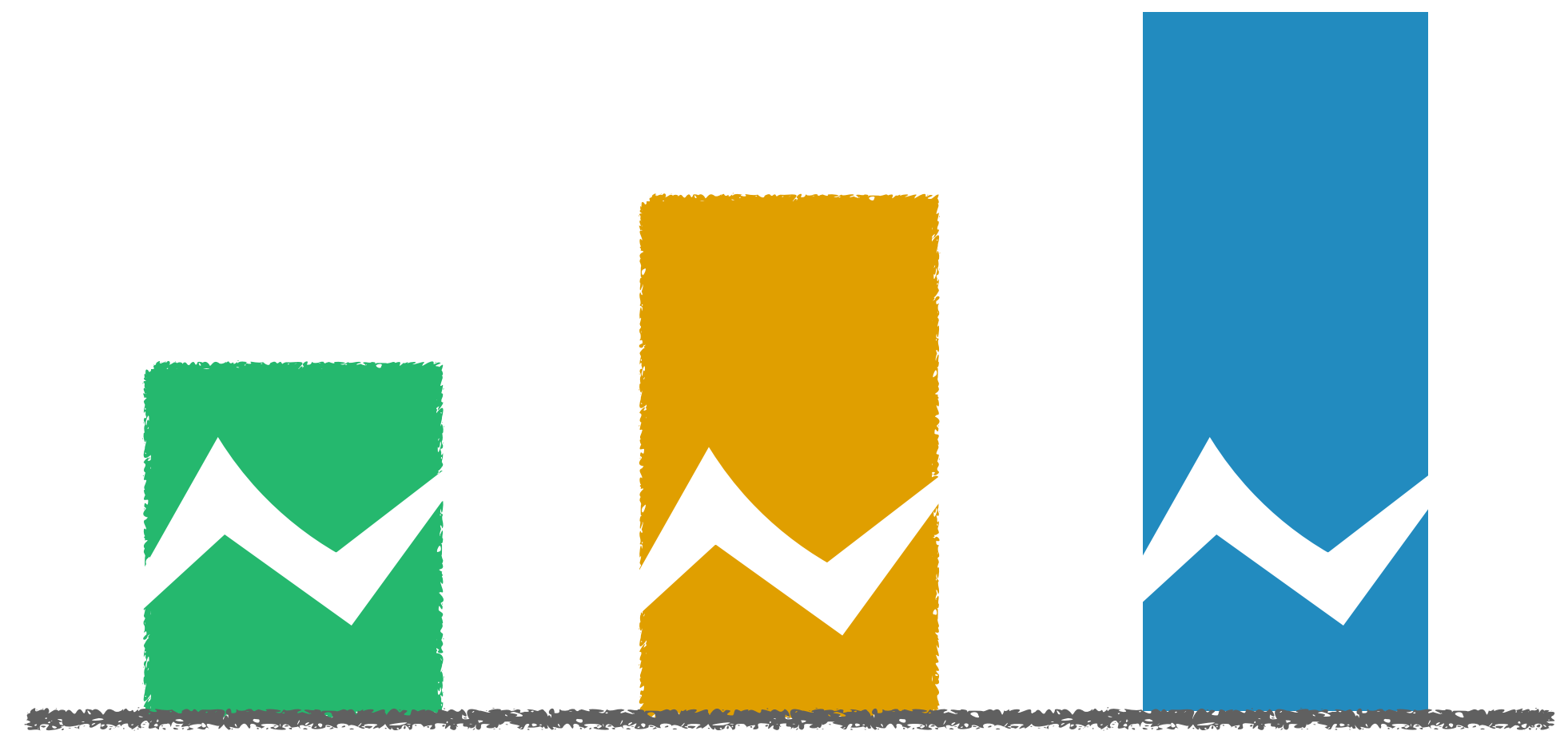


reading

maths

science

**Girls**



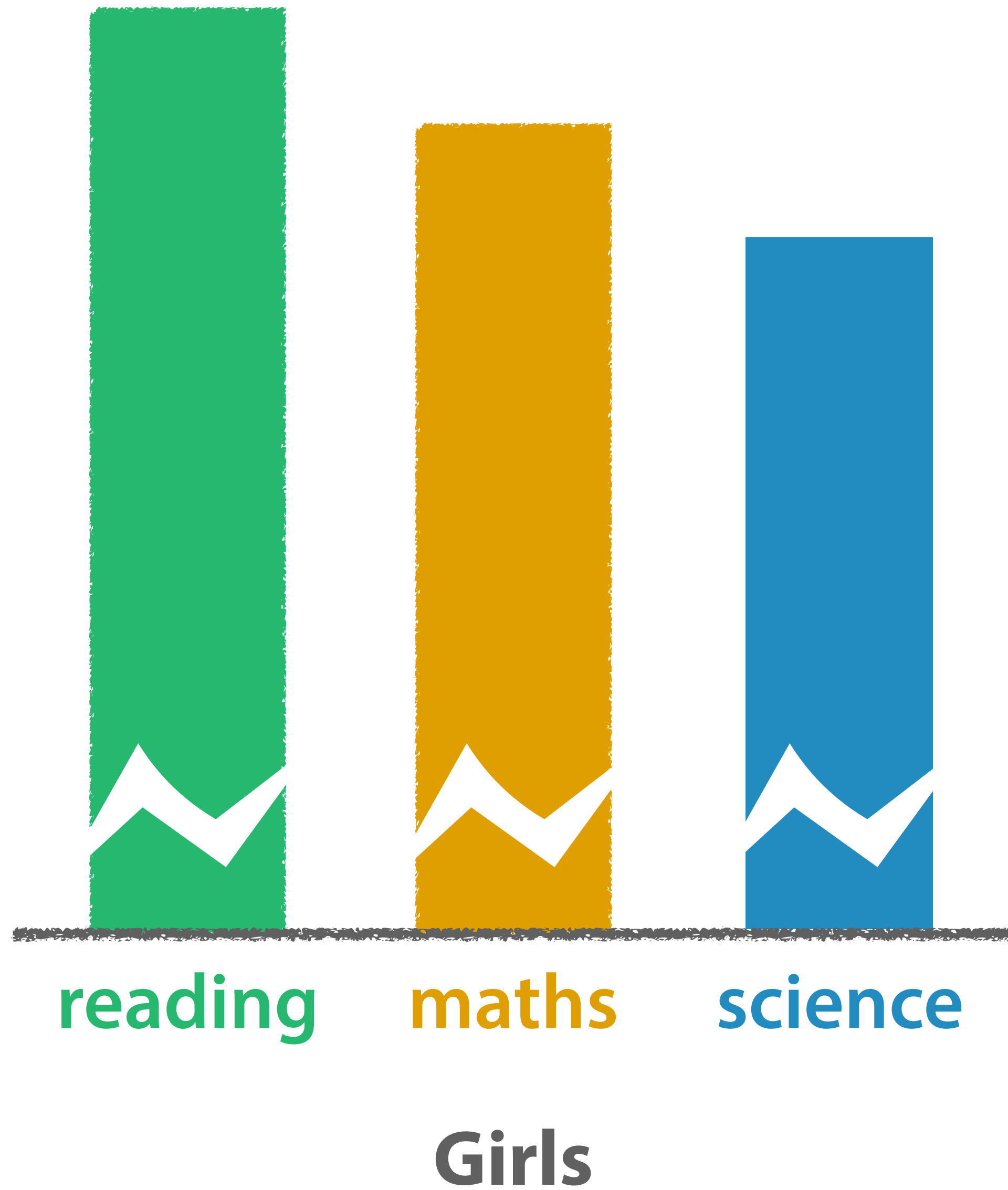
reading

maths

science

**Boys**

# intra-individual achievement



high social security  
high gender equality

**economic freedom  
to follow interests**

low social security  
low gender equality

**high economic  
premium for STEM  
careers**

The image features a pair of rich red theater curtains with a heavy, draped top edge. The curtains are pulled back on both sides, revealing gold tassels. The background behind the curtains is a dark, almost black, stage floor. Centered on the curtains is the text "SLASH OR DASH" in a large, white, bold, sans-serif font, arranged in three lines.

**SLASH  
OR  
DASH**

**RFC 3986**

# UNIFORM RESOURCE IDENTIFIER (URI): GENERIC SYNTAX

**/// SLASH ///**

**Peter Sommerlad - FOOL**

**Michel Grootjans - Crafting Guitars**

**Rob Smallshire - The Gender Equality Paradox**

**Florian Gilcher - Trains**

**Graham Haynes - On Automati**

**Marshall Clow - Fuzzing Your Code**

**Chris Oldwood - The Far Side**

**Jon Kalb - This is Why We Can't Have Nice Things**

**Phil Nash - East All The Things**

**Jim Hague - A Brief of one-line abuses**

**Mike Seymour - Sparsity Parsery**

There's an issue I'd  
like to speak about.

SL(1)



====  
\_D\_| | \_\_\_\_\_/ \\ I I \_\_\_\_\_ === |  
| ( ) — | H \\ \_\_\_\_\_/ | | = | \_\_\_\_\_ |  
/ | | H | | | | | | | |  
| | | H | \_\_\_\_\_ | [ ] |  
| \_\_\_\_\_ | \_\_\_\_\_ H \_\_\_\_\_/ \_\_\_\_\_/ [ ] ~ \\ \_\_\_\_\_ |  
| / | | ——— I \_\_\_\_\_ I [ ] [ ] D | ===== | \_\_\_\_\_  
\_\_\_\_\_/ = | o | = - ~ ~ \\ / ~ ~ \\ / ~ ~ \\ / ~ ~ \\ \_\_\_\_\_ Y \_\_\_\_\_  
| / - = | \_\_\_\_\_ | = || || || | \_\_\_\_\_/ ~ \\ \_\_\_\_\_/  
\\ \\_ / \\ \\_ O ===== O ===== O ===== O \\_ / \\ \\_ /

$sl(1)$  is in danger

appreciation is sinking

distributions ship it with options  
allowing to stop the train (debian)

in outdated versions (debian)

I'm pretty sure someone is  
rebuilding it as a systemd module.

Outstanding bugs are left unfixed!

```
$ man sl
```

```
BUGS
```

It rarely shows contents of current directory.



A revival!

Rebuild your own  $sl(1)$ , as close to the original as possible, in your favourite language, system, whatever.

<https://github.com/mtoyoda/sl>

Tweet to: @argorak

I've got another issue to talk about!

# Error messages

```
$ rustc -version
```

```
Rust C++ Linter, version 1.25
```

```
fn main() {  
    let mut vec = vec![1,2,3];  
  
    let foo = &vec[2];  
  
    // copious amount of work  
  
    vec[1] = 2;  
}
```



error[E0502]: cannot borrow `vec` as mutable because it is also borrowed as immutable

→ src/main.rs:8:5

```
|  
4 |     let foo = &vec[2];  
    |                — immutable borrow occurs here  
...  
8 |     vec[1] = 2;  
    |     ^^^ mutable borrow occurs here  
9 | }  
    | - immutable borrow ends here
```

Great for readability,  
bad for colleagues.

```
$ rustc -error-format verbal
```

Rust prides itself in bringing ideas  
good ideas from the past to use.

This one goes hundreds of years back!

## Format:

- 2 lines for context
- 2 lines for explaining the problem
- a call to action (my designer told me to!)

```
$ rustc -error-format verbal vec.rs
```

That ``vec`` which you wanted to borrow

Is giving my checker much sorrow

for meanwhile you mutate

it in line eight

let's fix it until tomorrow!

Finally! Compilers for humans!



Also: Compilers for the IoT world!

```
echo "Alexa, `rustc -error-format verbal vec.rs`" | say
```

[rust-lang/rfcs/pull/2378](https://github.com/rust-lang/rfcs/pull/2378)

The image features a pair of rich red theater curtains with vertical pleats and gold tassels on the sides. The curtains are drawn back, revealing a dark background. Centered on the curtains is the text "SLASH OR DASH" in a bold, white, sans-serif font, arranged in three lines.

**SLASH  
OR  
DASH**

**HTML**

**/// SLASH ///**

**(OR NO SLASH)**

**Peter Sommerlad - FOOL**

**Michel Grootjans - Crafting Guitars**

**Rob Smallshire - The Gender Equality Paradox**

**Florian Gilcher - Trains**

**Graham Haynes - On Automati**

**Marshall Clow - Fuzzing Your Code**

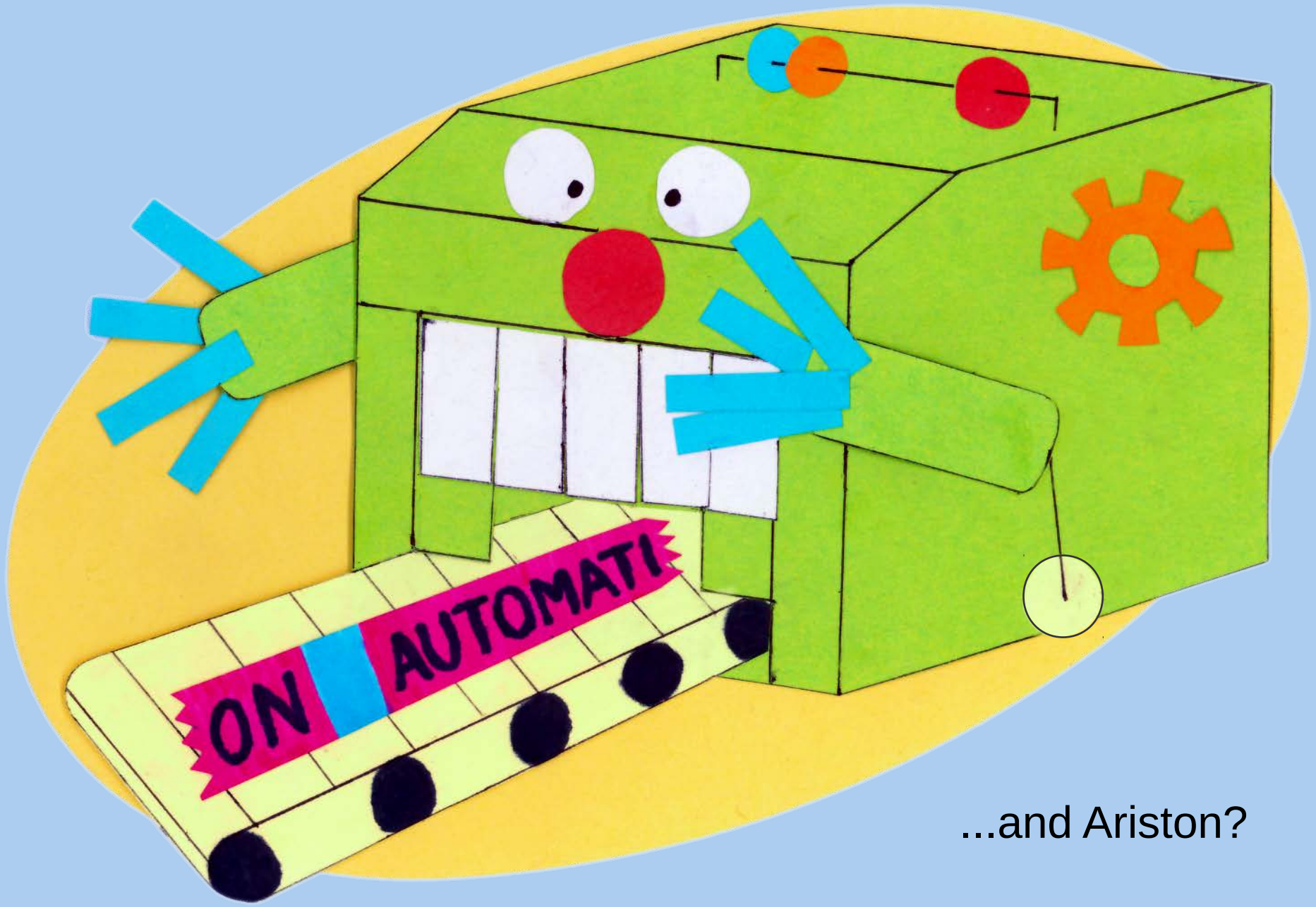
**Chris Oldwood - The Far Side**

**Jon Kalb - This is Why We Can't Have Nice Things**

**Phil Nash - East All The Things**

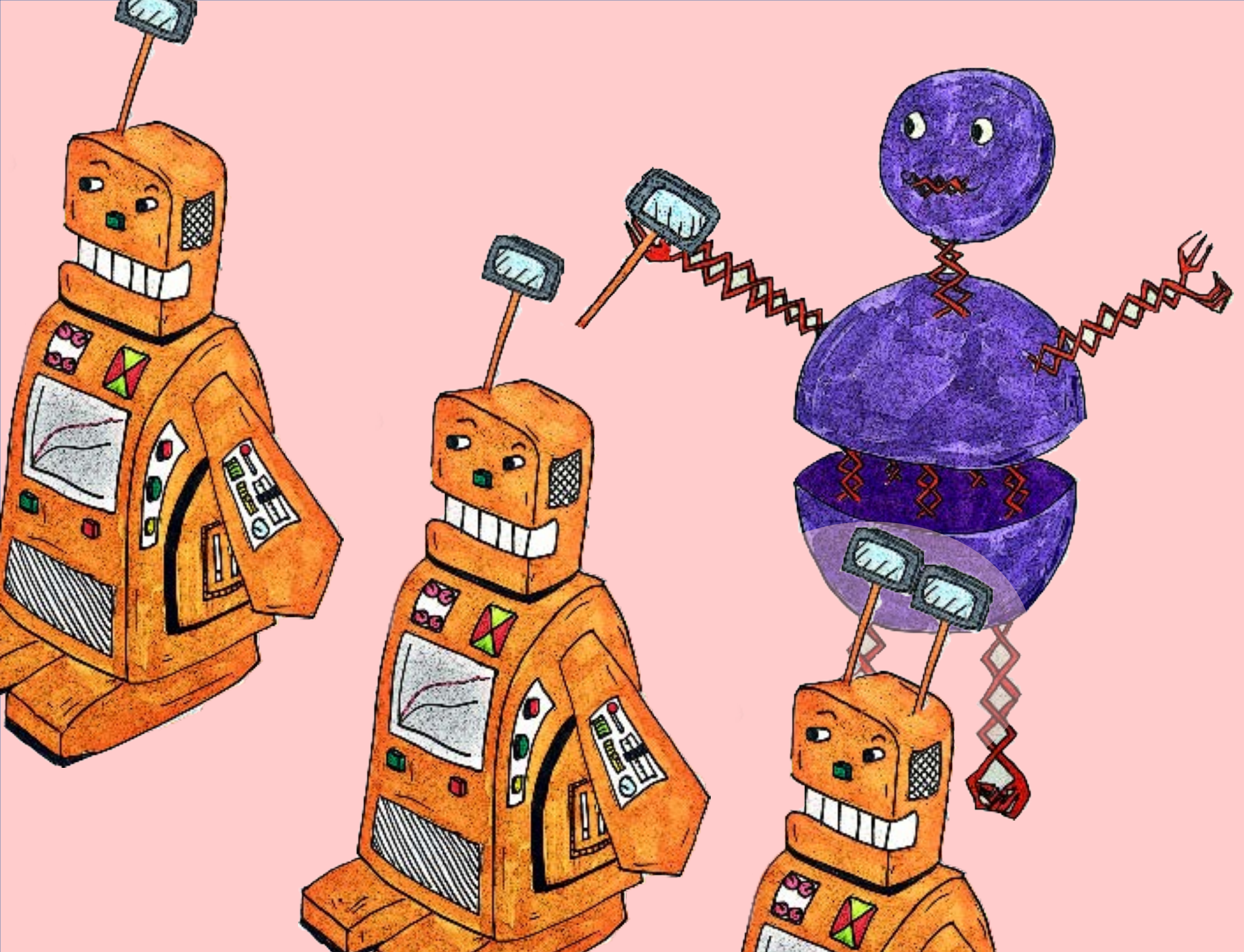
**Jim Hague - A Brief of one-line abuses**

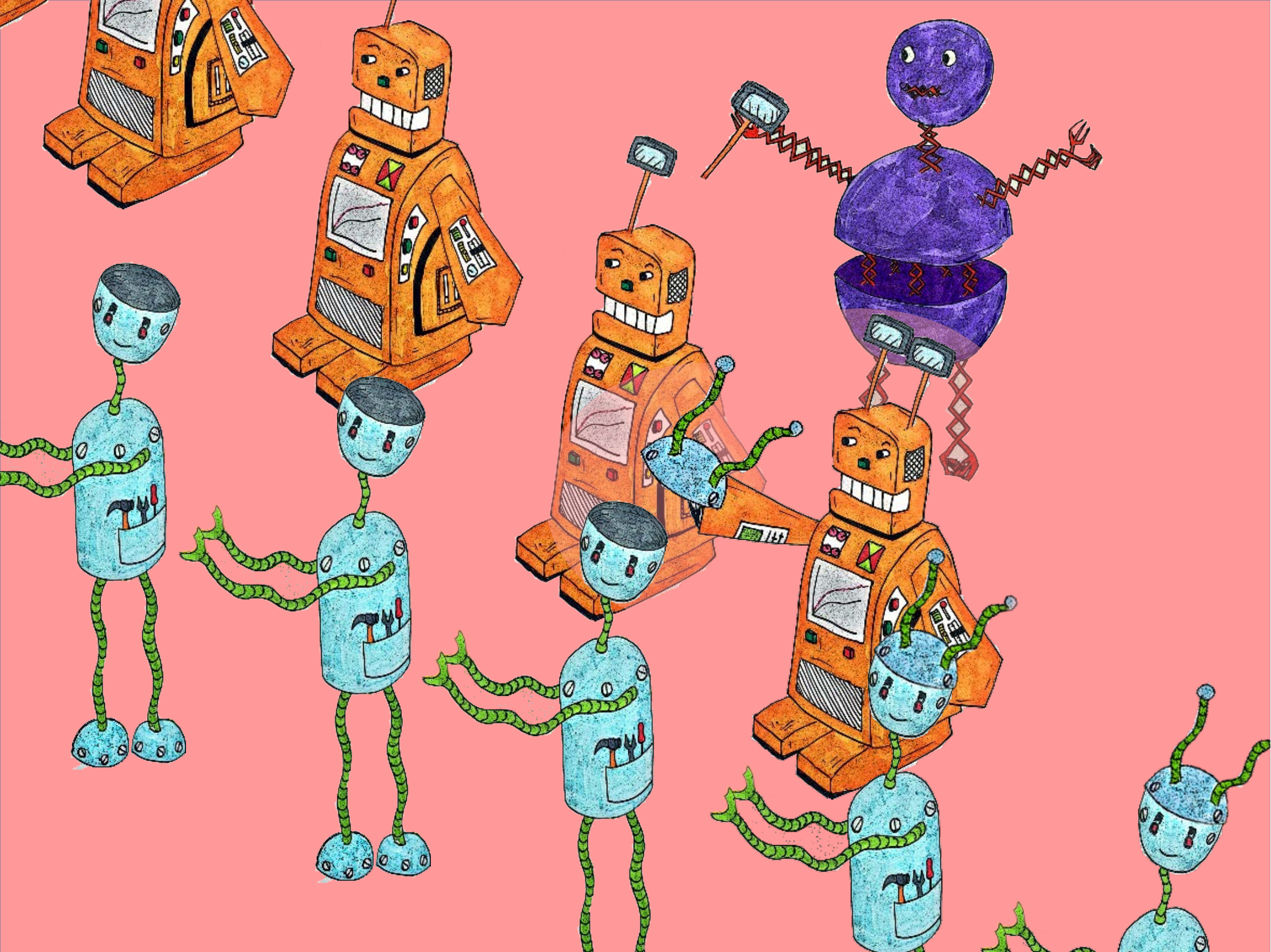
**Mike Seymour - Sparsity Parsery**



...and Ariston?

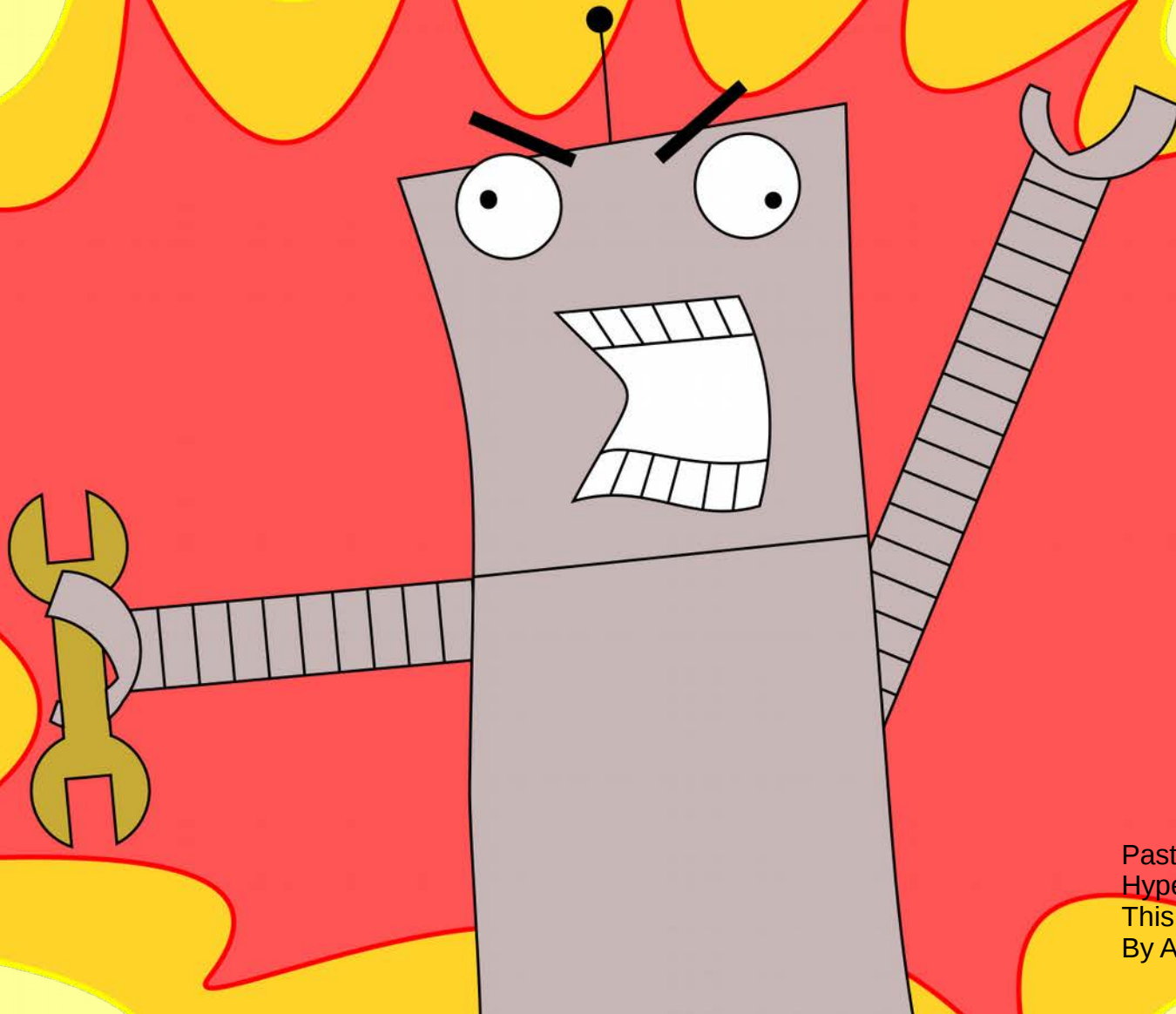






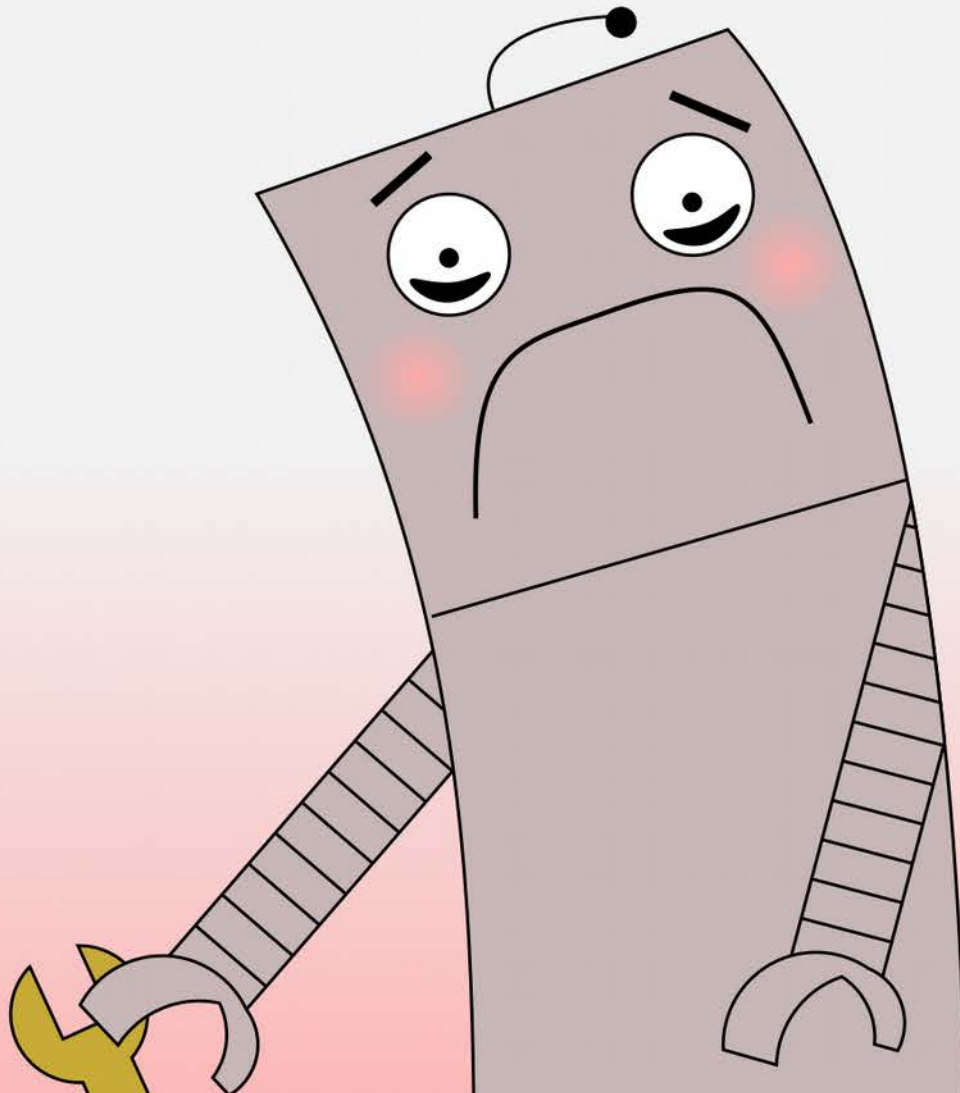


# Automate all the things!



Pastiche of  
Hyperbole and a Half  
This is Why I'll Never be an Adult  
By Allie Brosh

# Automate all the things?



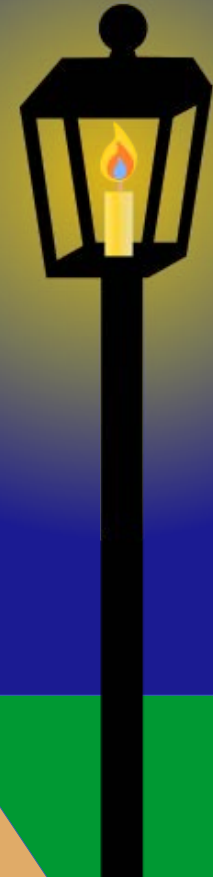
Pastiche of  
Hyperbole and a Half  
This is Why I'll Never be an Adult  
By Allie Brosh

IKEA

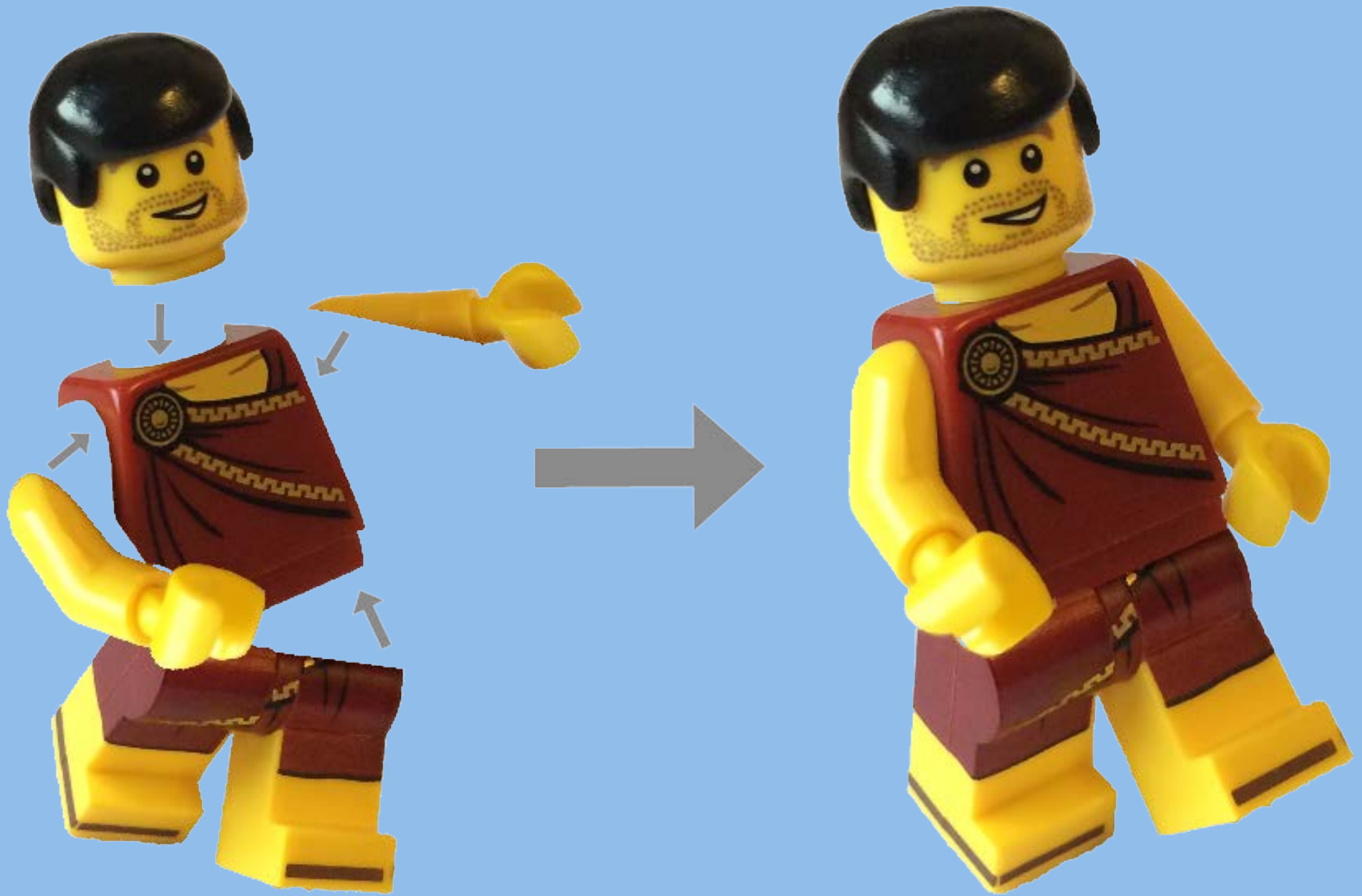


FLAT PACK DESK

Instructions



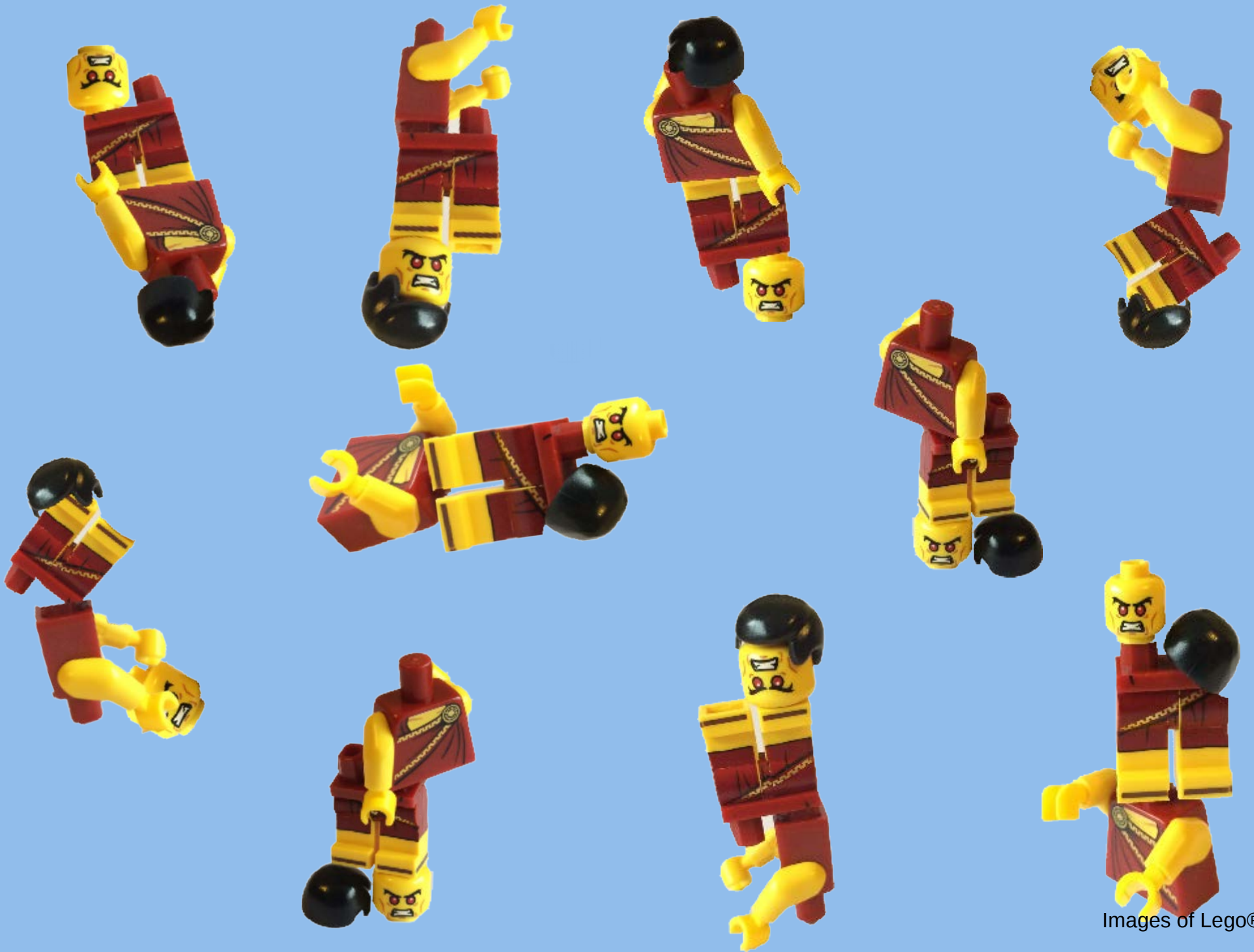




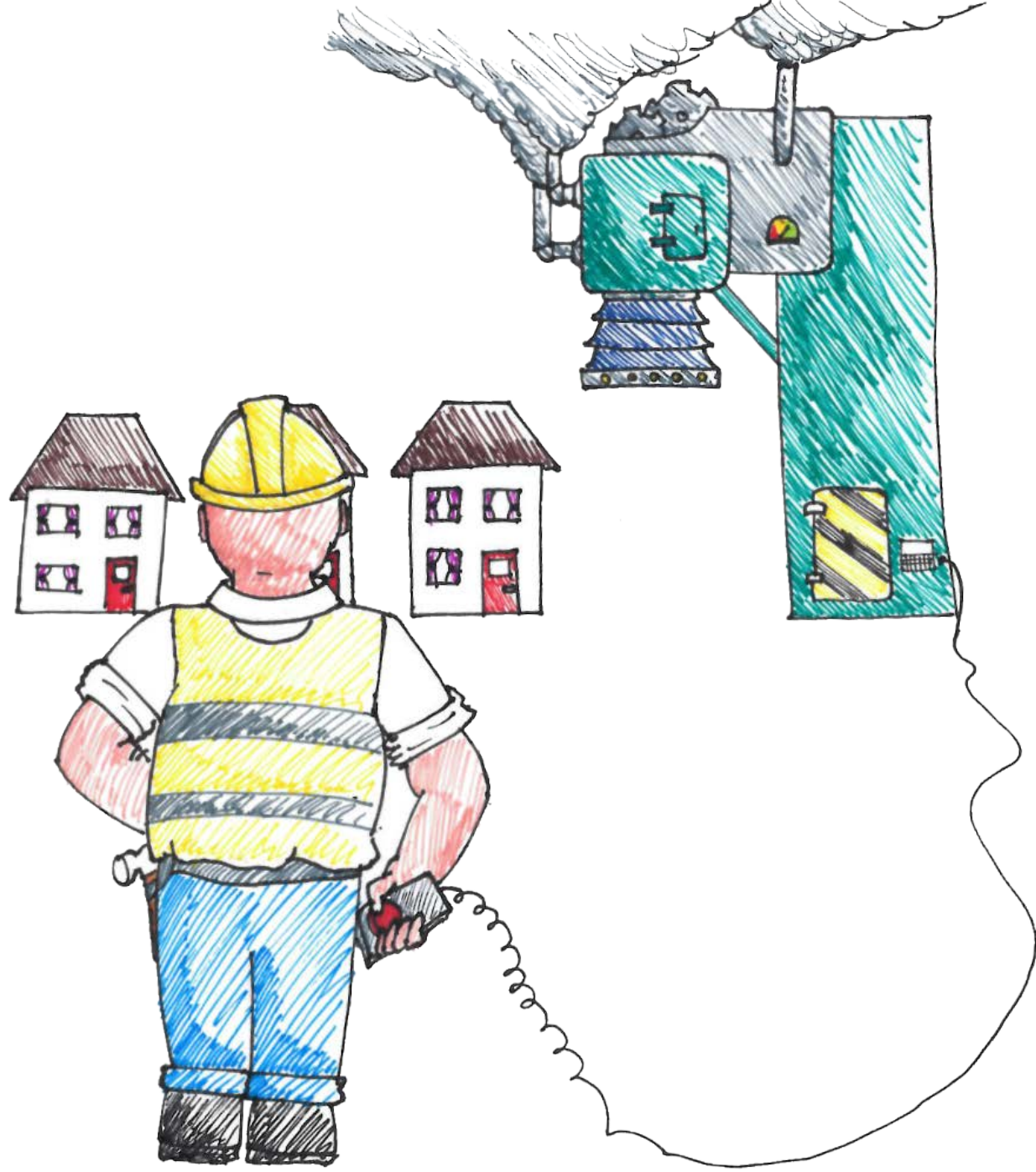


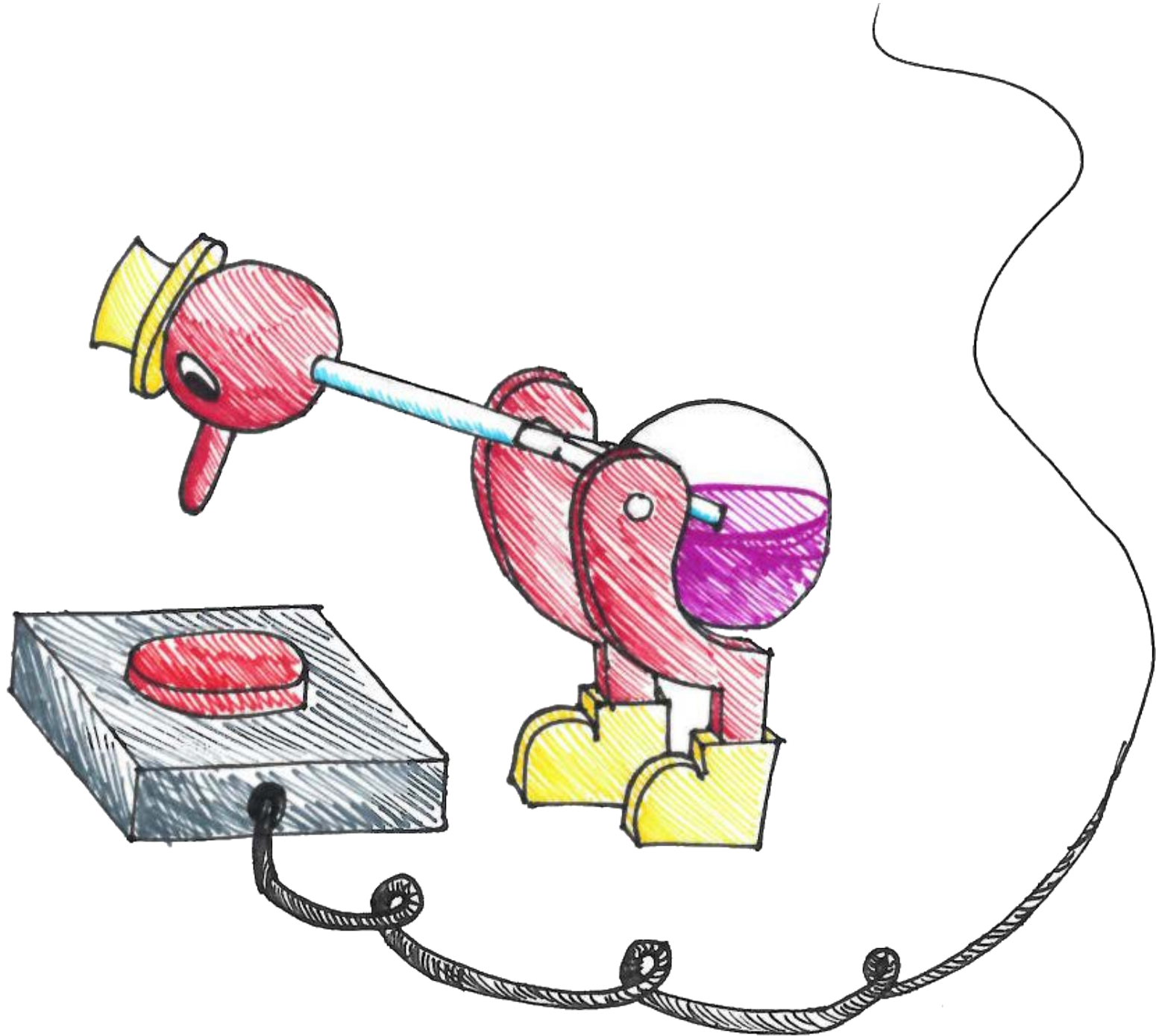








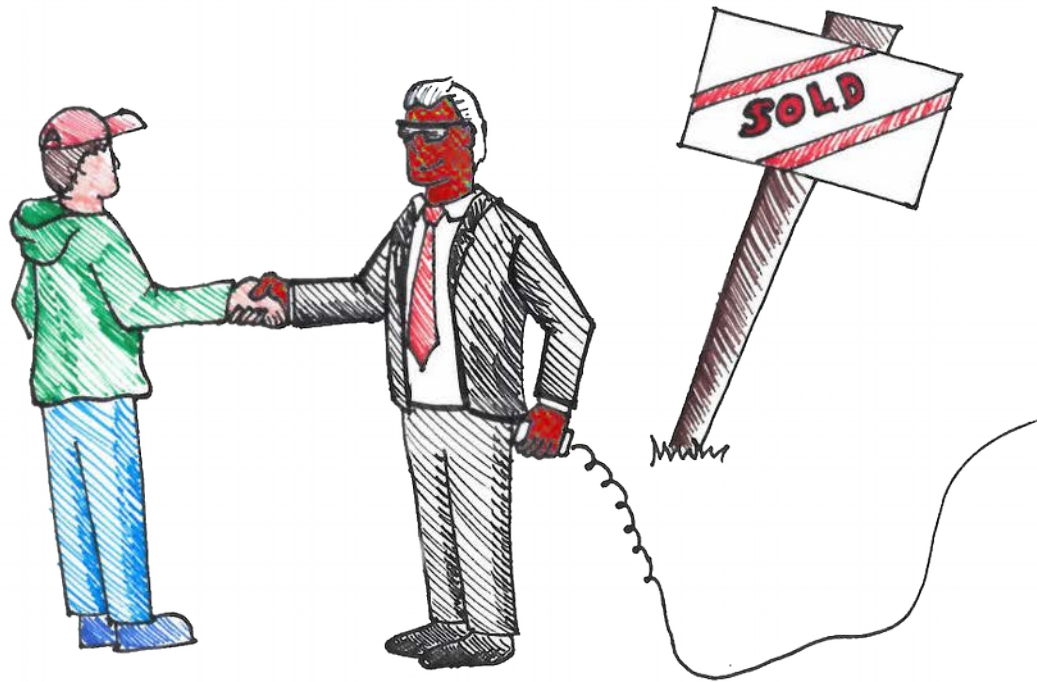


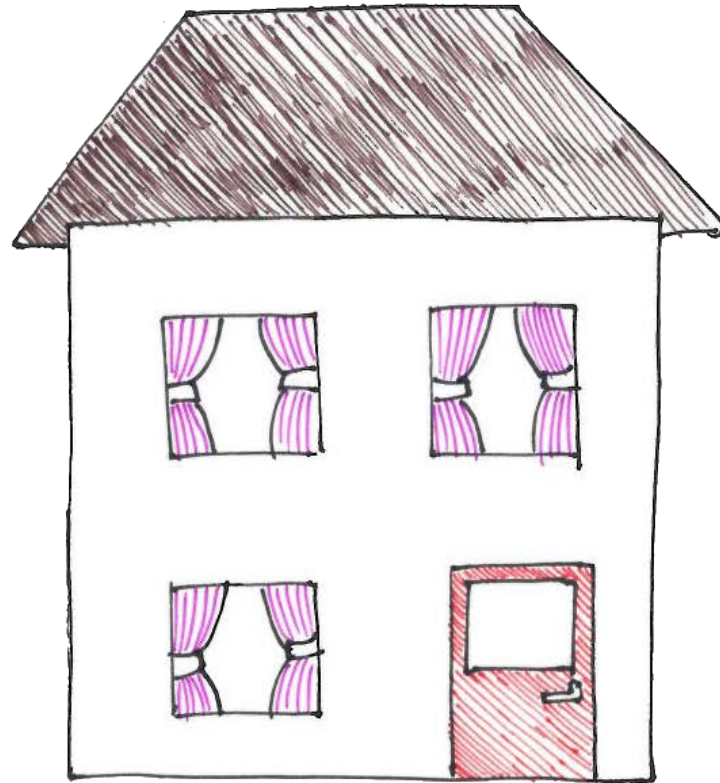
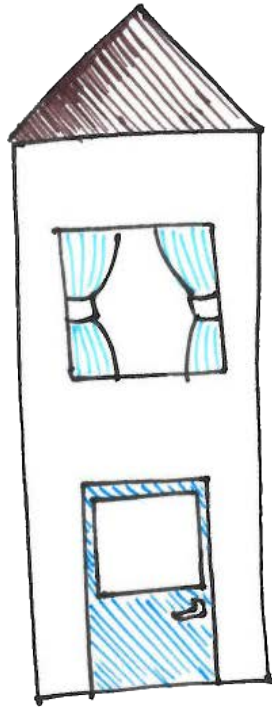
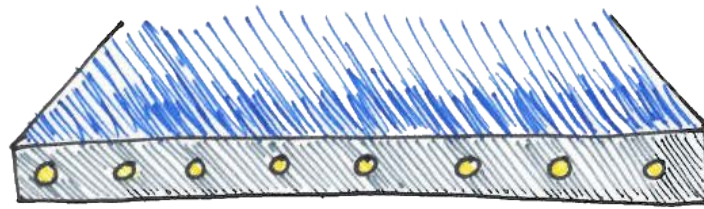


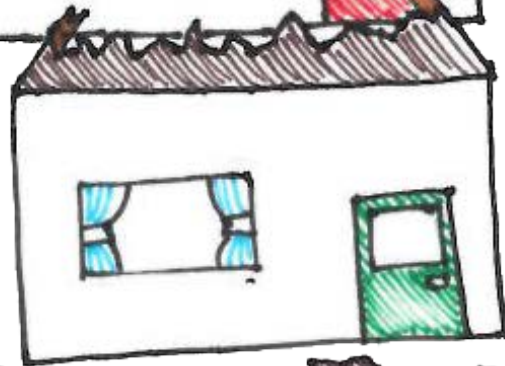
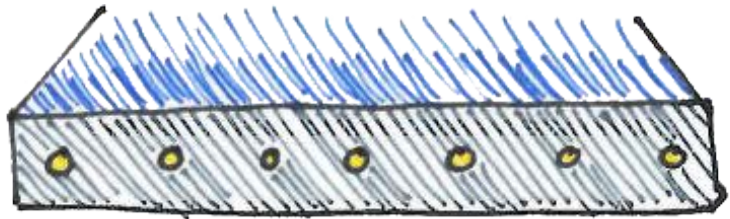




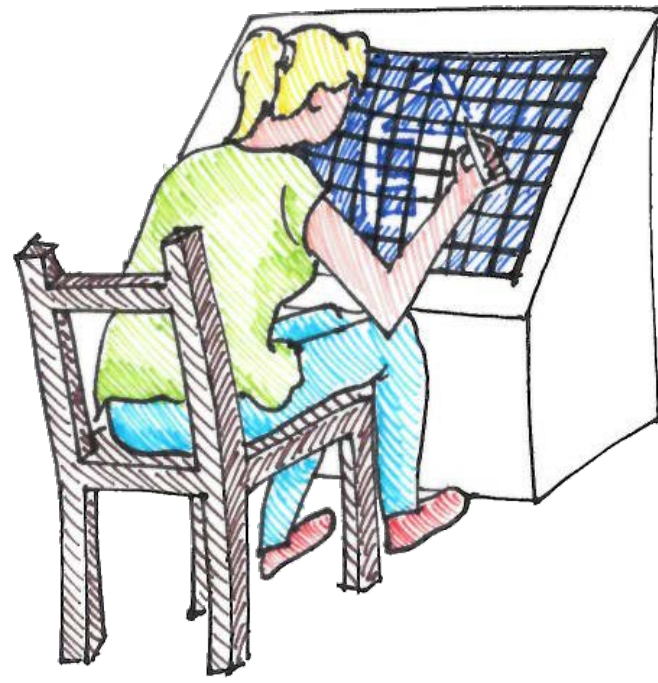
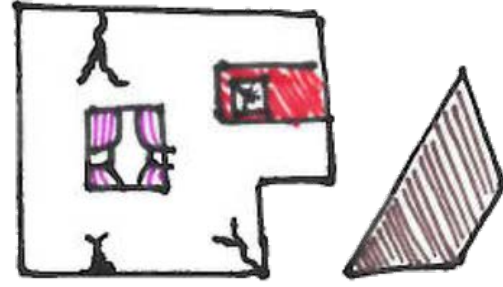
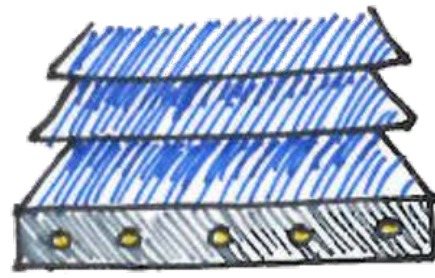












```
./configure  
make
```

```
#!/bin/sh
git clone http://github.com/project/my_project.git &&
cd my_project &&
cmake . &&
make -j
make test
```

```
#!/bin/env sh -e
git clone http://github.com/project/my_project.git project
cd project
mkdir build
cd build
cmake ..
nice make -j4
make test
```



```
#!/bin/env sh -e
apt-get install -y boost
git clone https://github.com/project/my_project.git project
mkdir build
cd build
cmake ../project
nice cmake --build . -- -j4
make test
cpack -G deb
cp *.deb /releases/
```

```
#!/bin/env sh -e
curl -X POST http://build/status -d @in_progress
apt-get install -y boost
git clone https://github.com/project/my_project.git project
git submodule update --init
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=RELWITHDEBINFO ../project
nice cmake --build . -- -j4
valgrind --tool=memcheck ./tests
cpack -G deb
cp *.deb /releases/
curl -X POST http://build/status -d @successful
```































```
#!/bin/env sh -e
curl -X POST -u username:password http://build/status -d @in_progress
apt-get install -y boost
git clone https://github.com/project/my_project.git project
git submodule update -init
for compiler in clang gcc; do
    mkdir $compiler
    cd $compiler
    cmake -DCMAKE_BUILD_TYPE=RELWITHDEBINFO \
          -DCMAKE_EXPORT_COMPILE_COMMANDS=ON \
          -DCMAKE_CXX_COMPILER=/usr/bin/${compiler} ../project
    nice cmake --build . -- -j4
    cd ..
done
cd clang
find ../project -name "*.cpp" -print | xargs clang-tidy
valgrind --tool=memcheck ./tests
cpack -G deb
mkdir /releases/$version
cp *.deb /releases/$version
curl -X POST -u username:password http://build/status -d @successful
```

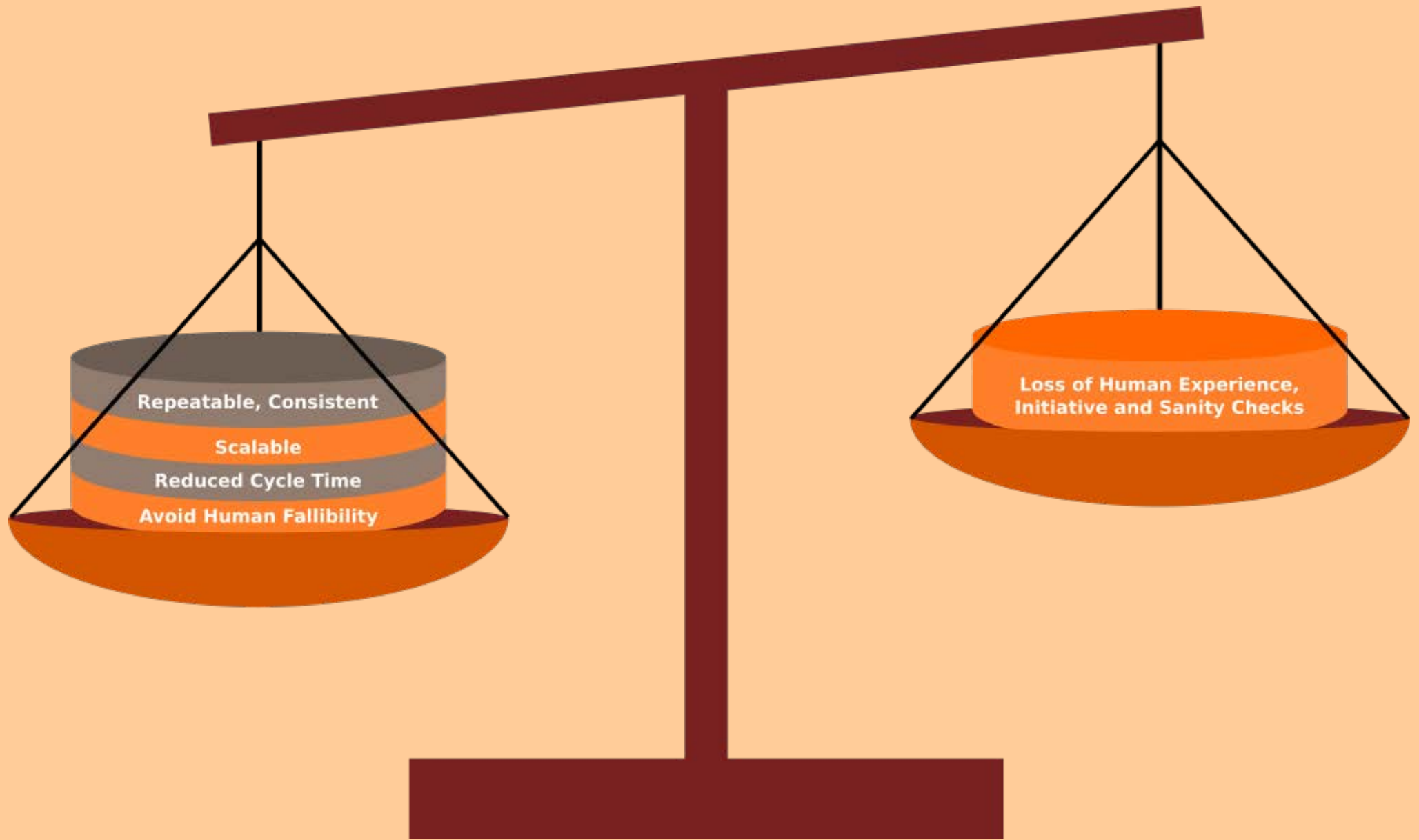
```
#!/bin/env sh -e
POST="curl -x POST -u username:password"
${POST} http://build/status -d #in_progress
apt-get install -y boost
git clone https://github.com/project/my_project.git project
git submodule update -init
for compiler in clang gcc; do
    for bits in 32 64; do
        mkdir ${compiler}-${bits}
        cd ${compiler}-${bits}
        cmake -DCMAKE_BUILD_TYPE=RELWITHDEBINFO \
            -DCMAKE_EXPORT_COMPILE_COMMANDS=ON \
            -DCMAKE_CXX_COMPILER=/usr/bin/${compiler} \
            -DCMAKE_CXX_FLAGS=-m${bits} ../project
        nice cmake --build . -- -j4
        cd ..
    done
done
cd clang
find ../project -name "*.cpp" -print | xargs clang-tidy
valgrind --tool=memcheck ./tests
valgrind --tool=cachegrind ./soak
cpack -G deb
mkdir -p /releases/${version}
cp *.deb /releases/${version}/
chmod 777 /releases/${version}/
${POST} http://build/status -d #successful
```

```
#!/bin/env sh -e
POST="curl -x POST -u username:password"
$(POST) https://build/status -d $in_progress
$(POST) https://slack/post -d "Build of $(version) in progress!"
apt-get install -y boost
git clone https://github.com/project/my_project.git project
git submodule update -init
for compiler in clang gcc; do
    for bits in 32 64; do
        BUILD_DIR="$(compiler)-$(bits)"
        mkdir "$BUILD_DIR"
        cd "$BUILD_DIR"
        cmake -DCMAKE_BUILD_TYPE=RELWITHDEBINFO \
            -DCMAKE_EXPORT_COMPILE_COMMANDS=ON \
            -DCMAKE_CXX_COMPILER="/usr/bin/$(compiler)" \
            -DCMAKE_CXX_FLAGS="-m$(bits)" ../project
        nice cmake --build . -- -j4
        cd ..
    done
done
cd clang
find ../project -name "*.cpp" -print | xargs clang-tidy
valgrind --tool=memcheck ./tests
valgrind --tool=cachegrind ./soak
cpack -G deb
RELEASE_DIR="/releases/$(version)"
mkdir -p "$RELEASE_DIR"
cp *.deb "$RELEASE_DIR/"
chmod -R ugo+r "$RELEASE_DIR"
cd ..
git log >"$RELEASE_DIR/release_notes.txt"
$(POST) https://build/status -d $successful
$(POST) https://slack/post -d "Build of $(version) succeeded!"
```

[Code](#)[Issues 82](#)[Pull requests 2](#)[Wiki](#)[Insights](#)

## Commits

Configure, make, done :-D	 7100bee	<a href="#">&lt;&gt;</a>
Oops – somebody cares about running the tests	 06099ea	<a href="#">&lt;&gt;</a>
Oops – somebody cares about the return code from the tests	 413343c	<a href="#">&lt;&gt;</a>
Publish build success to Slack channel	 87fb7d5	<a href="#">&lt;&gt;</a>
Publish build success to Slack channel only if build succeeded	 8f23870	<a href="#">&lt;&gt;</a>
Cross-compile on half a dozen platforms	 fc508b0	<a href="#">&lt;&gt;</a>
Support parallel builds	 bdabcc7	<a href="#">&lt;&gt;</a>
Throttle Slack channel spam	 ba3819f	<a href="#">&lt;&gt;</a>
Use HTTPS. Oh certificate pain!	 38ba19f	<a href="#">&lt;&gt;</a>
Set up ccache and distcc	 849a6c9	<a href="#">&lt;&gt;</a>
Build on Windows too	 24b0e3d	<a href="#">&lt;&gt;</a>
Escape backslashes	 3ff6b22	<a href="#">&lt;&gt;</a>
Clean workspace between builds	 494077f	<a href="#">&lt;&gt;</a>
Rewrite scripts in Python	 826db7d	<a href="#">&lt;&gt;</a>
Code coverage? Profile builds?	 58f803b	<a href="#">&lt;&gt;</a>
Rewrite scripts as Jenkins pipelines. Or is it Huson? Travis?	 563ee60	<a href="#">&lt;&gt;</a>
Valgrind, sanitizers, static analysis – no dodgy code shall pass!	 dfe5d31	<a href="#">&lt;&gt;</a>
Fix dodgy static analysis on dodgy code	 d49ea36	<a href="#">&lt;&gt;</a>
git submodule for the win!	 5db4c98	<a href="#">&lt;&gt;</a>
git submodule – never again!	 492e190	<a href="#">&lt;&gt;</a>
Roll our own git submodule mechanism	 3878f82	<a href="#">&lt;&gt;</a>
Wrap all the CMake Functions!	 e070d54	<a href="#">&lt;&gt;</a>
Provision prerequisites before building	 f34e283	<a href="#">&lt;&gt;</a>
Update dependency! (This change is coupled to the other repo)	 49de30e	<a href="#">&lt;&gt;</a>
Clone before setting env or set env before cloning?	 725b5eb	<a href="#">&lt;&gt;</a>
Use CPack and deploy packages	 05852e3	<a href="#">&lt;&gt;</a>
RPMs Pkgs, Debs, oh my!	 6f9d24c	<a href="#">&lt;&gt;</a>
Let's try Conan, it can't do any harm	 9af20f1	<a href="#">&lt;&gt;</a>
Docker will solve everything	 554a310	<a href="#">&lt;&gt;</a>
That was not the right way to do containers	 3a40c20	<a href="#">&lt;&gt;</a>
Can I get back to coding again please?	1010S0S	<a href="#">&lt;&gt;</a>
Yet Another Commit Comment /yawn	zzzzzzz	<a href="#">&lt;&gt;</a>



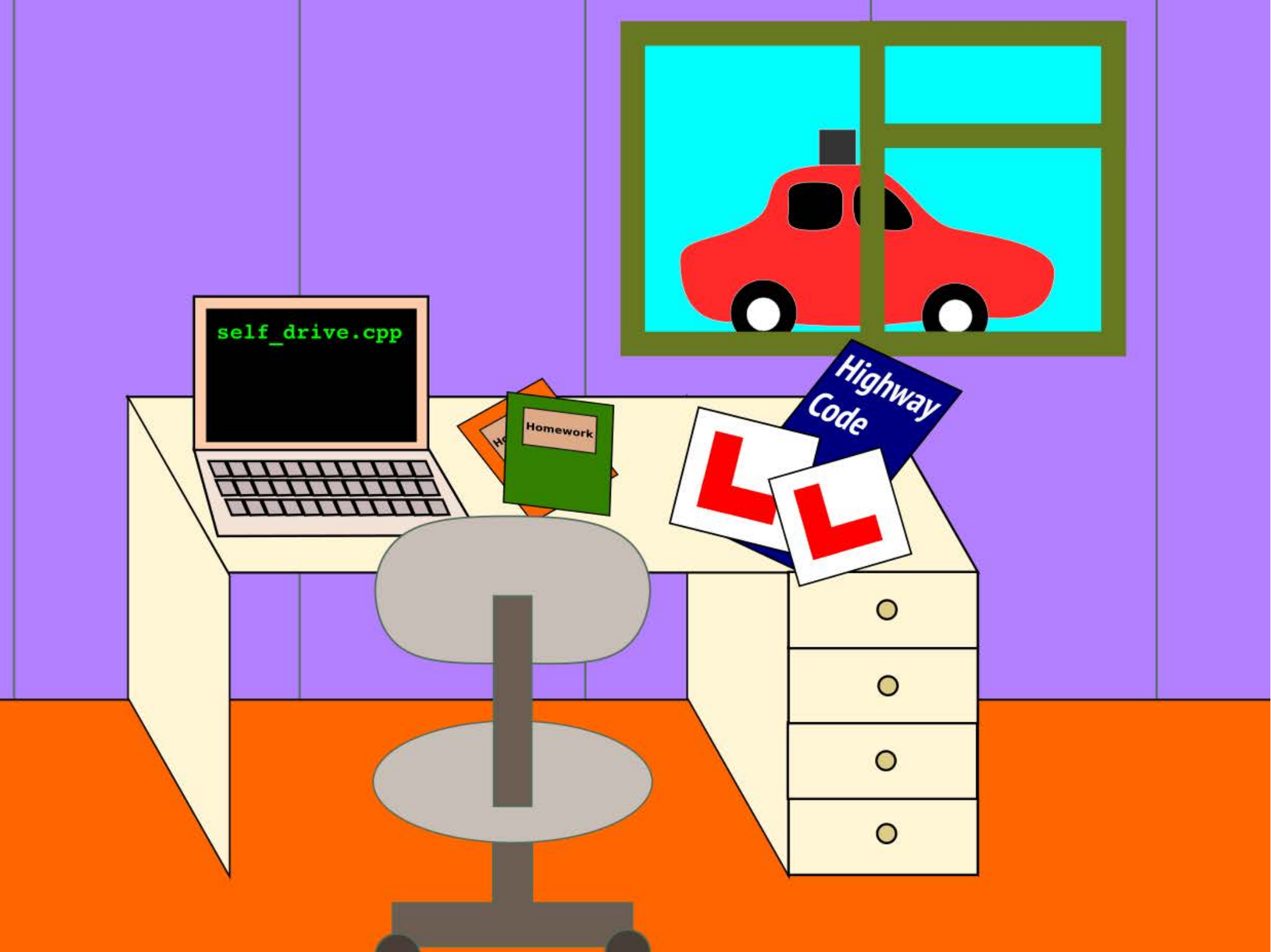
**Repeatable, Consistent**

**Scalable**

**Reduced Cycle Time**

**Avoid Human Fallibility**

**Loss of Human Experience,  
Initiative and Sanity Checks**



self\_drive.cpp

Homework

Highway Code

L

L



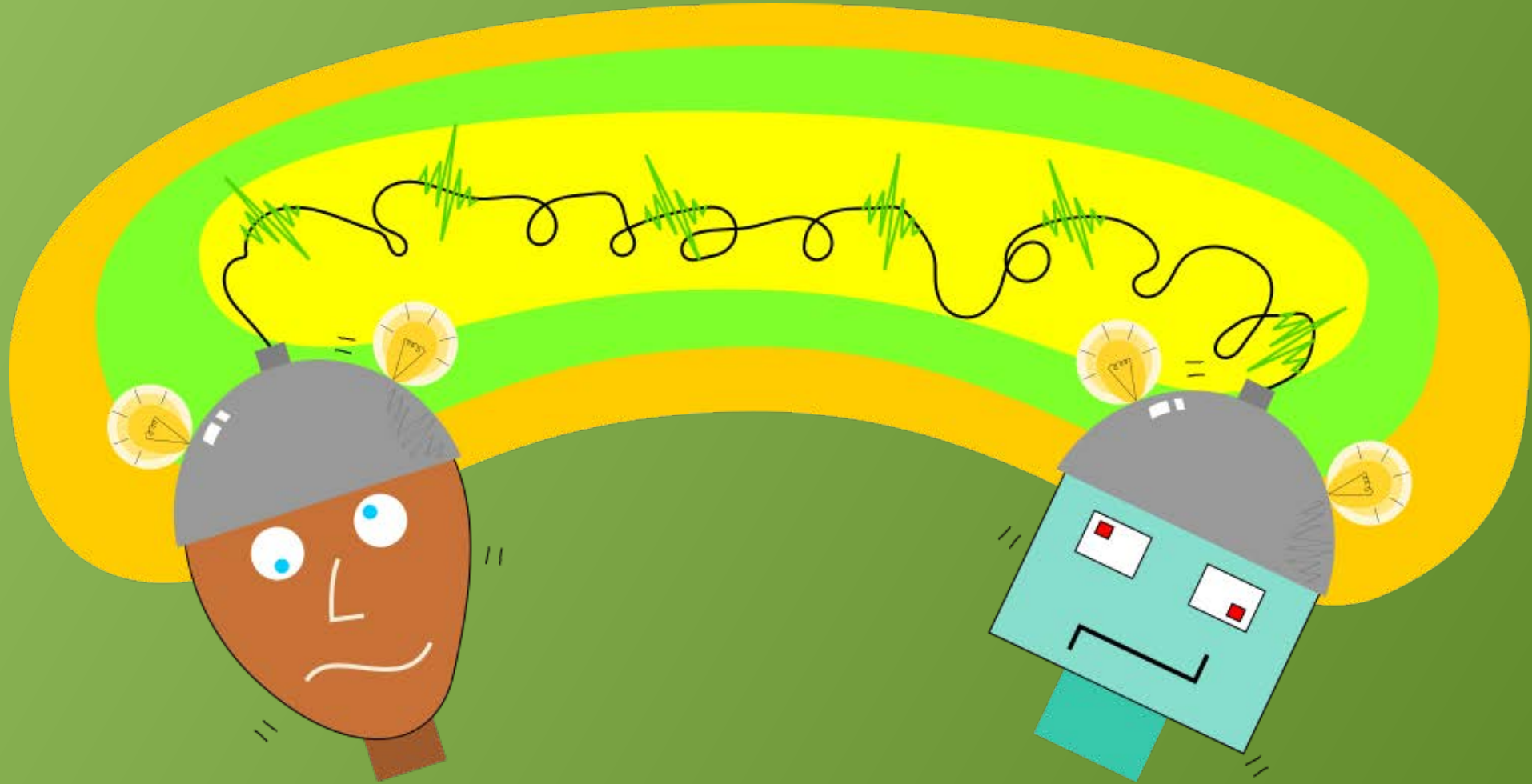
cost to  
automate

cost to  
maintain



cost of  
manual

break even







**INSTANT  
LEGACY!**

Products can become **unmaintainable**  
when their *supporting infrastructure* rots

Development becomes **coupled** to automated tooling –  
making it hard to develop, debug, or run manually

People rely on it working - but **forget** how it works and  
are unfamiliar with the technologies used

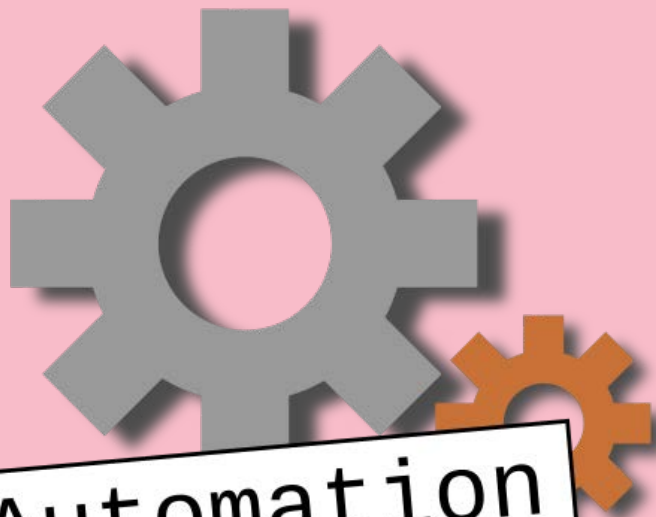
Running automation is **not handed over**  
or is locked to users who have left

Changes **untested?** Untestable? Self-testing?

**Some problems...**

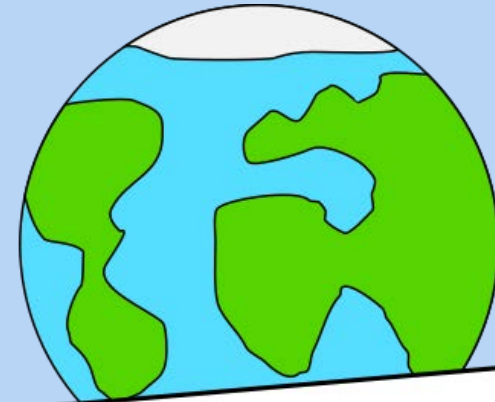
# Considered...

Harmful



Automation

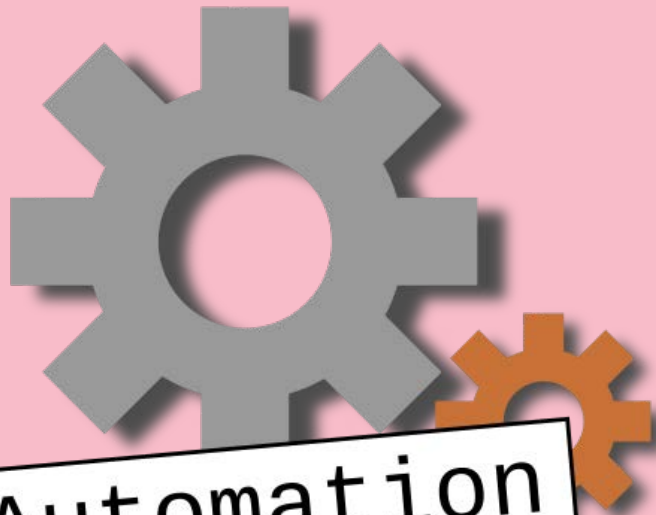
Harmless



Planet Earth

# Considered...

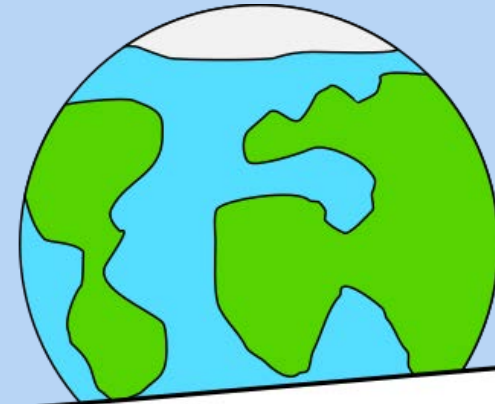
## Harmful



Automation

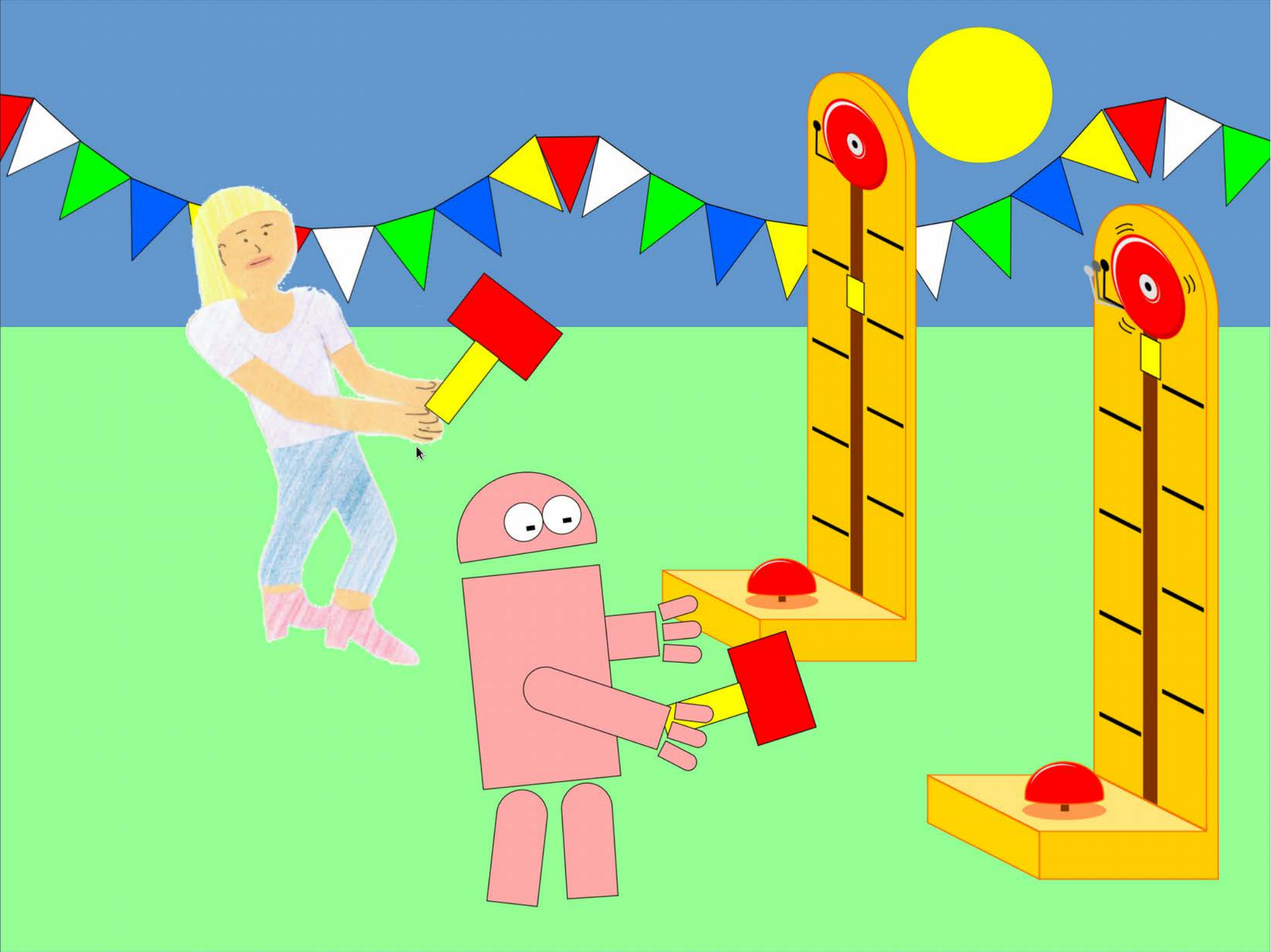
(bad, excessive,  
overly complex,  
premature)

## Harmless

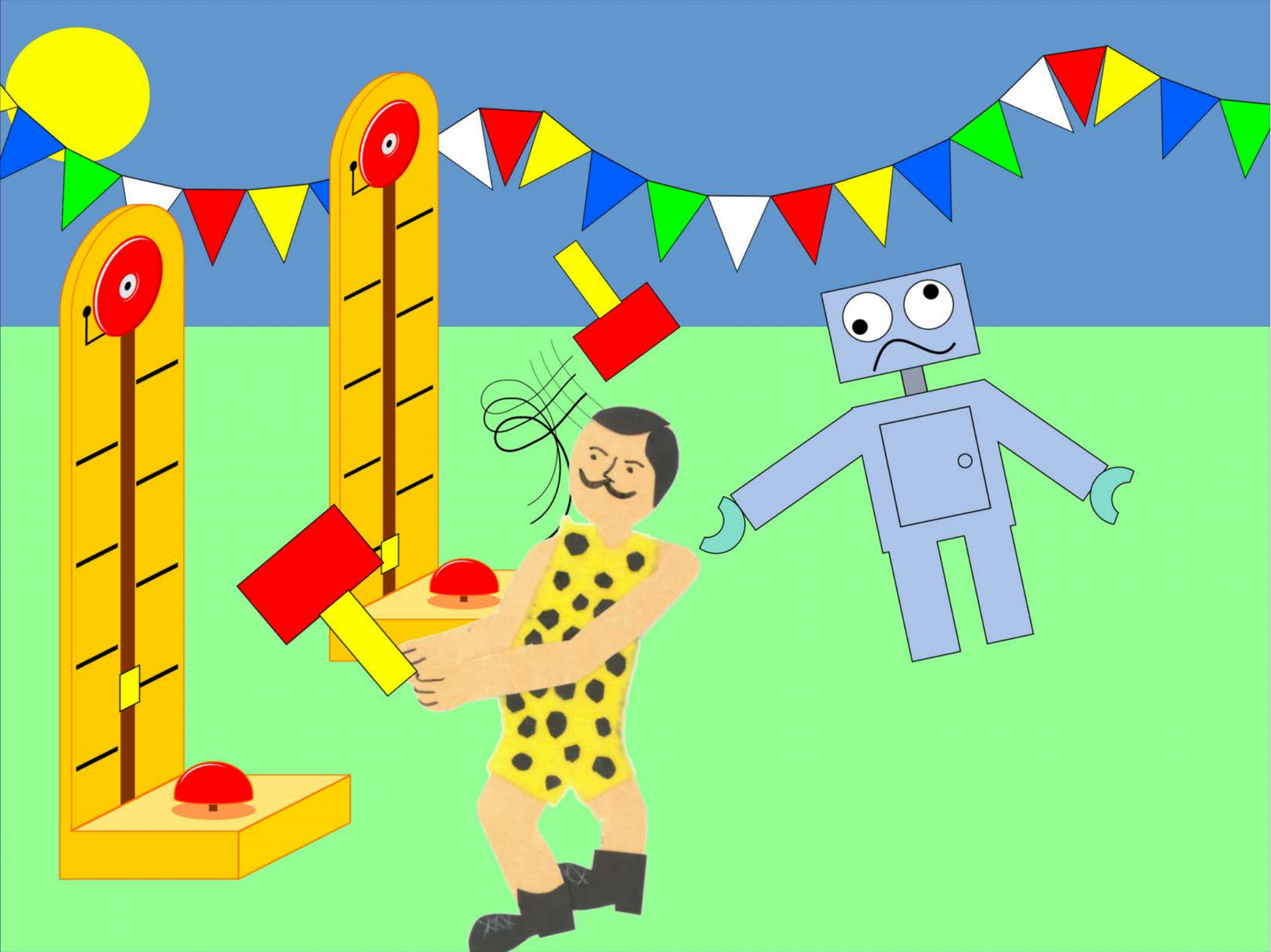


Planet Earth

(mostly)







Is it **viable** (or possible) to automate?

Do I **understand** the kind of solution required?

Do I know how it could **fail**?

How can I ensure it gives me  
the required levels of **assurance**?

## Build on the experience of others

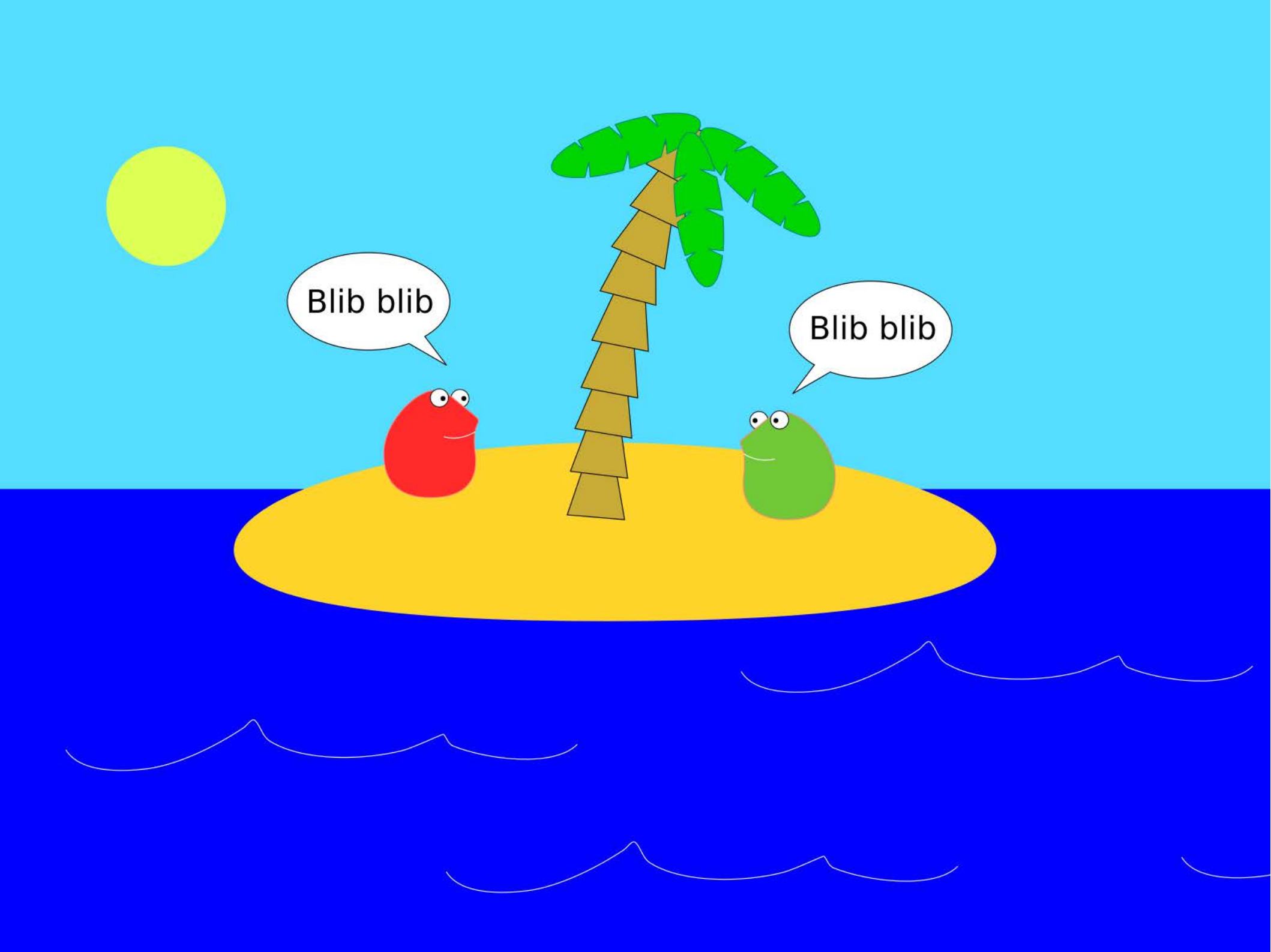
- Draw on people's **collective wisdom**
- Favour mature, open and **compatible tooling** over esoteric and bespoke
- Use **DevOps practices**, e.g. Infrastructure as Code

## Mind the human

- You won't get it right first time
- Build up your own experience
  - Deliberate Practice

## Learn by trying it out

- **Reduce the opportunity for conflict** between human and automated processes
- **Don't run under individuals' accounts** – use an automation account to avoid single points of failure



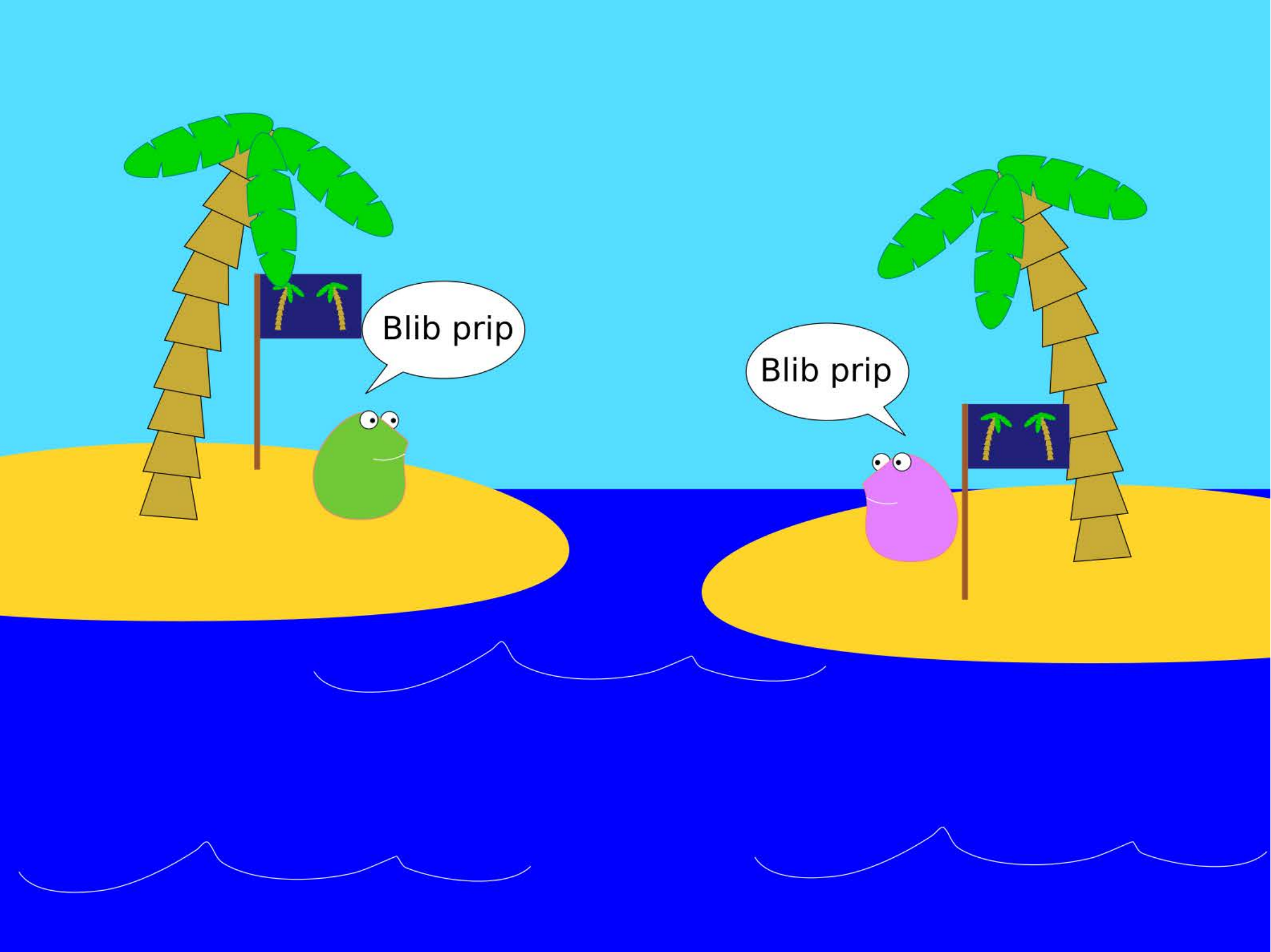
Blib blib

Blib blib



Blib blib

Prip prip



Blib prip

Blib prip

## Design automation to evolve

- Don't write and forget - **let it live**, budget for it to; waterfall deliveries will rot

- Beware vendor lock-in, immature and unsupported tools  
– **use DSLs with care**

- **Keep it simple** - more dependencies and complexity means more frequent maintenance

- **Refactor** for readability and (re)use

- Comment the intent: **say what the code can't**, especially if nuanced or infrequently maintained

- Set and apply **Coding Standards**, review changes against them and combat rot

- **Test** what you can

## Exercise good coding practices

# Automate the solution



only once you've begun to understand the problem



The image features a pair of rich red theater curtains with vertical pleats and gold tassels on the sides. The curtains are drawn back, revealing a dark background. Centered on the curtains is the text "SLASH OR DASH" in a bold, white, sans-serif font, arranged in three lines.

**SLASH  
OR  
DASH**

**CD**

**- / - EITHER - / -**

**Peter Sommerlad - FOOL**  
**Michel Grootjans - Crafting Guitars**  
**Rob Smallshire - The Gender Equality Paradox**  
**Florian Gilcher - Trains**  
**Graham Haynes - On Automati**  
**Marshall Clow - Fuzzing Your Code**  
**Chris Oldwood - The Far Side**  
**Jon Kalb - This is Why We Can't Have Nice Things**  
**Phil Nash - East All The Things**  
**Jim Hague - A Brief of one-line abuses**  
**Mike Seymour - Sparsity Parsery**

# Fuzzing your code

Marshall Clow  
Qualcomm  
[marshall@idio.com](mailto:marshall@idio.com)

# What is fuzzing?

- Pass random (ish) inputs to your code, look for misbehavior. (Over and over)
- Initial fuzzers just generated random inputs
- More modern fuzzers are “guided”

# Profile-guided fuzzing

- Build your program with code coverage enabled.
- After each run, the fuzzer examines the coverage data, and uses the information to generate the next test case.
- American Fuzzy Lop - <http://lcamtuf.coredump.cx/afl/>
- Clang libfuzzer - <https://llvm.org/docs/LibFuzzer.html>

# OSS-Fuzz

## Fuzzing as a service

- Continuous fuzzing for open source projects
- You write glue code that tells OSS-Fuzz how to test your code. It takes some data (ptr, length), and returns an int. 0 ==> success.
- You write a config that tells OSS-Fuzz how to get/build your code
- And... that's it!



# What do you get?

- Ideally ... nothing!
- Whenever OSS-Fuzz finds a problem, it opens an issue in a bugzilla, and sends you an email with the details.
- After the bug is fixed, OSS-Fuzz pulls the new version, builds and tests it, and closes the bug.
- If it doesn't get fixed, after 90 days, the bug is made public.

# An example test

```
extern "C" int LLVMFuzzerTestOneInput
(const uint8_t *data, size_t size)
{
    std::vector<uint8_t> working(data, data + size);
    std::sort(working.begin(), working.end());

    if (!std::is_sorted(working.begin(), working.end()))
        return 1;
    if (!std::is_permutation(data, data + size, working.cbegin()))
        return 99;
    return 0;
}
```

The image features a pair of rich red theater curtains with vertical pleats and gold tassels on the sides. The curtains are drawn back, revealing a dark background. Centered on the curtains is the text "SLASH OR DASH" in a bold, white, sans-serif font, arranged in three lines.

**SLASH  
OR  
DASH**

**SAUL HUDSON**

/// SLASH ///



**Peter Sommerlad - FOOL**  
**Michel Grootjans - Crafting Guitars**  
**Rob Smallshire - The Gender Equality Paradox**  
**Florian Gilcher - Trains**  
**Graham Haynes - On Automati**  
**Marshall Clow - Fuzzing Your Code**  
**Chris Oldwood - The Far Side**  
**Jon Kalb - This is Why We Can't Have Nice Things**  
**Phil Nash - East All The Things**  
**Jim Hague - A Brief of one-line abuses**  
**Mike Seymour - Sparsity Parsery**

# The far\* Side

{ Chris Oldwood



@chrisoldwood / gort@cix.co.uk / chrisoldwood.com



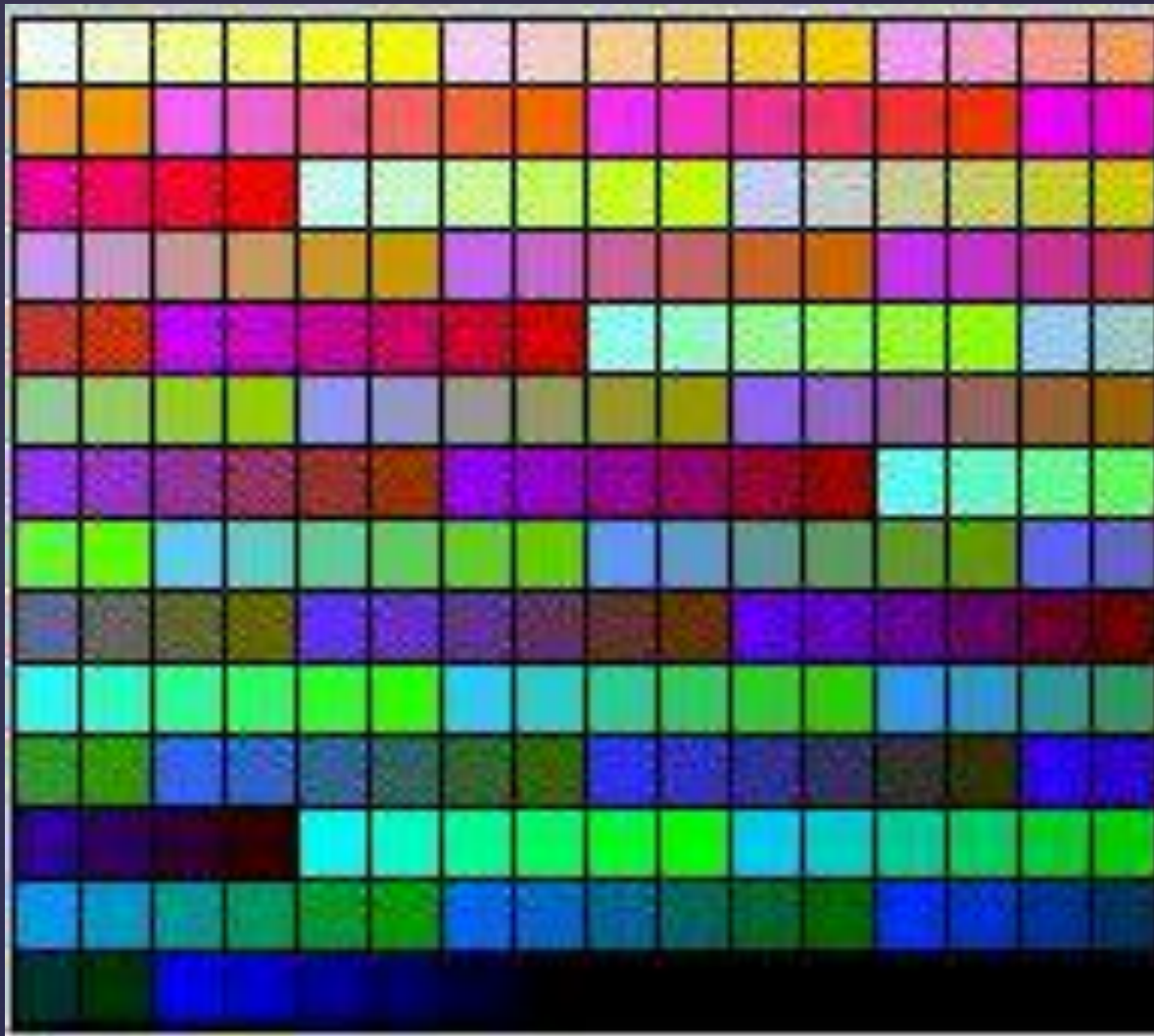












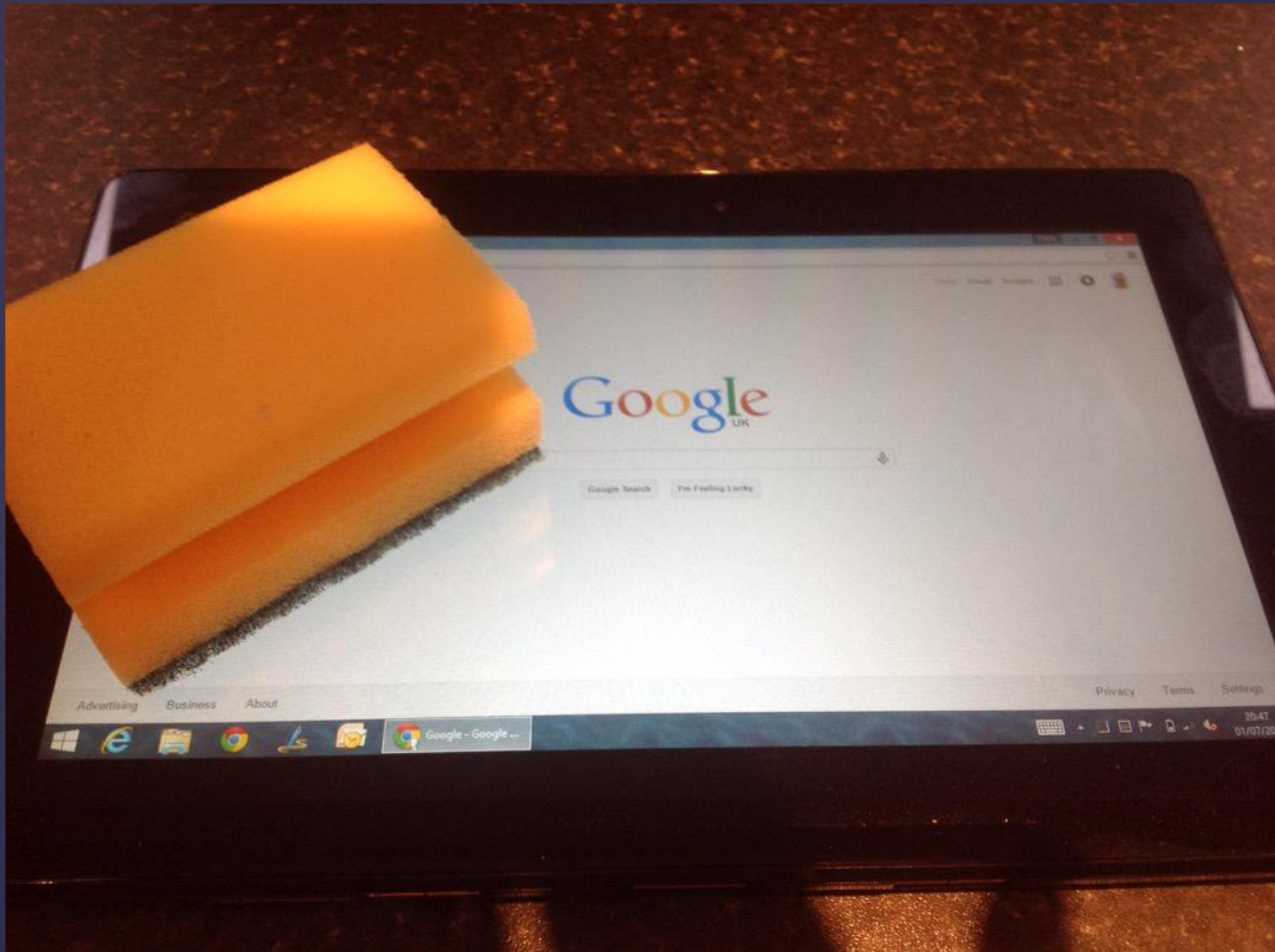
@chrisoldwood / gort@cix.co.uk / chrisoldwood.com











@chrisoldwood / gort@cix.co.uk / chrisoldwood.com





@chrisoldwood / gort@cix.co.uk / chrisoldwood.com





@chrisoldwood / gort@cix.co.uk / chrisoldwood.com









@chrisoldwood / gort@cix.co.uk / chrisoldwood.com





```
5 CLEAR 49999: BORDER 0: PAPER 0: BRIGHT 1: INK 7: CLS
10 FOR i=10 TO 20: BEEP .001+i/300,i: NEXT i: PRINT AT 10,10: INK 1: PAPER 7: BRIGHT 1:
20 FOR i=USR "a" TO USR "t"+7: READ a: POKE i,a: NEXT i
30 DATA 10,0,83,168,85,170,85,138,240,0,222,31,127,255,255,51,10,5,10,0,42,85,42,85,240,240
40 DATA 42,85,42,21,0,5,10,5,254,254,254,252,0,240,240,240,10,5,10,5,10,5,10,5,240,240,240
50 DATA 255,231,195,193,97,51,31,0,255,231,195,131,134,204,248
51 DATA 0,255,85,255,195,129,195,255,0,255,0,255,0,255,0,255,0,0,223,223,223,0,251,251,251
60 DATA 0,251,251,219,251,56,59,0,0,224,224,0,224,224,224,0,126,90,231,126,102,60,24,255,
70 DATA 58,56,254,116,116,116,68,254,0,31,209,255,253,253,129,255,124,198,186,162,186,198
100 FOR x=15616 TO 16383: POKE x+48384,PEEK x: NEXT x
110 RESTORE 200: FOR x=64264 TO 64479: READ y: POKE x,y: NEXT x
200 DATA 124,254,246,254,254,246,246,0,252,254,230,252,230,254,252,0,124,254,246,240,246,254
210 DATA 124,254,240,252,240,254,124,0,124,254,240,252,240,240
211 DATA 240,0,124,254,240,246,246,254,124,0,246,246,254,254,254,246,246,0,254,254,56,56,56
220 DATA 30,30,30,222,222,254,124,0,238,254,252,248,252,254,238,0,240,240,240,240,240,254,
230 DATA 198,238,254,254,254,214,214,0,124,254,254,246,246,246,246,0,124,254,238,238,254,254
240 DATA 124,254,254,246,246,250,124,0,252,254,230,254,252,252,238,0,126,254,248,124,30,254
250 DATA 246,246,246,246,254,254,124,0,246,246,246,246,254,124,56,0,214,214,214,254,254,238
260 DATA 222,222,254,254,30,254,252,0,254,254,62,124,248,254,254,0
280 DATA 220,220,220,0,220,220,220,0
300 RESTORE 310: FOR x=64128 TO 64207: READ y: POKE x,y: NEXT x
```











Add Customer

Customer Details

Name:

OK Close



AB 517097

419 NINE







@chrisoldwood / gort@cix.co.uk / chrisoldwood.com

f /thetrumpet @thep

# pTrumpet.

**authentic. innovative. accessible.**

www

f /thetrumpet

Fun to play. Easy to  
Durable and light

**NUIII PTR**

PC PLAIN 2004/100



The image features a pair of rich red theater curtains with gold tassels, framing the central text. The curtains are drawn back, revealing a dark background behind the text.

**SLASH  
OR  
DASH**

**ITU M.1677**

# **M.1677 : INTERNATIONAL MORSE CODE**

**— — — DASH — — —**

**Peter Sommerlad - FOOL**

**Michel Grootjans - Crafting Guitars**

**Rob Smallshire - The Gender Equality Paradox**

**Florian Gilcher - Trains**

**Graham Haynes - On Automati**

**Marshall Clow - Fuzzing Your Code**

**Chris Oldwood - The Far Side**

**Jon Kalb - This is Why We Can't Have Nice Things**

**Phil Nash - East All The Things**

**Jim Hague - A Brief of one-line abuses**

**Mike Seymour - Sparsity Parsery**



# This is Why We Can't Have Nice Things

Jon Kalb  
ACCU 2018

# *East const*

 **Simon Brand**  
@TartanLlama

Following 

Tag yourself

West-const      East-const  
const T&    vs    T const&

11:16 AM - 6 Nov 2017

22 Retweets 81 Likes



 34     22     81    

# Dan Saks

Welcome Jon | Sign Out | Update Profile

# Dr. Dobb's

THE WORLD OF SOFTWARE DEVELOPMENT

Search:  Search

Search:  Site  Source Code

Home Articles News Blogs Source Code Dobb's TV Webinars & Events

Cloud Mobile Parallel .NET JVM Languages C/C++ Tools Desig

## C/C++

[Tweet](#) [Like 0](#) [Share](#) [G+](#)

---

## Simplifying const Syntax

By Dan Saks, September 26, 2011

[2 Comments](#)

**The simplest way to read and write const declarations correctly is to use an unconventional style**

When support for the const qualifier appeared in compilers some 20 years ago, I learned to



# | *Consistent const*

The rule for const is:



# | *Consistent const*

The rule for const is:

const applies to what is on its left,

# | *Consistent const*

The rule for const is:

const applies to what is on its left,  
unless there is nothing to its left,

# | *Consistent const*

The rule for const is:

const applies to what is on its left,  
unless there is nothing to its left,  
then it applies to what's on its right.

# | *Consistent const*

*My* rule for const is:

# | *Consistent const*

*My* rule for const is:

const applies to what is on its left

# | *Consistent const*

*My* rule for const is:

const applies to what is on its left.

# | *Read Order*

*When declaring an integer const,  
don't you want to read it as "a constant integer?"*

# | *Read Order*

**Read declarations “inside out” and “right to left.”**



# | *Read Order*

Read declarations “inside out” and “right to left.”

```
void (*fn)(int);
```

# | *Read Order*

Read declarations “inside out” and “right to left.”

```
void (*fn)(int);
```

“fn is a pointer to a function that takes an int and returns void.”

# | *Read Order*

**Read declarations “inside out” and “right to left.”**

# | *Read Order*

Read declarations “inside out” and “right to left.”

```
long (&ra)[3] = a;
```

# | *Read Order*

Read declarations “inside out” and “right to left.”

```
long (&ra)[3] = a;
```

“ra is a reference to an array of three longs.”

# | *Read Order*

**Read declarations “inside out” and “right to left.”**

# | *Read Order*

Read declarations “inside out” and “right to left.”

```
char * const pc = str;
```

# | *Read Order*

Read declarations “inside out” and “right to left.”

```
char * const pc = str;
```

“pc is a constant pointer to character.”



# | *Read Order*

**Read declarations “inside out” and “right to left.”**

# | *Read Order*

Read declarations “inside out” and “right to left.”

```
char const * const pc = str;
```

# | *Read Order*

Read declarations “inside out” and “right to left.”

```
char const * const pc = str;
```

“pc is a constant pointer to constant character.”

# | *Read Order*

**Read declarations “inside out” and “right to left.”**

# | *Read Order*

Read declarations “inside out” and “right to left.”

```
int const & ri = i;
```

# | *Read Order*

Read declarations “inside out” and “right to left.”

```
int const & ri = i;
```

“ri is a reference to a constant interger.”

# | *Required East const Cases*

Constant pointers

```
char * const pc = str;
```

Constant member functions

```
size_type size() const;
```

## | *What does this mean?*

**This case may be confusing to for programmers that don't know the const rule.**

```
char const * pc = str;
```



## | *What does this mean?*

This case may be confusing to for programmers that expect the `const` to apply to the “Widget” in the alias.

```
using WidgetPtr = Widget*;
```

```
const WidgetPtr wp = &w;
```

## | *What does this mean?*

This case may be confusing to for programmers that expect the `const` to apply to the “Widget” in the alias.

```
using WidgetPtr = Widget*;
```

```
const WidgetPtr wp = &w;
```

```
WidgetPtr const wp = &w;
```

# A Foolish Consistency



Jon Kalb

2018-02-26

comments

Edit

## The Hobgoblin of Little Minds



Ralph Waldo Emerson famously said, “A foolish consistency is the hobgoblin of little minds, adored by little statesmen and philosophers and divines.” I don’t think he was talking about code, but that statement couldn’t be more relevant to software engineers.

I’ve experienced a scenario like this a number of times in my career:

*I’m sharing a new approach to writing code that offers some clear improvements to what we’ve been doing. Perhaps it is more readable, more efficient, or safer. But the response that I hear from colleagues is, “But we can’t do that here. We have <some large number> lines of code where we didn’t do it that way, so it wouldn’t be consistent.”*

**This is Why We Can’t Have Nice Things**

## NL.26: Use conventional `const` notation

### Reason

Conventional notation is more familiar to more programmers. Consistency in large code bases.

### Example

```
const int x = 7;    // OK
int const y = 9;    // bad

const int *const p = nullptr; // OK, constant pointer to constant int
int const *const p = nullptr; // bad, constant pointer to constant int
```

### Note

We are well aware that you could claim the "bad" examples more logical than the ones marked "OK", but they also confuse more people, especially novices relying on teaching material using the far more common, conventional OK style.

As ever, remember that the aim of these naming and layout rules is consistency and that aesthetics vary immensely.

### Enforcement

Flag `const` used as a suffix for a type.

## NL.26: Use conventional `const` notation

### Reason

Conventional notation is more familiar to more programmers. Consistency in large code bases.

### Example

```
const int x = 7;    // OK
int const y = 9;    // bad

const int *const p = nullptr; // OK, constant pointer to constant int
int const *const p = nullptr; // bad, constant pointer to constant int
```

### Note

We are well aware that you could claim the "bad" examples **more logical** than the ones marked "OK", but they also confuse more people, especially novices relying on teaching material using the far more common, conventional OK style.

As ever, remember that the aim of these naming and layout rules is consistency and that aesthetics vary immensely.

### Enforcement

Flag `const` used as a suffix for a type.

## NL.26: Use conventional `const` notation

### Reason

Conventional notation is more familiar to more programmers. Consistency in large code bases.

### Example

```
const int x = 7;    // OK
int const y = 9;    // bad

const int *const p = nullptr; // OK, constant pointer to constant int
int const *const p = nullptr; // bad, constant pointer to constant int
```

### Note

We are well aware that you could claim the "bad" examples more logical than the ones marked "OK", but they also confuse more people, especially novices relying on teaching material using the far more common, conventional OK style.

As ever, remember that the aim of these naming and layout rules is consistency and that aesthetics vary immensely.

### Enforcement

Flag `const` used as a suffix for a type.

## NL.26: Use conventional `const` notation

### Reason

Conventional notation is more familiar to more programmers. Consistency in large code bases.

### Example

```
const int x = 7;    // OK
int const y = 9;    // bad

const int *const p = nullptr; // OK, constant pointer to constant int
int const *const p = nullptr; // bad, constant pointer to constant int
```

### Note

We are well aware that you could claim the "bad" examples more logical than the ones marked "OK", but they also confuse more people, especially novices relying on teaching material using the far more common, conventional OK style.

As ever, remember that the aim of these naming and layout rules is consistency and that aesthetics vary immensely.

### Enforcement

Flag `const` used as a suffix for a type.

## NL.26: Use conventional `const` notation

### Reason

Conventional notation is more familiar to more programmers. Consistency in large code bases.

### Example

```
const int x = 7;    // OK
int const y = 9;    // bad

const int *const p = nullptr; // OK, constant pointer to constant int
int const *const p = nullptr; // bad, constant pointer to constant int
```

### Note

We are well aware that you could claim the "bad" examples more logical than the ones marked "OK", but they also confuse more people, especially novices relying on teaching material using the far more common, conventional OK style.

As ever, remember that the aim of these naming and layout rules is consistency and that aesthetics vary immensely.

### Enforcement

Flag `const` used as a suffix for a type.



| This is why we can't have nice things

| This is why we can't have nice things

**If you must remain consistent with the  
past, you can never improve.**

***Join the Revolution***



# Join the Revolution



## // info

comments on c++ and issues of interest to c++ programmers

# Petition for const Consistency

The background of the image consists of rich red theater curtains with deep vertical pleats. The curtains are pulled back slightly on both sides, revealing gold tassels hanging from the top edge. The lighting is dramatic, with the center of the stage area being brighter than the edges.

**SLASH  
OR  
DASH**

**COMMENTS**

# COMMENTS

C++

**/// SLASH ///**



**COMMENTS**

**LUA**

**— — — DASH — — —**

**COMMENTS**

**SQL**

**/// EITHER ---**

**Peter Sommerlad - FOOL**  
**Michel Grootjans - Crafting Guitars**  
**Rob Smallshire - The Gender Equality Paradox**  
**Florian Gilcher - Trains**  
**Graham Haynes - On Automati**  
**Marshall Clow - Fuzzing Your Code**  
**Chris Oldwood - The Far Side**  
**Jon Kalb - This is Why We Can't Have Nice Things**  
**Phil Nash - East All The Things**  
**Jim Hague - A Brief of one-line abuses**  
**Mike Seymour - Sparsity Parsery**

**We have always been at  
war with West Constia**

```
auto someFunc( int i ) -> std::string;
```

```
// instead of
```

```
std::string someFunc( int i );
```

```
auto someFunc( int i ) const -> std::string;
```

```
// instead of
```

```
std::string someFunc( int i ) const;
```



```
template <typename Lhs, typename Rhs>
auto add( Lhs const& lhs, Rhs const& rhs ) -> decltype( lhs + rhs ) {
    return lhs + rhs;
}
```

```
auto lambda = [] -> double { return 0; }
```

# Simplify C++!

[WHY "SIMPLIFY C++"?](#) [CONTRIBUTIONS WANTED!](#) [ABOUT ME](#) [CONTACT](#)

*Write clean and maintainable C++*



[Home](#) > [2016](#) > [November](#) > [Trailing return types everywhere](#)

## Trailing return types every- where

RECENT POSTS

# Trailing return types everywhere

Arne Mertz November 30, 2016 11

Contents [\[show\]](#)

Trailing return types are an oddity in C++ – we should use them only when necessary.

A few days ago one of my coworkers asked me to explain an [odd line of code](#) he had encountered in an open source library. The line was similar to this:

```
auto getMulticastHops() const -> int;
```

Some people will know that this is a way of declaring functions that came into the language with C++11. The part `-> int` is called “trailing return type”, and the line is exactly the same as

```
int getMulticastHops() const;
```

## RECENT POSTS

- [Tailor Standard Containers to Your Needs](#) April 11, 2018
- [Raw Pointers Are Gonna Stay!](#) April 4, 2018
- [Raw Pointers Are Gone!](#) April 1, 2018

## ★ POPULAR POSTS

C++ Online Compilers  
[#include - Don't get fancy](#)  
[Clean Code Generation](#)

## 🕒 ARCHIVES

Select Month

## 📁 CATEGORIES

- C++ (94)

Modern C++ Features (88)

# Trailing return types everywhere

Arne Mertz November 30, 2016 11

Contents [show]

Trailing return types are an oddity in C++ – we should use them only when necessary.

A few days ago one of my coworkers asked me to explain an odd line of code he had encountered in an open source library. The line was similar to this:

```
auto getMulticastHops() const -> int;
```

Some people will know that this is a way of declaring functions that came into the language with C++11. The part `-> int` is called “trailing return type”, and the line is exactly the same as

```
int getMulticastHops() const;
```

## RECENT POSTS

- Tailor Standard Containers to Your Needs April 11, 2018
- Raw Pointers Are Gonna Stay! April 4, 2018
- Raw Pointers Are Gone! April 1, 2018

## ★ POPULAR POSTS

C++ Online Compilers  
#include - Don't get fancy  
Clean Code Generation

## 🕒 ARCHIVES

Select Month

## 📁 CATEGORIES

- C++ (94)
- Modern C++ Features (88)

**But...**

Swift

```
func factorial(of : Int) -> Int
```

Swift

```
func factorial(of : Int) -> Int
```

Haskell

```
factorial :: (Integral a) => a -> a
```



Swift

```
func factorial(of : Int) -> Int
```

Haskell

```
factorial :: (Integral a) => a -> a
```

Maths

$$f(x) \rightarrow y$$

```
auto doesItBlend() -> bool;  
auto whatsYourFavouriteNumber() -> int;  
auto add( double a, double b ) -> double;  
void setTheControls();
```

```
auto doesItBlend() -> bool;  
auto whatsYourFavouriteNumber() -> int;  
auto add( double a, double b ) -> double;  
void setTheControls();
```

```
virtual void foo();  
virtual auto bar() -> int;
```

// info

comments on c++ and issues of interest to c++ programmers

C++ Community Calendar

CppChat

East const

About Jon...

# A Foolish Consistency



Jon Kalb

2018-02-26

comments

## The Hobgoblin of Little Minds



Ralph Waldo Emerson famously said, “A foolish consistency is the hobgoblin of little minds, adored by little statesmen and philosophers and divines.” I don’t think he

---

### RECENT

- [Developing Talk Ideas](#)
- [A Foolish Consistency](#)
- [My take at times](#)
- [Undefined Behavior and CERT’s Vulnerability Note](#)
- [Undefined Behavior and Apple’s Secure Coding Guide](#)

---

### ARCHIVES

```
template <typename Lhs, typename Rhs>  
auto add( Lhs const& lhs, Rhs const& rhs ) -> decltype( lhs + rhs ) {  
    return lhs + rhs;  
}
```

```
auto lambda = [] -> double { return 0; }
```

```
#define func auto
```

```
#define func auto
```

```
#define var auto
```

```
#define let auto const
```

```
#define func auto
```

```
#define var auto
```

```
#define let auto const
```

```
func len( std::string s ) -> size_t {  
    let length = s.size();  
    return length;  
}
```



# Level of Indirection

*All problems in computer science can be solved by another level of indirection*

## East End Functions

6th April 2018 at 17:40

There has been a recent stirring of attention, in the C++ community, for the practice of always placing the `const` modifier to the right of the thing it modifies. The practice has even been gifted a catchy name: East Const (which, I think, is what has stirred up the interest).

As purely a matter of style it's fascinating that it seems to have split the community so strongly! There are cases for and against, but both sides seem to revolve around the idea of "consistency". For the East Const believers the consistency is in the sense that you can always apply one, simple, rule about what `const` means and where it goes. For the West Consts the consistency is with the majority of existing code out there - as

[home](#)  
[index](#)  
[appearances](#)  
[about](#)  
[rss](#)

The image features a pair of rich red theater curtains with vertical pleats and gold tassels on the sides. The curtains are drawn back, revealing a dark background. Centered on the curtains is the text "SLASH OR DASH" in a bold, white, sans-serif font, arranged in three lines.

**SLASH  
OR  
DASH**

**2018-04-13**

# FRIDAY THE 13<sup>TH</sup>

/// SLASH ///



**Peter Sommerlad - FOOL**  
**Michel Grootjans - Crafting Guitars**  
**Rob Smallshire - The Gender Equality Paradox**  
**Florian Gilcher - Trains**  
**Graham Haynes - On Automati**  
**Marshall Clow - Fuzzing Your Code**  
**Chris Oldwood - The Far Side**  
**Jon Kalb - This is Why We Can't Have Nice Things**  
**Phil Nash - East All The Things**  
**Jim Hague - A Brief of one-line abuses**  
**Mike Seymour - Sparsity Parsery**

# A Brief History Of Online Abuses

Jim Hague  
jim.hague@acm.org  
@banburybill

SPAM

Path: gmd.de!xlink.net!rz.uni-karlsruhe.de!news.uni-stuttgart.de!news.be  
From: nike@indirect.com (Laurence Canter)  
Newsgroups: rec.juggling,us.legal  
Subject: Green Card Lottery- Final One?  
Date: 12 Apr 1994 08:12:17 GMT  
Organization: Canter & Siegel  
Lines: 34  
Message-ID: <2odl51\$45t@herald.indirect.com>  
NNTP-Posting-Host: idl.indirect.com

Green Card Lottery 1994 May Be The Last One!  
THE DEADLINE HAS BEEN ANNOUNCED.

The Green Card Lottery is a completely legal program giving away a certain annual allotment of Green Cards to persons born in certain countries. The lottery program was scheduled to continue on a permanent basis. However, recently, Senator Alan J Simpson introduced a bill into the U. S. Congress which could end any future lotteries. THE 1994 LOTTERY IS SCHEDULED TO TAKE PLACE SOON, BUT IT MAY BE THE VERY LAST ONE.

PERSONS BORN IN MOST COUNTRIES QUALIFY, MANY FOR FIRST TIME.

The only countries NOT qualifying are: Mexico; India; P.R. China; Taiwan, Philippines, North Korea, Canada, United Kingdom (except Northern Ireland), Jamaica, Dominican Republic, El Salvador and Vietnam.

Lottery registration will take place soon. 55,000 Green Cards will be given to those who register correctly. NO JOB IS REQUIRED.

THERE IS A STRICT JUNE DEADLINE. THE TIME TO START IS NOW!!

For FREE information via Email, send request to  
cslaw@indirect.com

--

\*\*\*\*\*  
Canter & Siegel, Immigration Attorneys  
3333 E Camelback Road, Ste 250, Phoenix AZ 85018 USA  
cslaw@indirect.com telephone (602)661-3911 Fax (602) 451-7617



29<sup>th</sup> March

1864

---

*TO THE EDITOR OF THE TIMES.*

---

Sir,—On my arrival home late yesterday evening a “telegram,” by “London District Telegraph,” addressed in full to me, was put into my hands. It was as follows:—

“Messrs. Gabriel, dentists, 27, Harley-street, Cavendish-square. Until October Messrs. Gabriel’s professional attendance at 27, Harley-street, will be 10 till 5.”

I have never had any dealings with Messrs. Gabriel, and beg to ask by what right do they disturb me by a telegram which is evidently simply the medium of advertisement? A word from you would, I feel sure, put a stop to this intolerable nuisance. I enclose the telegram, and am,  
Your faithful servant,

Upper Grosvenor-street, May 30.

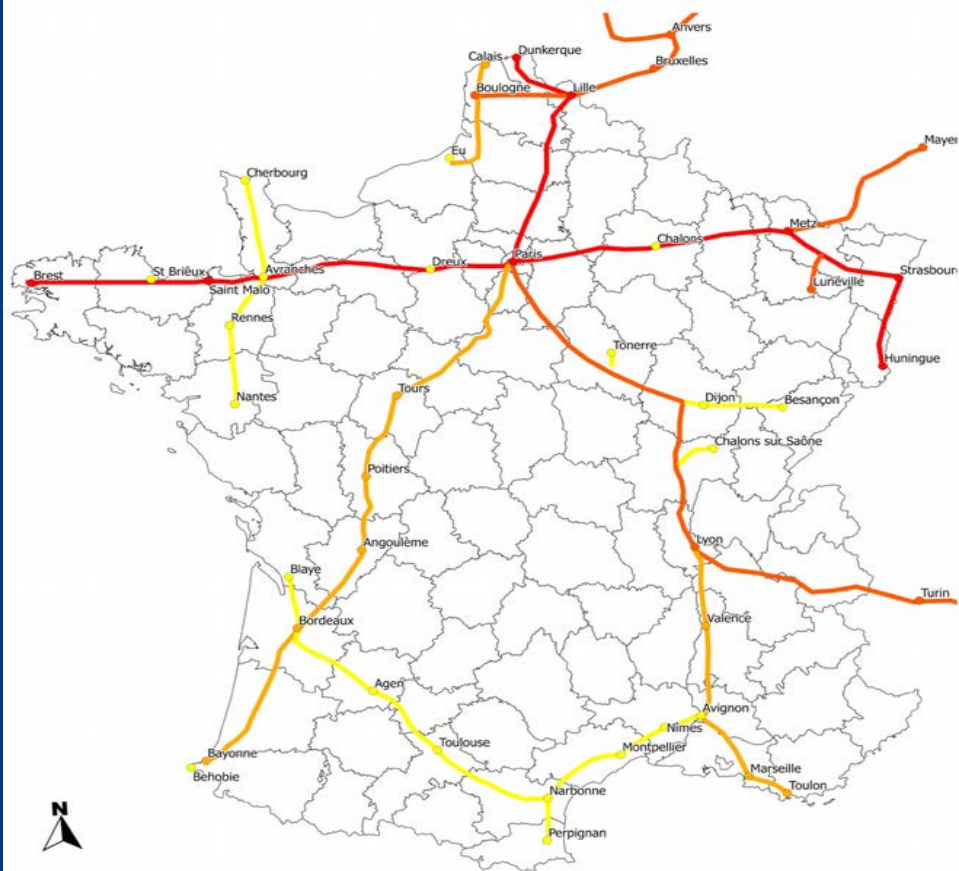
---

M. P.

WIREF

1834





### Le réseau Chappe en France

Directions (date de création)

- 1793-1800
- 1800-1815
- 1815-1830
- Après 1830

Lignes (date de création)

- 1793-1800
- 1800-1815
- 1815-1830
- Après 1830

419

180?



Military fortress of Barcelona 11/11/65

Mrs. Paul Webb

Dear Sir,

Although I only have the honour of being acquainted with you by the references that my dear wife a relative of yours gave me and that remembering the personalities of our family, always venerated the honesty and good qualities that distinguished you, I address myself to you perhaps for the first and last time, because the gravness of my health compels me to make you know my sad position and beg your assistance and protection for my only daughter Mary, a young girl fourteen years old who is now in a Pension House, owing inform you that my deceased wife was Mrs. Elizabeth Webb.

As I am here very watched by my enemies I must pray you to keep a whole discretion and to trust not anybody with the slightest particulars of this letter.

Being the private Secretary and Treasurer of General Moratinos Campes in the last Cuban war and enjoying his whole confidence, I succeeded to make a good position to my daughter and by transactions with public funds I saw my fortune increase every day, although with the sorrow of losing her mother, my very dear wife, I would have succeeded in all my wishes, should my protector have followed in his place till the end of war; but General Weyler came to take the commandment of all the army and as I could not accompany my protector to Spain we could suffer to be under the orders of a political adversary I deserted and joined the army of rebellion in behalf of Republic. But there we were victims of the greatest treason and I was compelled to emigrate to English ground with all my property valuable £ 37000.

When I was being some time in London I received the sad news that my wife had suddenly deceased leaving my dear daughter in despair and without any help. Under this misfortune I was compelled to come back to Spain, intending to take my daughter and bring her to America, but before starting and considering how imprudent it was to take along with me so important an amount, I decided to place it in a sure English Bank against a private contract and only as a deposit, as it appears from the receipt payable to bearer that the Bank gave me, as a guarantee, whose receipt I kept into a secret drawer of a portmanteau, so cleverly made that the nearest eye cannot find it out.

Then, wholly satisfied that the money was assured I embarked for Spain, but on my arrival I was recognised, arrested and brought before the military authorities, that by my desertion to the

enemy and without considering my political antecedents, condemned me to sixteen years of penal servitude that I must extinguish in the fortress, when I am suffering so bitterly that I am deprived of all communication outside, even with my daughter.

When I was sentenced, all my luggage, as well as my portmanteau remained saved at the disposal of that military Tribunal in order to respond to the payment of all the charges of my process.

I am only visited here by the kind Chaplain of the castle, who is become my best friend and protector and it is thanks to him that I can write this letter, but I am aware of the gravness of my health and I foresee a very short and fatal end for me, by his reason and trusting on your discretion I dare beg of you protection, praying to be good enough to be my daughter's support and make her happy, as I fear that I shall never see her again.

My sized luggage is kept by the military authorities, but only you are acknowledged with the existence of the secret drawer. If you are so good as to become my dear daughter's protector and advance the necessary amount to rescue my luggage, I hope you will inform me by cable. I then will send the kind Chaplain and my daughter to your house with all my luggage and I will also send you my last will, in which I will bequeath the fourth part of all my property to your profit as a reward for your bounty and assistance.

Occasionally and trusting always in your discretion, all my daughter's future, I remain

Faithfully yours

Luis Cabrer

P.S. As you can perfectly understand I cannot receive your reply directly, but in order to win time I beg you, if you accept my proposals to send me a cablegram telling me to be addressed as follows: Richard Estela de Borros = Vich = Barcelona = Spain

Sent sample  
Webb

What has been will be again,  
what has been done will be done again;  
there is nothing new under the sun.

*Ecclesiastes 1:9*

The image features a pair of rich red theater curtains with vertical pleats and gold tassels on the sides. The curtains are drawn back, revealing a dark background. Centered on the curtains is the text "SLASH OR DASH" in a bold, white, sans-serif font, arranged in three lines.

**SLASH  
OR  
DASH**



DASH

--- DASH ---

Disney PRESENTS A PIXAR FILM



**THE INCREDIBLES**  
IN THEATERS 11.5.04

**Peter Sommerlad - FOOL**  
**Michel Grootjans - Crafting Guitars**  
**Rob Smallshire - The Gender Equality Paradox**  
**Florian Gilcher - Trains**  
**Graham Haynes - On Automati**  
**Marshall Clow - Fuzzing Your Code**  
**Chris Oldwood - The Far Side**  
**Jon Kalb - This is Why We Can't Have Nice Things**  
**Phil Nash - East All The Things**  
**Jim Hague - A Brief of one-line abuses**  
**Mike Seymour - Sparsity Parsery**

# Sparsity Parsery

Or

Compile-time trickery for dealing  
with sparse key sets

Mike Seymour

[github.com/mikeseymour/wocca](https://github.com/mikeseymour/wocca)

# Problem

- Receiving key-value tags with integer keys
- Keys cover a large range of values
- Messages might contain many tags
- We're only interested in a small, fixed subset
- Examples:
  - Music tagging (like ID3v2)
  - Financial protocols (like FIX)



# Solution

- Parse the interesting tags into a small array:

```
parser<title, album, artist> p(reader);
```

- Read them by key, calculating the array index at compile time:

```
out << "Title: " << p.at<title>();
```

```
out << "Album: " << p.at<album>();
```

```
p.at<bpm>(); // ERROR! unspecified tag
```

# Basic types

- **Values:** perhaps a view over received data

```
using view = std::string_view;
```

- **Tags:** nullable pairs (philosophically awkward)

```
struct tag {  
    explicit operator bool() const;  
    int key;  
    view value;  
};
```

- **Reader:** functor returning sequential tags

# The Parser

- Contains an array of values
- Initialised from a reader

```
while (tag t = reader()) {  
    int i = index(t.key);  
    if (i >= 0) values[i] = t.value;  
}
```

- Read by key

```
static_assert(index(key) >= 0);  
return values[index(key)];
```

# Gory details: Key sets

- A compile-time set of integer keys:

```
template <int... Keys> struct keyset {  
    static constexpr int keys[] {Keys...};  
};
```

- Operations, including

```
// gory details omitted  
template <class Keys> using sort = keyset<??>;
```

# Gory details: Finding the index

- Binary search in a sorted key-set's array

```
using sorted = sort<keyset<Keys...>>;
int first = 0, last = std::size(sorted::keys);
while (first != last) {
    int mid = first + (last-first)/2;
    if (sorted::keys[mid] == key)
        return mid;
    if (sorted::keys[mid] < key)
        first = mid+1;
    else
        last = mid;
}
return -1;
```

# Gory details: Sorting the keys

```
template <int Key, class Keys> struct prepend_  
template <int Key, class Keys> using prepend = typename prepend_<Key, Keys>::result;  
  
template <int Key, class Keys> struct prepend_ {using result = keyset<Key>;};  
template <int Key, int... Keys> struct prepend_<Key, keyset<Keys...>>  
    {using result = keyset<Key, Keys...>;};  
  
template <int Key, class Keys> struct remove_  
template <int Key, class Keys> using remove = typename remove_<Key, Keys>::result;  
  
template <int Key, class Keys> struct remove_ {using result = keyset<>;};  
template <int Key, int... Tail> struct remove_<Key, keyset<Key, Tail...>>  
    {using result = keyset<Tail...>;};  
template <int Key, int Head, int... Tail> struct remove_<Key, keyset<Head, Tail...>>  
    {using result = prepend<Head, remove<Key, keyset<Tail...>>>;};  
  
template <class Keys> struct min_  
template <int Single> struct min_<keyset<Single>> {static constexpr int result = Single;};  
template <int Head, int... Tail> struct min_<keyset<Head, Tail...>> {  
    static constexpr int tail = min_<keyset<Tail...>>::result;  
    static constexpr int result = Head < tail ? Head : tail;  
};  
template <class Keys> static constexpr int min = min_<Keys>::result;  
  
template <class Keys> struct sort_  
template <class Keys> using sort = typename sort_<Keys>::result;  
  
template <> struct sort_<keyset<>> {using result = keyset<>;};  
template <int... Keys> struct sort_<keyset<Keys...>> {  
    static constexpr int first = min<keyset<Keys...>>;  
    using result = prepend<first, sort<remove<first, keyset<Keys...>>>>;};  
};
```

The image features a pair of rich red theater curtains with vertical pleats and gold tassels on the sides. The curtains are drawn back, revealing a dark background. Centered on the curtains is the text "SLASH OR DASH" in a bold, white, sans-serif font, arranged in three lines.

**SLASH  
OR  
DASH**

C++



**- > EITHER //**

**LISP**

**(NAH)**

**BRAINF\*\*K**

**NO**

The background of the image consists of rich, red theater curtains with deep vertical pleats. The curtains are pulled back on both sides, revealing gold tassels at the ends. The lighting is dramatic, with the center of the stage area being brighter than the edges.

**SLASH  
OR  
DASH**

**CALC.EXE**

**NEITHER**





**THANKS!**