

Building Worlds

&

Feeling Great

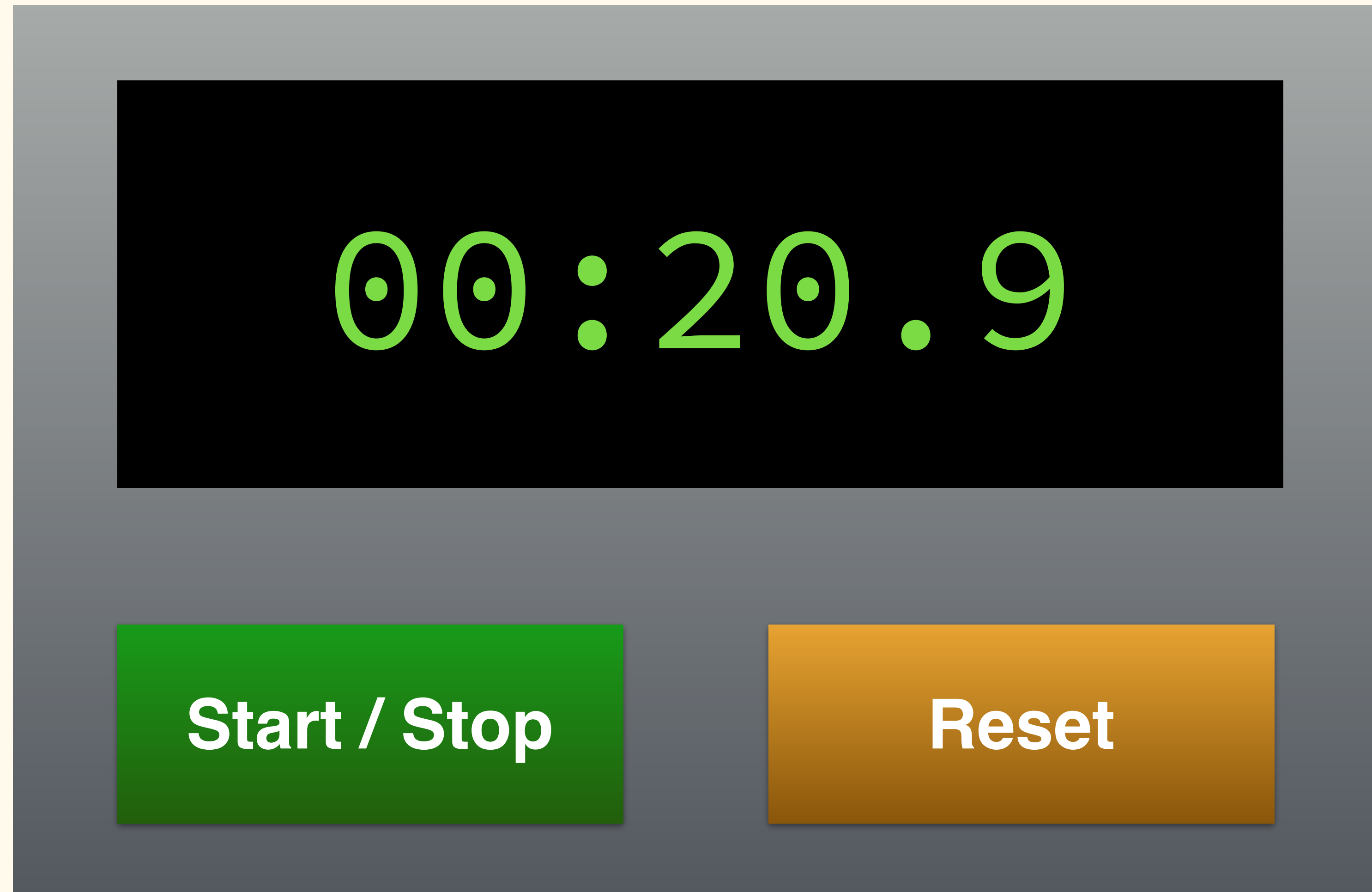
Brad Fults · @h3h

Head of Engineering Operations
Under Armour

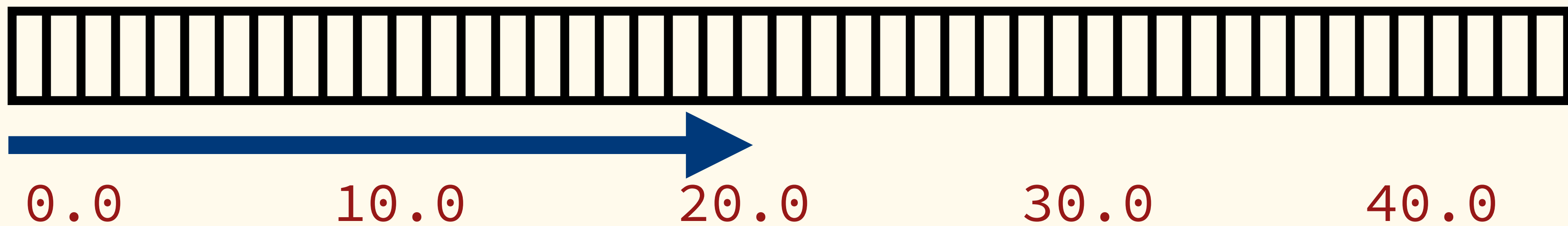
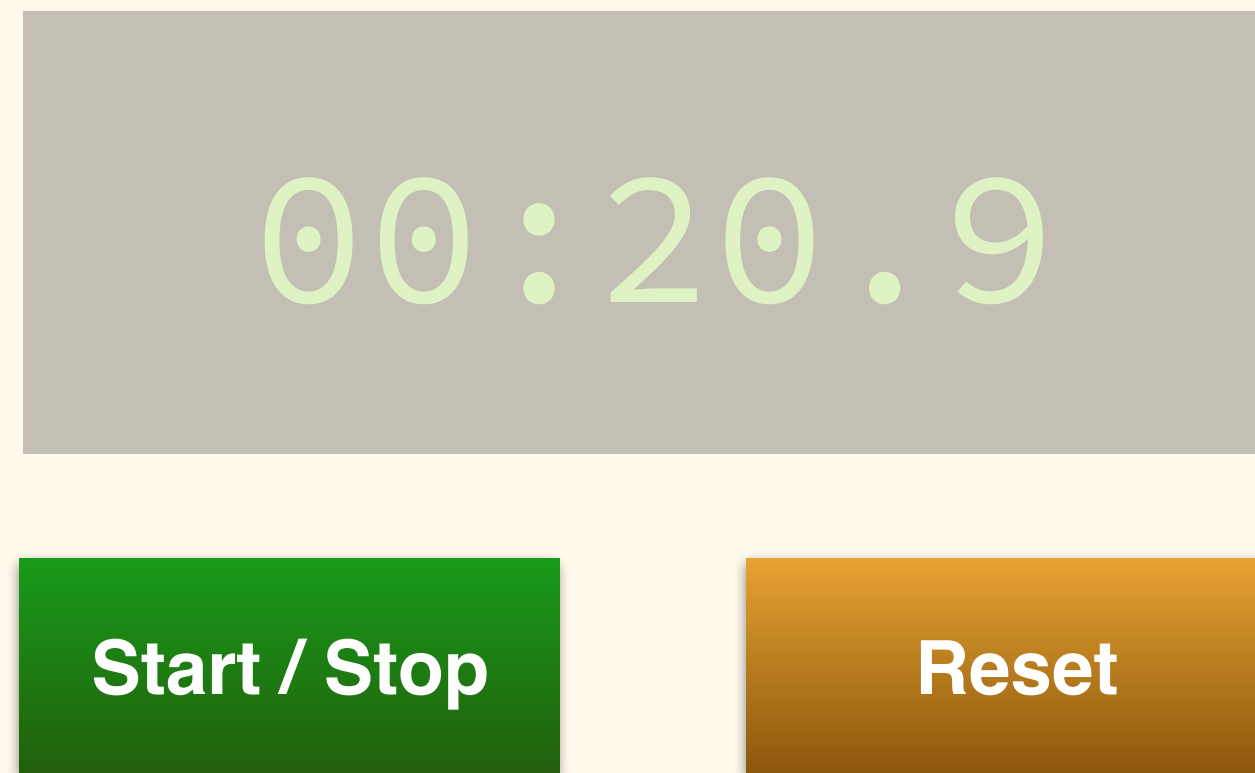
#buildingworlds

Interfaces

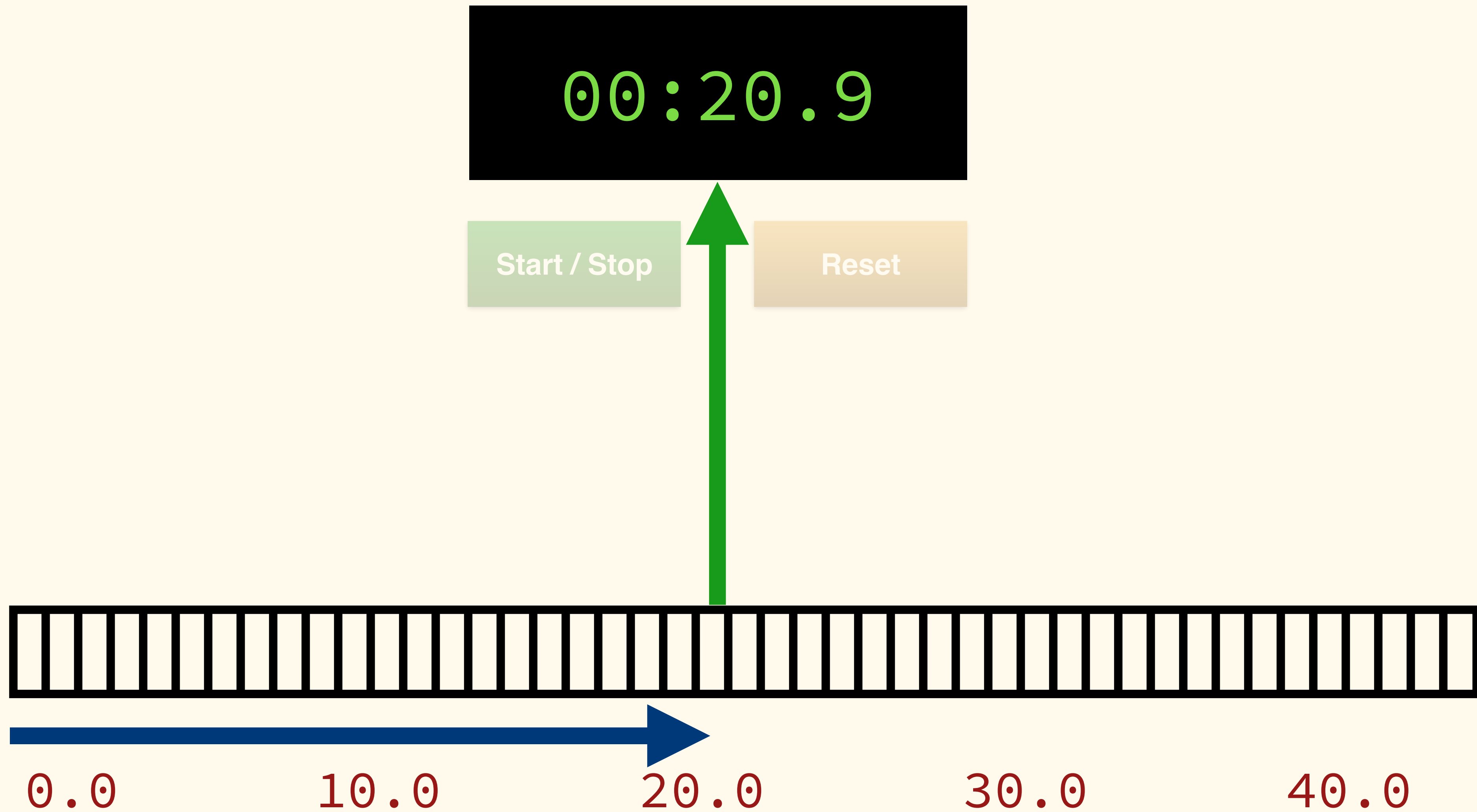
Interfaces



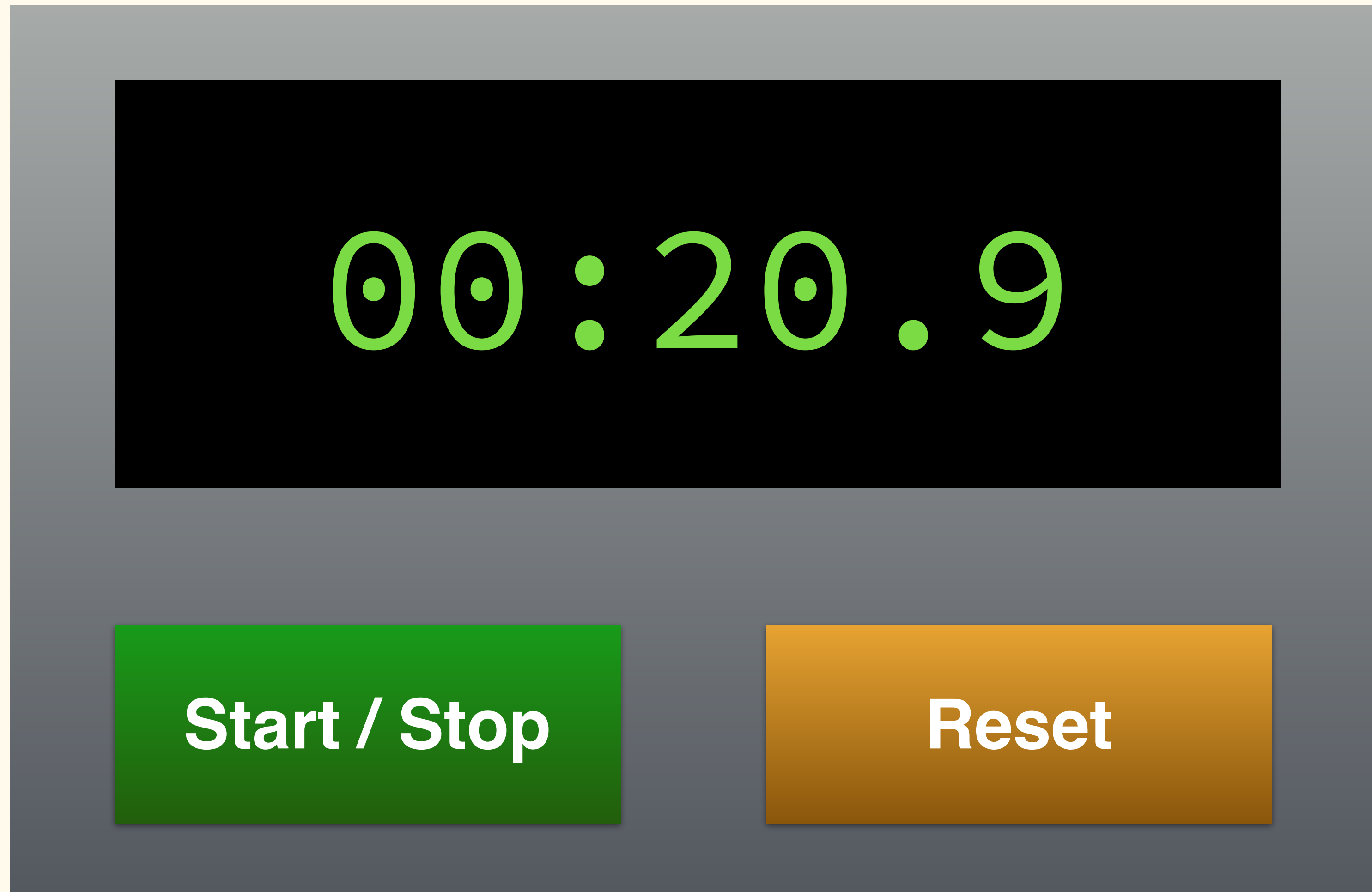
Interfaces



Interfaces



Interfaces



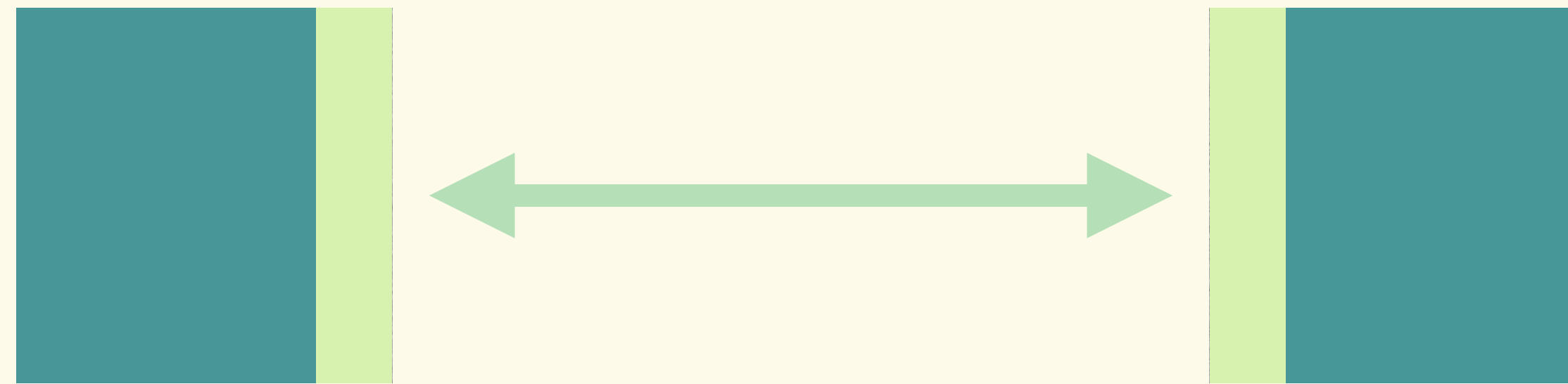
Interfaces

Stopwatch

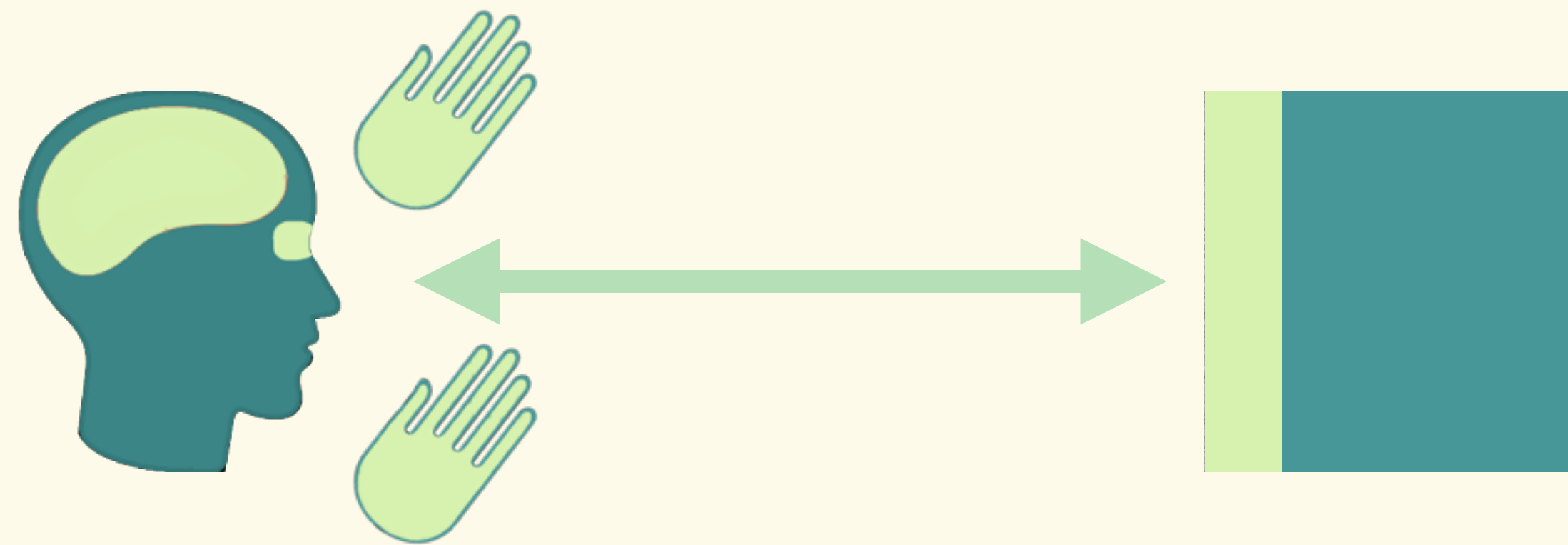
- start
- stop
- reset

- watch_time
- counting?

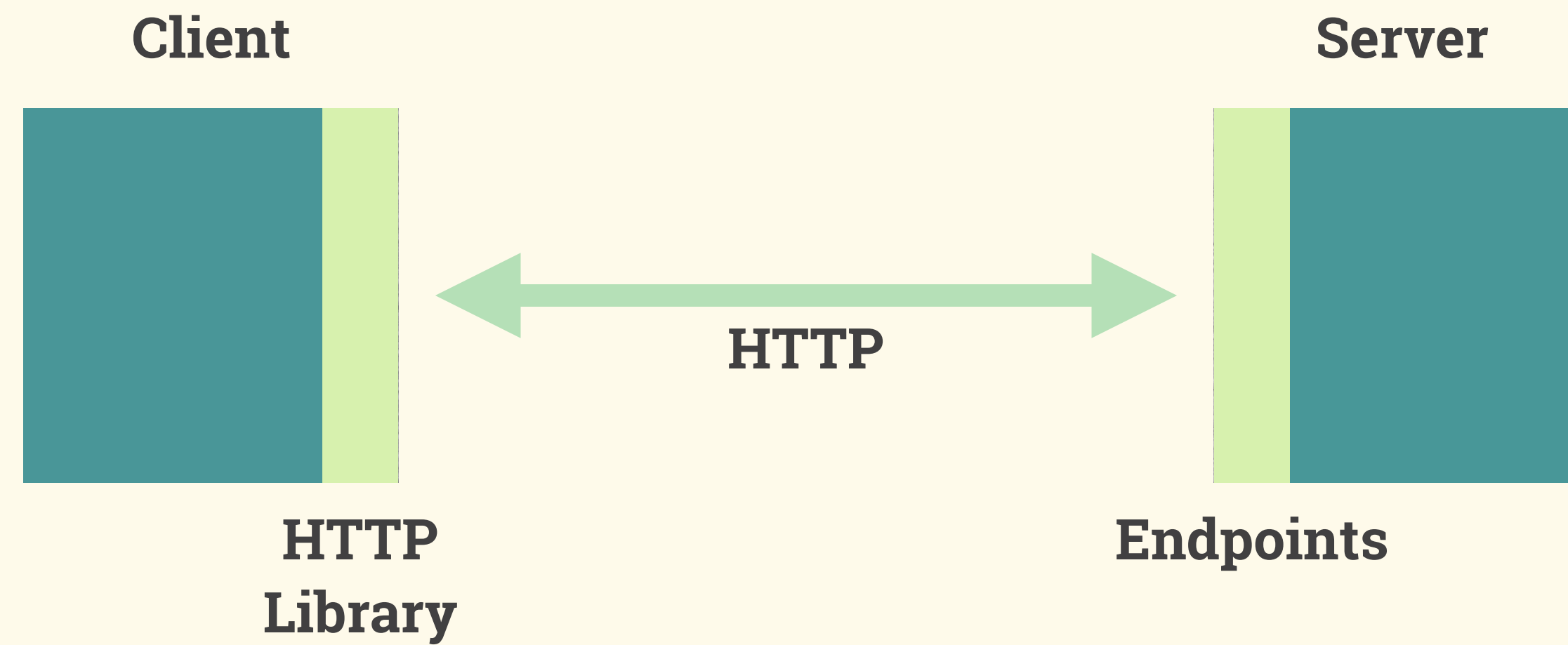
Interfaces



Interfaces

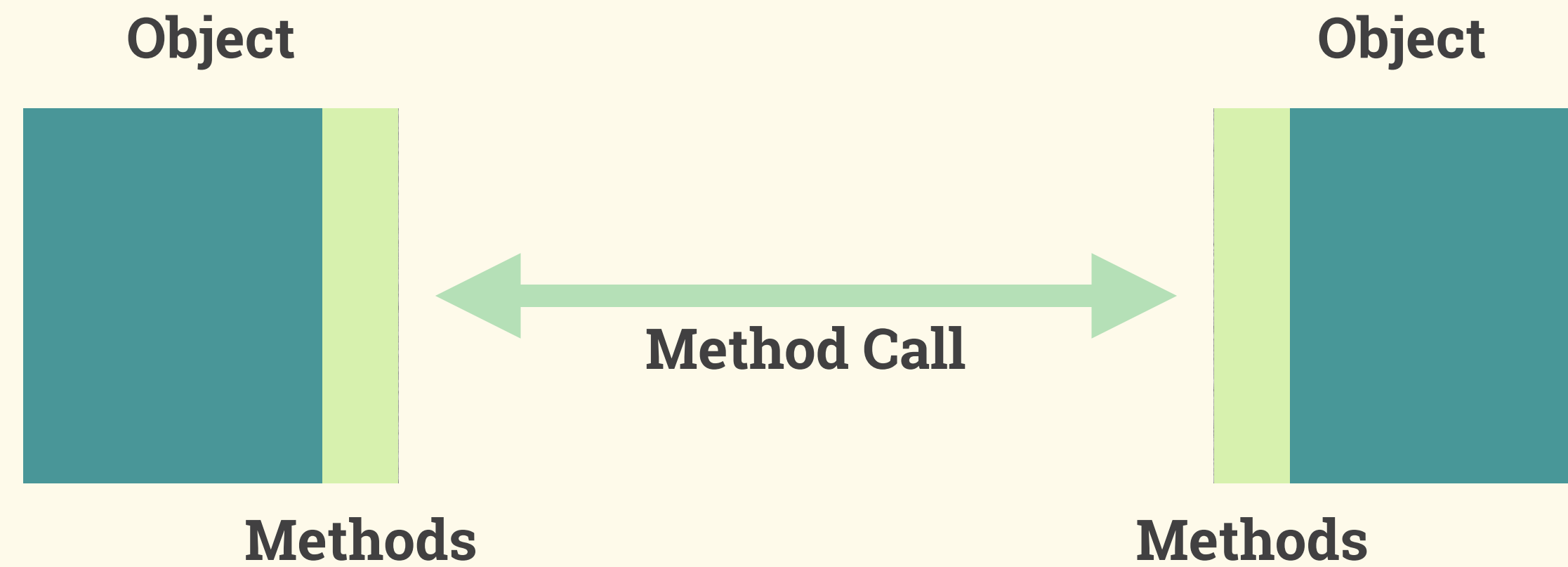


Interfaces



HTTP Interactions

Interfaces



**Object-Oriented
Programming**

Interfaces

```
java.util
```

```
public interface List<E>
```

- boolean add(E e)
- void clear()
- boolean contains(Object o)
- E get(int index)
- E remove(int index)
- int size()

Interfaces

Implementation

```
class Person

  def first_initial
    first_name.first.upcase
  end

  def last_initial
    last_name.first.upcase
  end

end
```

Interface

```
Person

- first_initial
- last_initial
```

Levels of Abstractions.io



Levels of Abstraction

Go to Mars

Abort



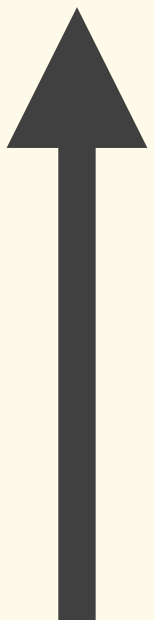
Levels of Abstraction

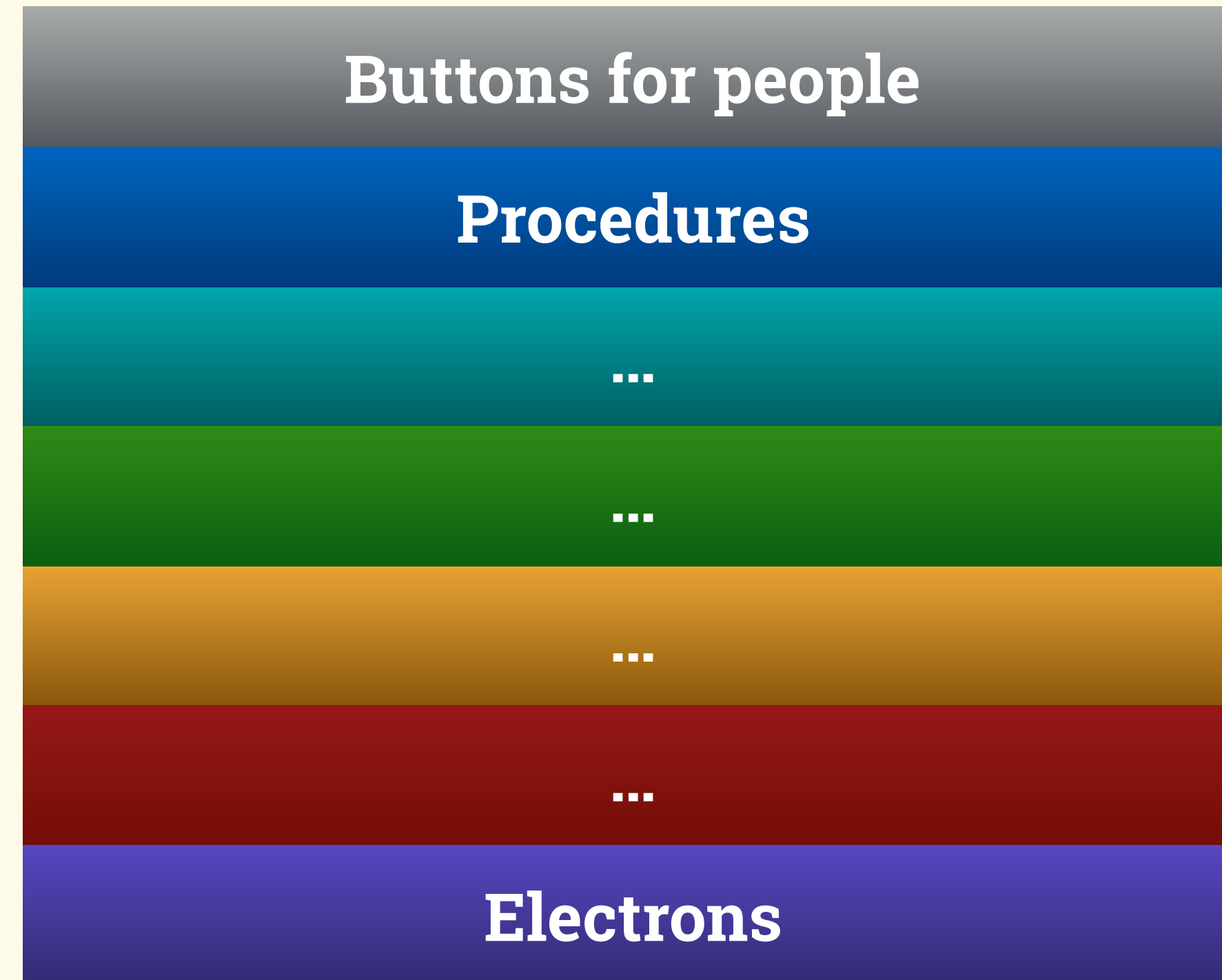
Go to Mars

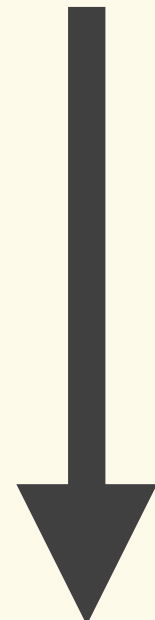
Abort

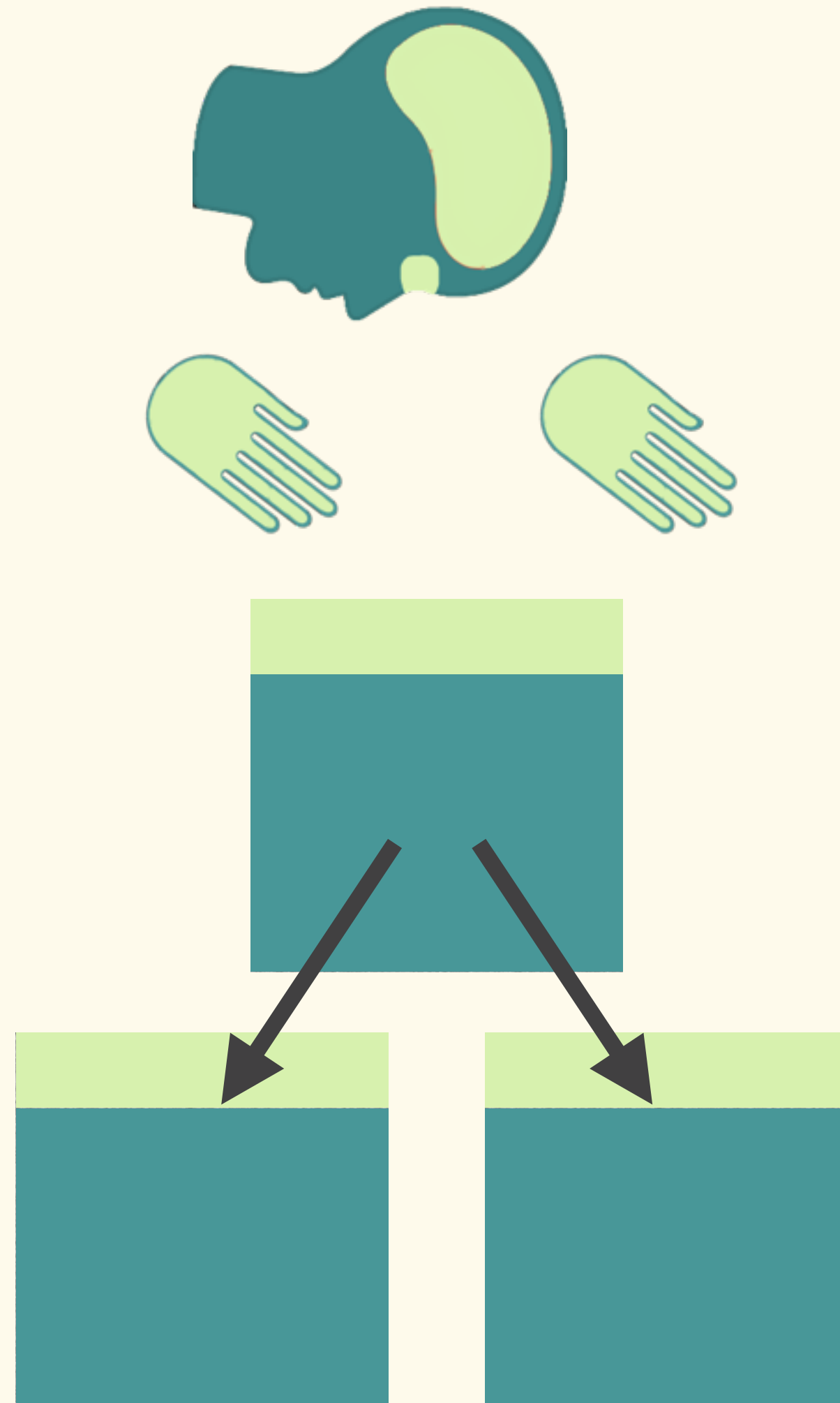


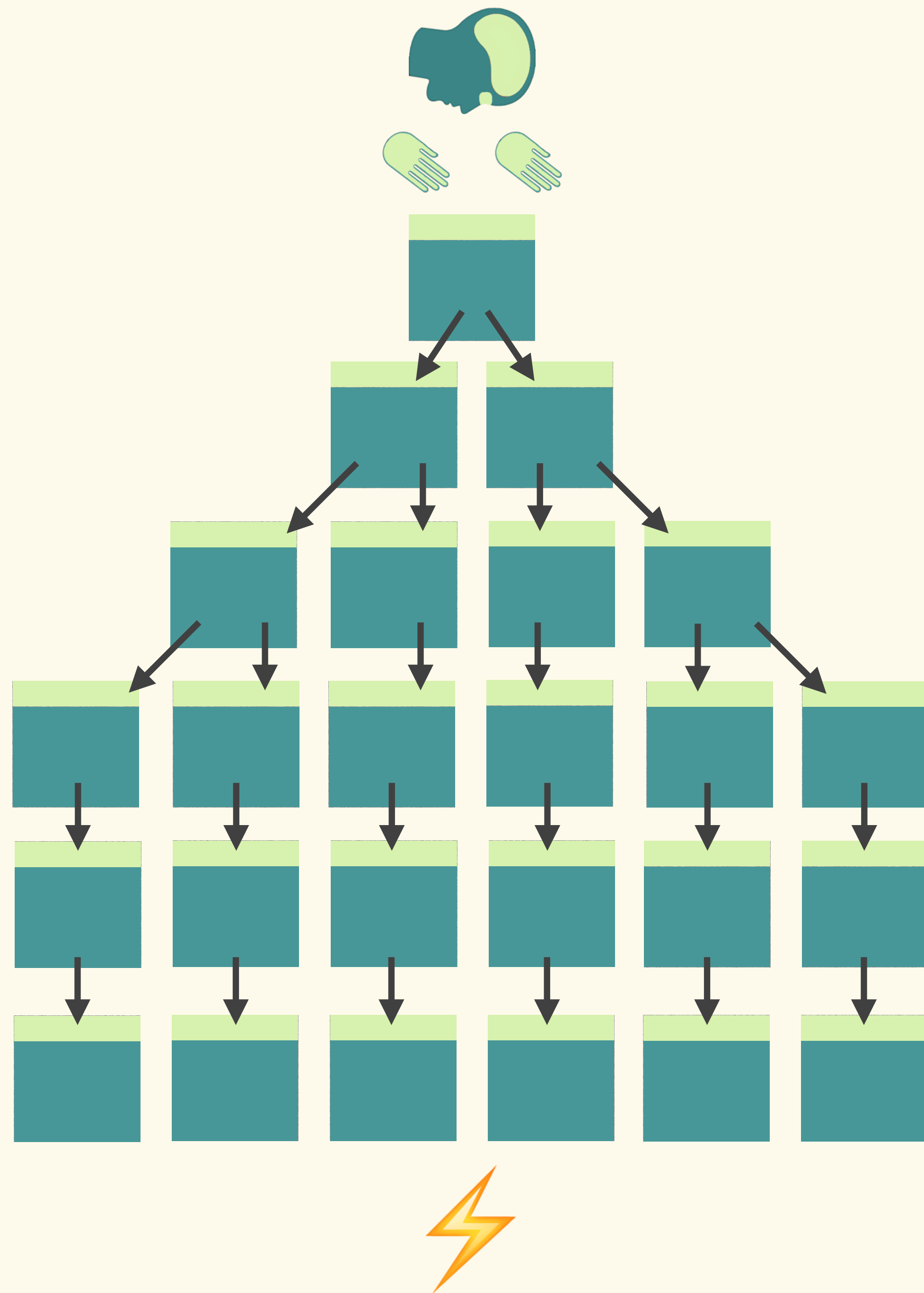
Levels of Abstraction


Higher



Lower






Levels of Abstraction

| | | | | | | | | | | |
|------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 0x051de510 | 01101001 | 01100101 | 00100100 | 01010100 | 01101000 | 01101101 | 01111010 | 11000101 | 01100101 | 00000000 |
| 0x051de520 | 01101001 | 01100101 | 00100100 | 01010100 | 01101000 | 01101101 | 01111010 | 11000101 | 01100101 | 00000000 |
| 0x051de530 | 01101001 | 01100101 | 00100100 | 01010100 | 01101000 | 01101101 | 01111010 | 11000101 | 01100101 | 00000000 |
| 0x051de540 | 01001000 | 01101001 | 00100000 | 01110100 | 01101000 | 01100101 | 01110010 | 01100101 | 00100001 | 00000000 |
| 0x051de550 | 01101001 | 01100101 | 00100100 | 01010100 | 01101000 | 01101101 | 01111010 | 11000101 | 01100101 | 00000000 |
| 0x051de560 | | | | | | | | | | |
| 0x051de570 | | | | | | | | | | |
| 0x051de580 | | | | | | | | | | |
| 0x051de590 | | | | | | | | | | |
| 0x051de5a0 | 01101001 | 01100101 | 00100100 | 01010100 | 01101000 | 01101101 | 01111010 | 11000101 | 01100101 | 00000000 |
| 0x051de5b0 | 01101001 | 01100101 | 00100100 | 01010100 | 01101000 | 01101101 | 01111010 | 11000101 | 01100101 | 00000000 |
| 0x051de5c0 | 01101001 | 01100101 | 00100100 | 01010100 | 01101000 | 01101101 | 01111010 | 11000101 | 01100101 | 00000000 |

```
char *str;  
str = malloc(32 * sizeof(char));  
strcpy(str, "Hi there!");
```

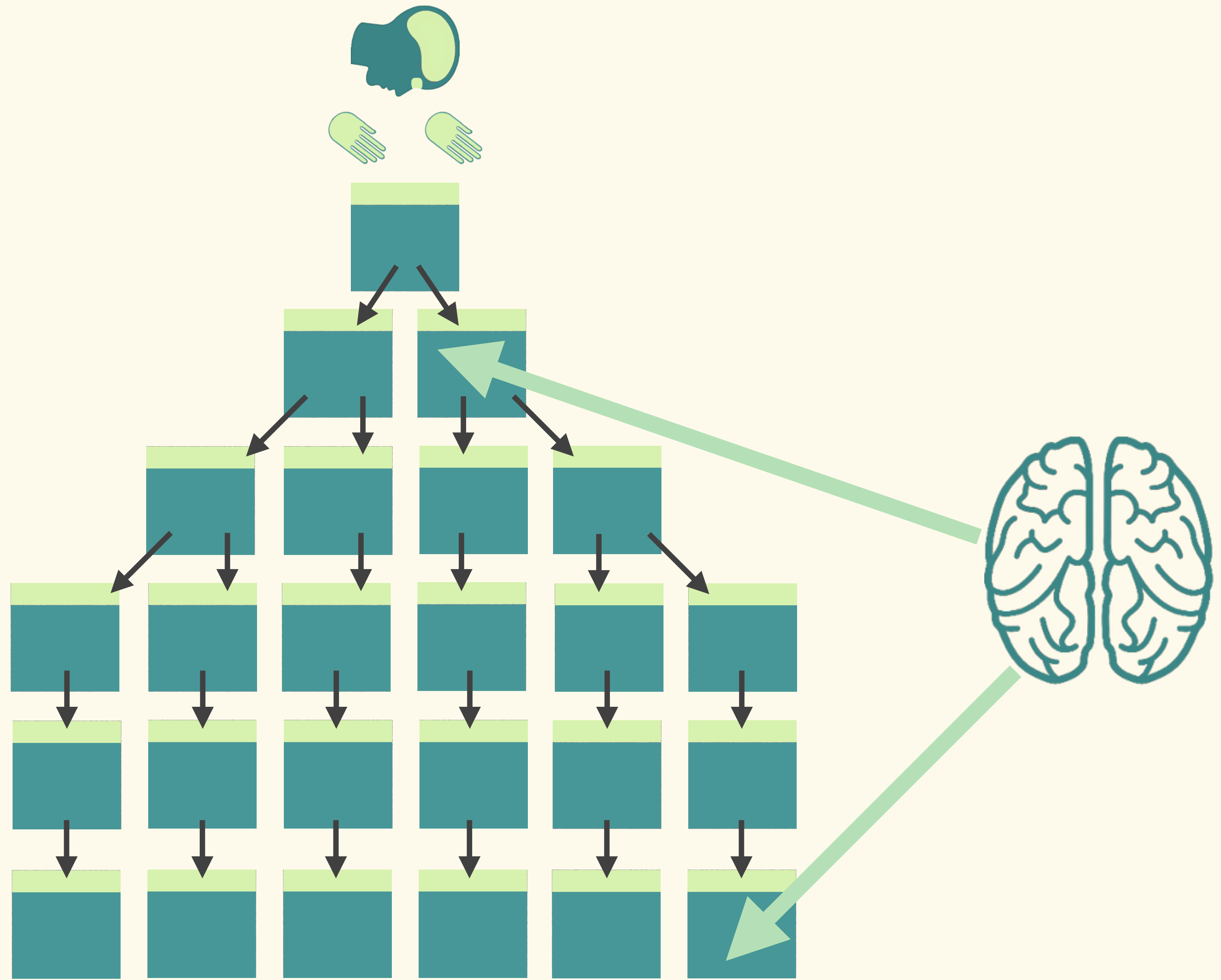
Levels of Abstraction

```
current_user.update!(  
  greeting: params['greeting']  
)
```

Higher

```
char *form;          /* 8 bytes */  
long form_size;      /* 8 bytes */  
char mode;           /* 1 byte  */
```

Lower



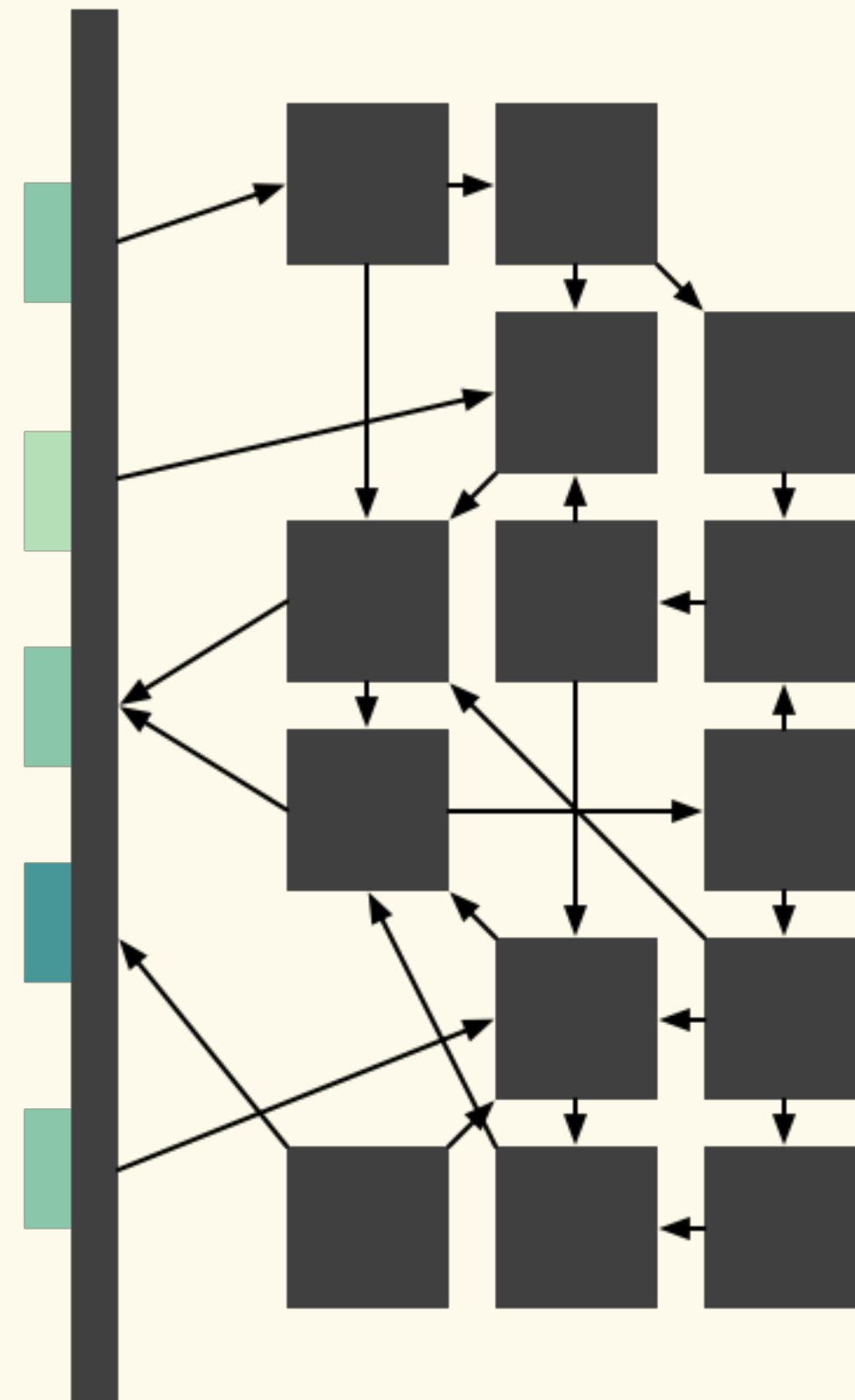
Building Worlds

Building Worlds

↑
Complexity



↑
Difficulty



Interface

World

Interfaces

Implementation

```
class Person

  def first_initial
    first_name.first.upcase
  end

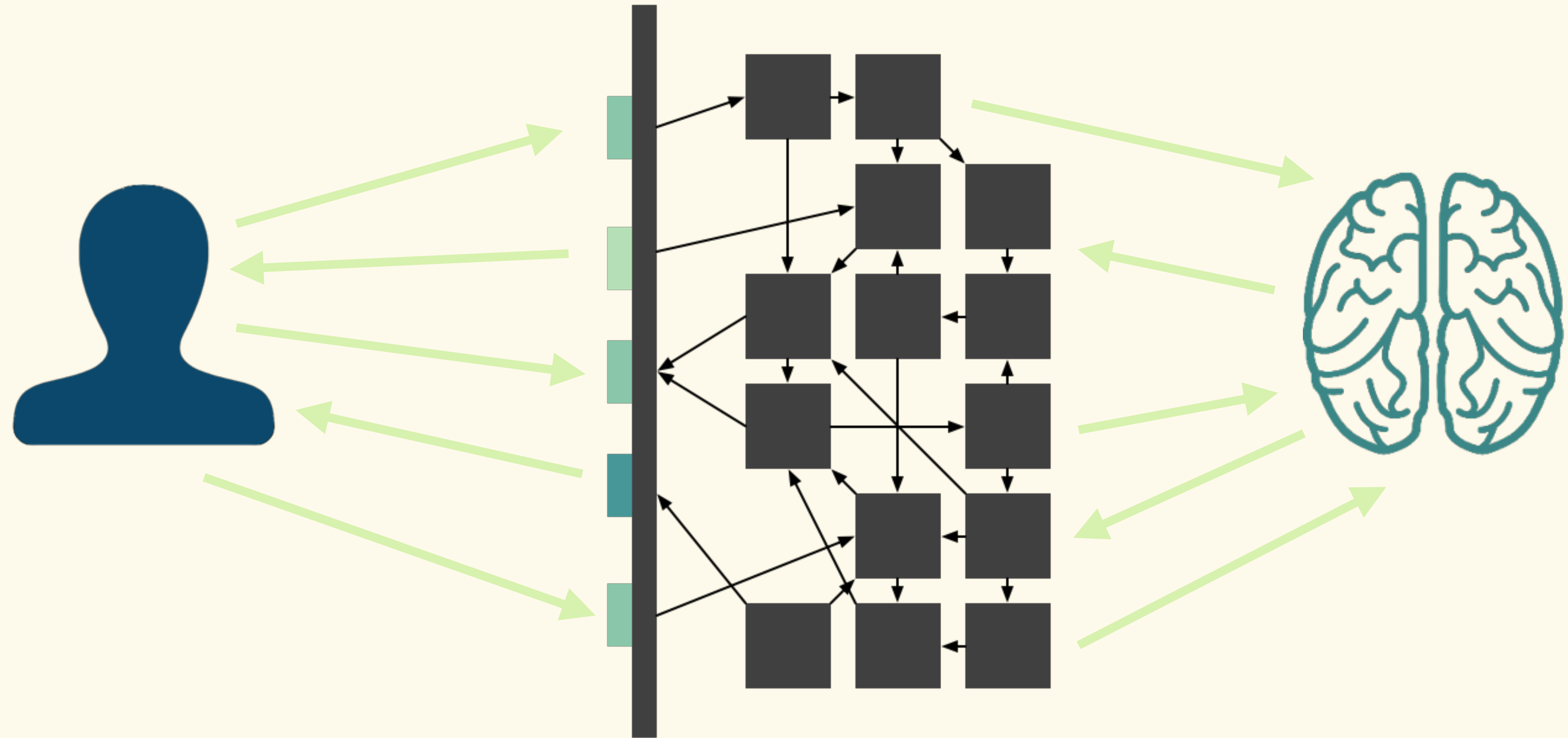
  def last_initial
    last_name.first.upcase
  end

end
```

Interface

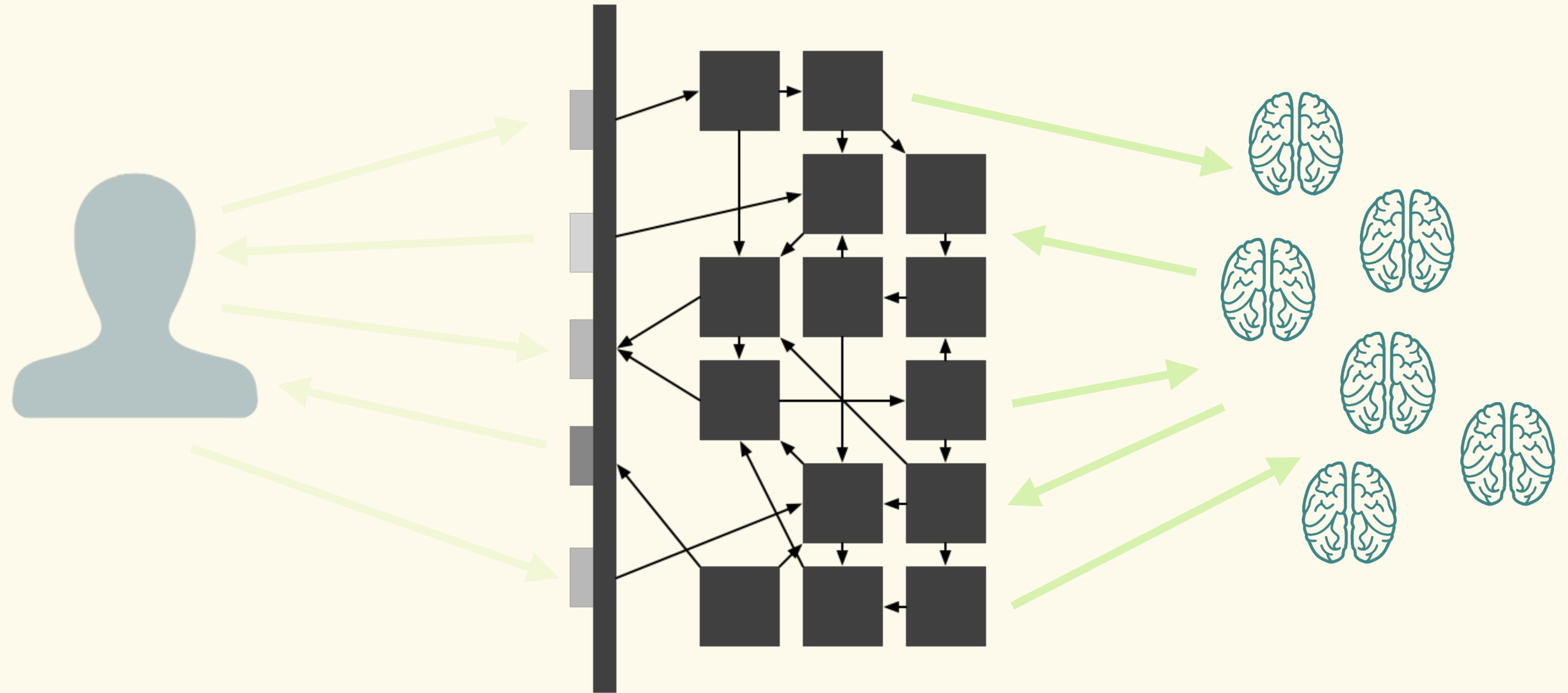
```
Person

- first_initial
- last_initial
```

User Interface Design

World **Building**



User Interface Design

World Building

Building Worlds

- 1. It's interfaces all the way down**
- 2. A good interface works at one level of abstraction**
- 3. Coupling is expensive forever**
- 4. Systems are for humans first, machines second**

It's interfaces all
the way down

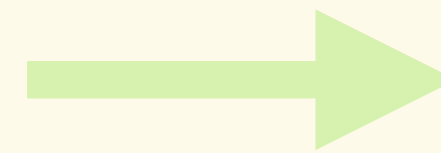


It's interfaces all the way down

00:20.9

Start / Stop

Reset



Stopwatch

- start
- stop
- reset
- watch_time
- counting?

Buttons

Procedures

...

...

...

...

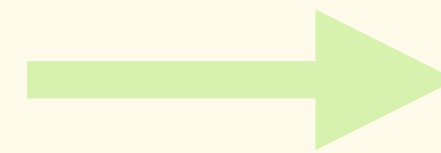
Electrons

It's interfaces all the way down

Stopwatch

- start
- stop
- reset

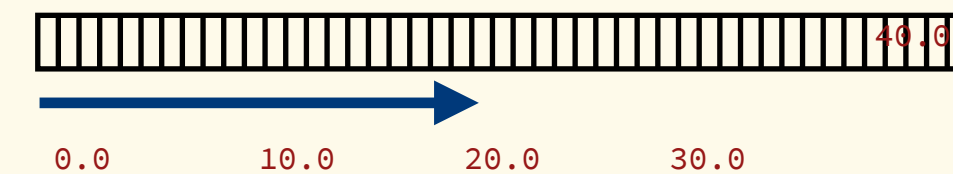
- watch_time
- counting?



Clock

- tick

- value



Buttons

Procedures

...

...

...

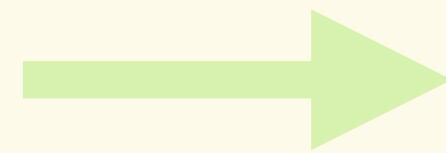
...

Electrons

It's interfaces all the way down

Clock

- tick
- value



```
# CLOCK_GETRES(2)
```

```
int clock_getres(  
    clockid_t clk_id,  
    struct timespec *res);
```

```
int clock_gettime(  
    clockid_t clk_id,  
    struct timespec *tp);
```

```
int clock_settime(  
    clockid_t clk_id,  
    const struct timespec *tp);
```

Buttons

Procedures

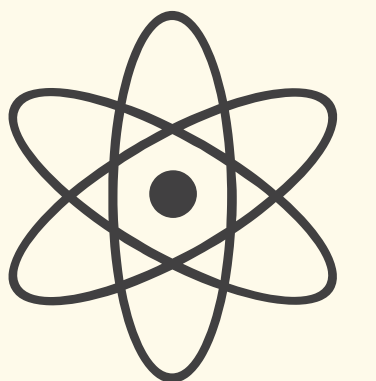
...

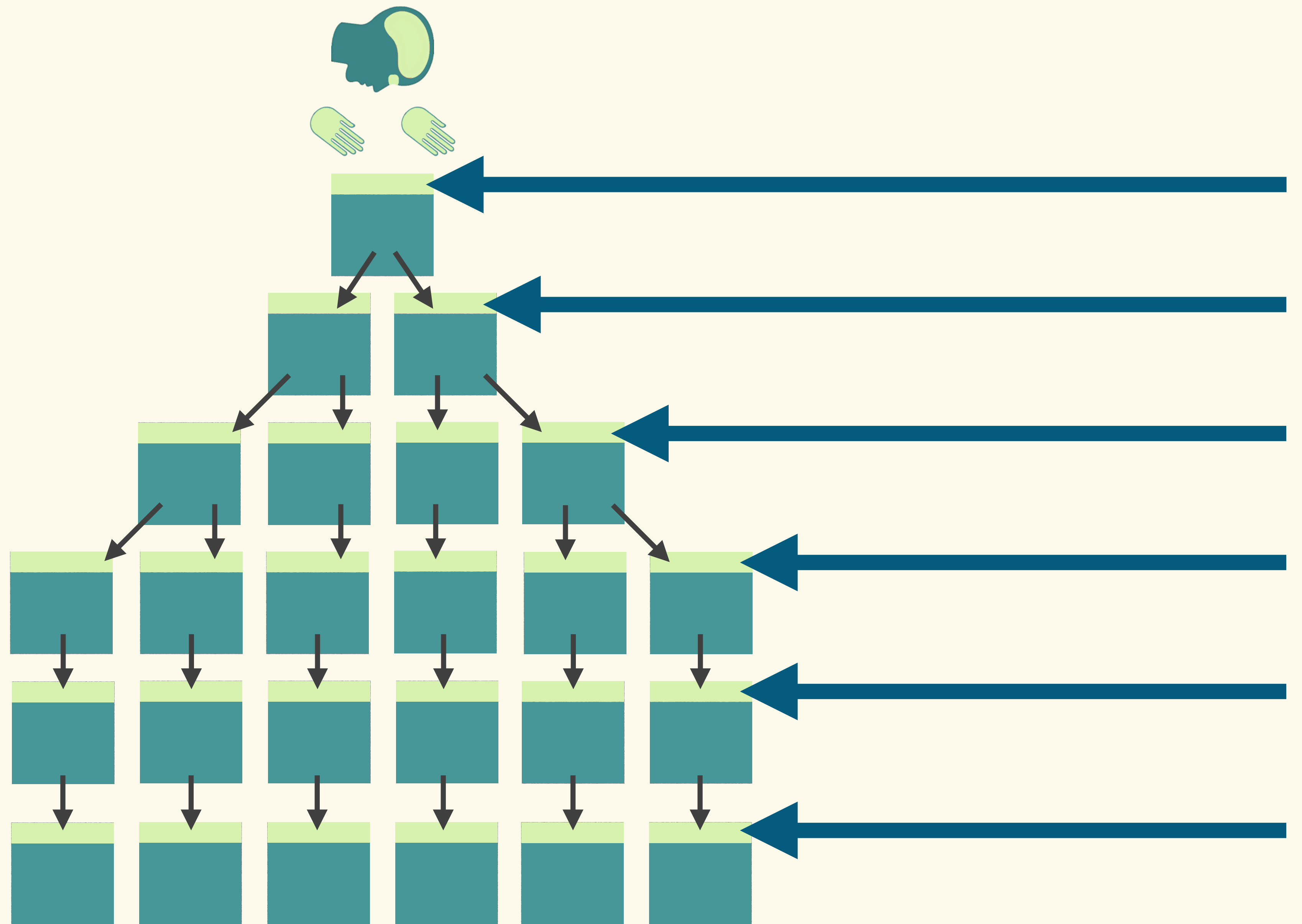
...

...

...

Electrons





It's interfaces all the way down

- There are interfaces at many levels of abstraction
- Each of these interfaces is a chance to be intentional
- These are **opportunities!**

A good interface works at
one level of abstraction

A good interface works at one level of abstraction

Go to Mars

Abort

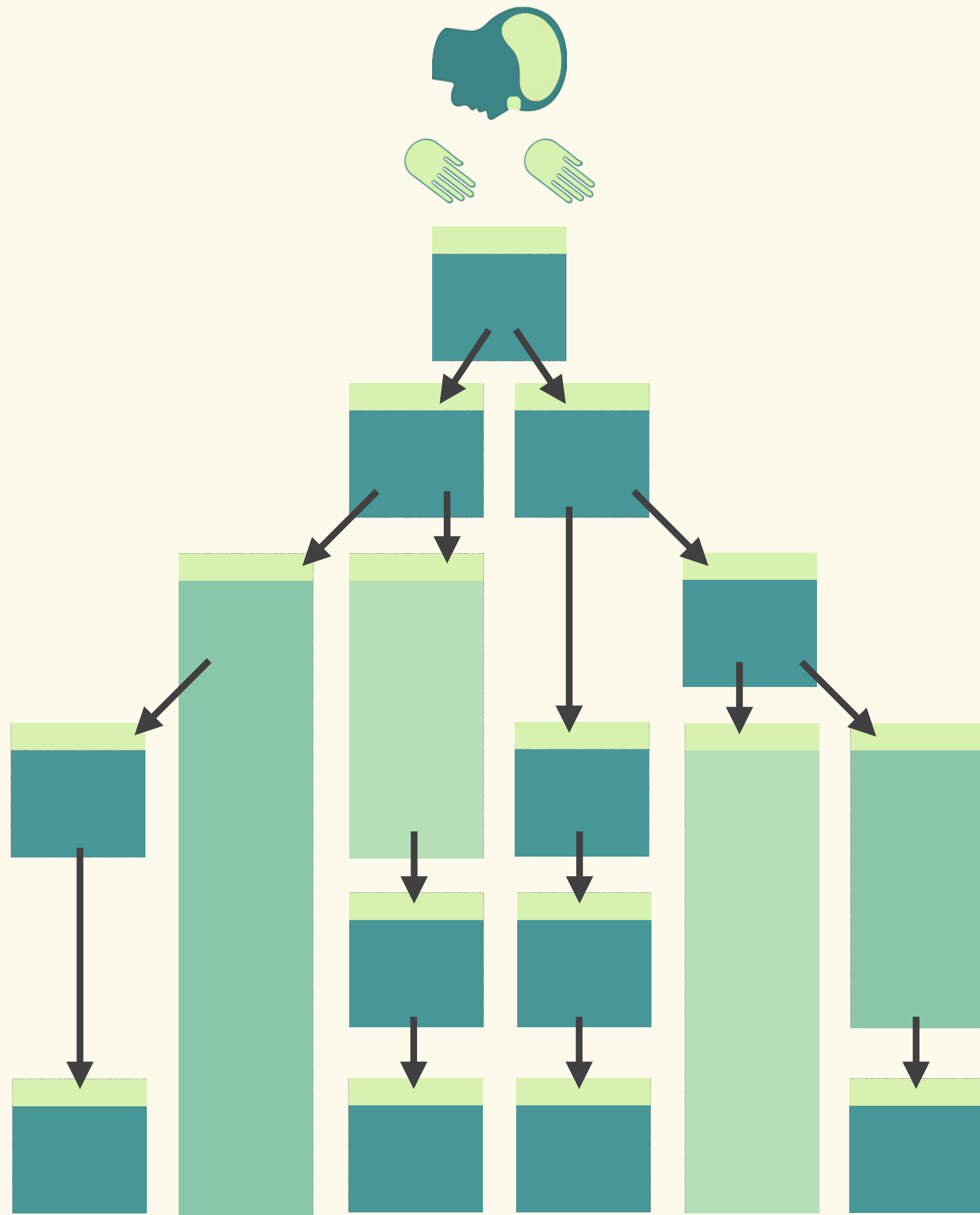
A good interface works at one level of abstraction

Go to Mars

| | | | |
|------------------------|-----------------------|-------------------------------|-------------------------|
| Shut down engine | Notify Earth of abort | Create guidance plan to Earth | Maneuver with thrusters |
| Get guidance plan | Notify Earth of plan | Fire engine | Get progress |
| Send progress to Earth | Deploy parachute | Unlock crew door | |

A good interface works at one level of abstraction

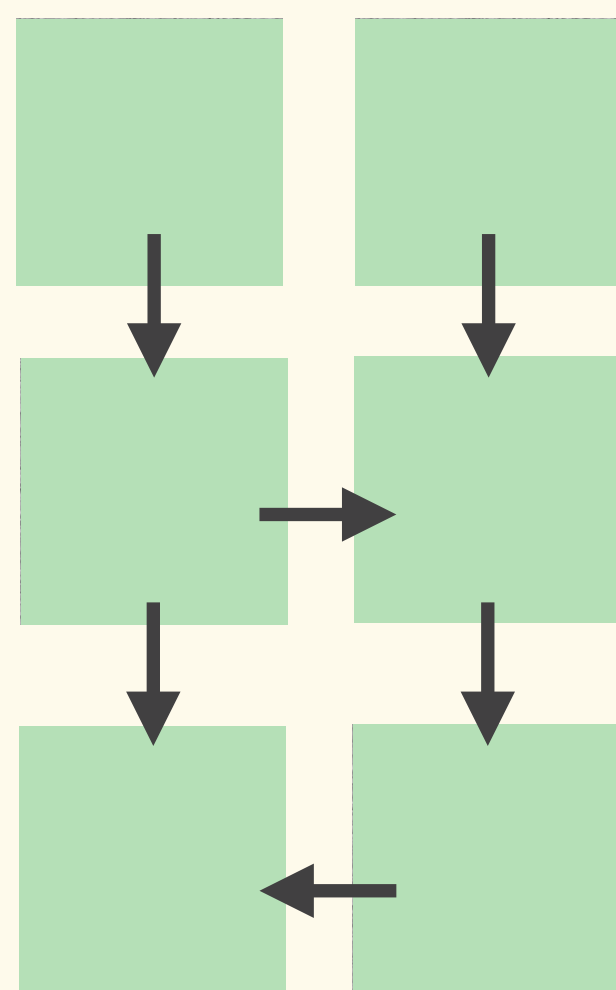


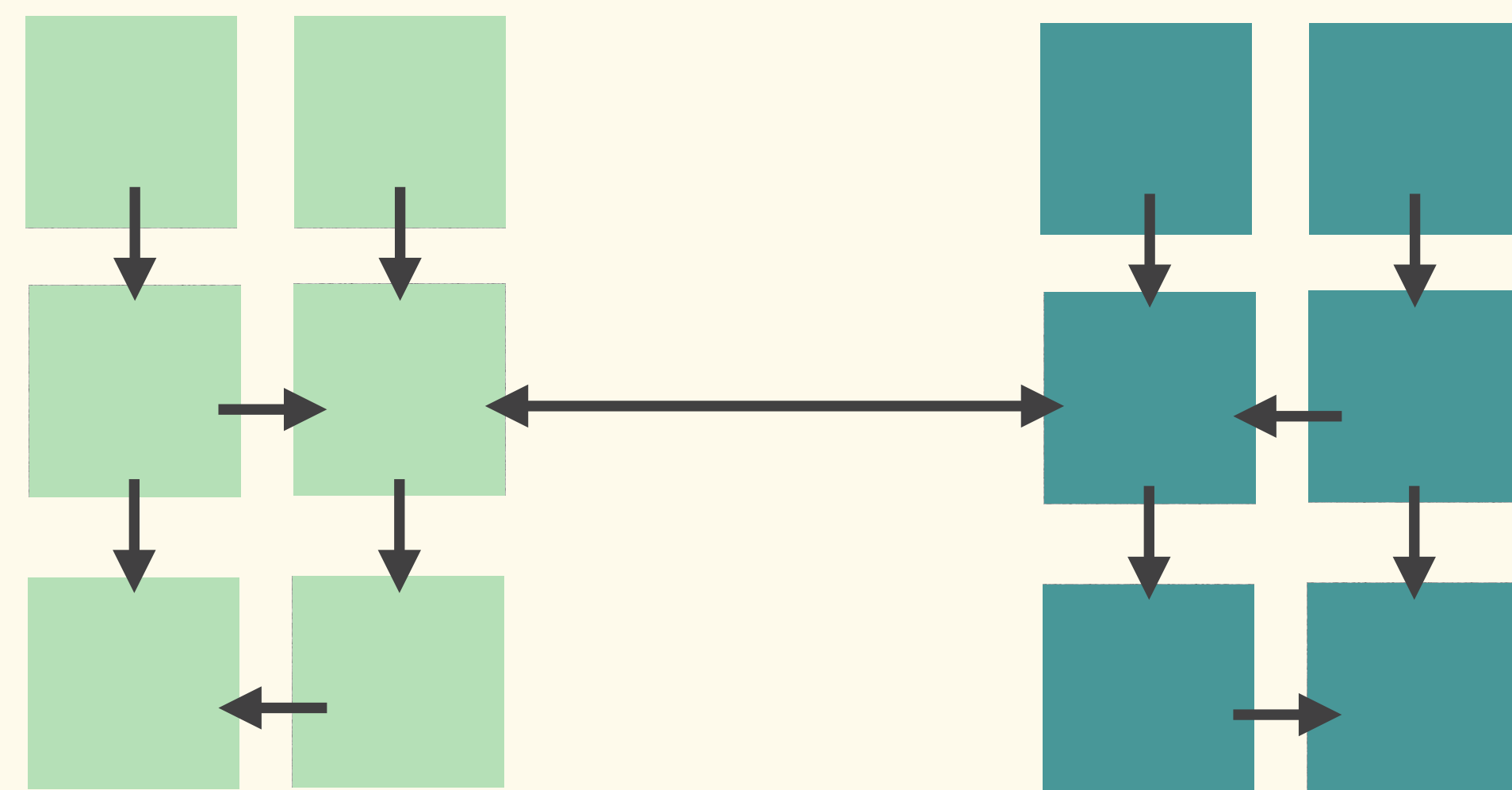


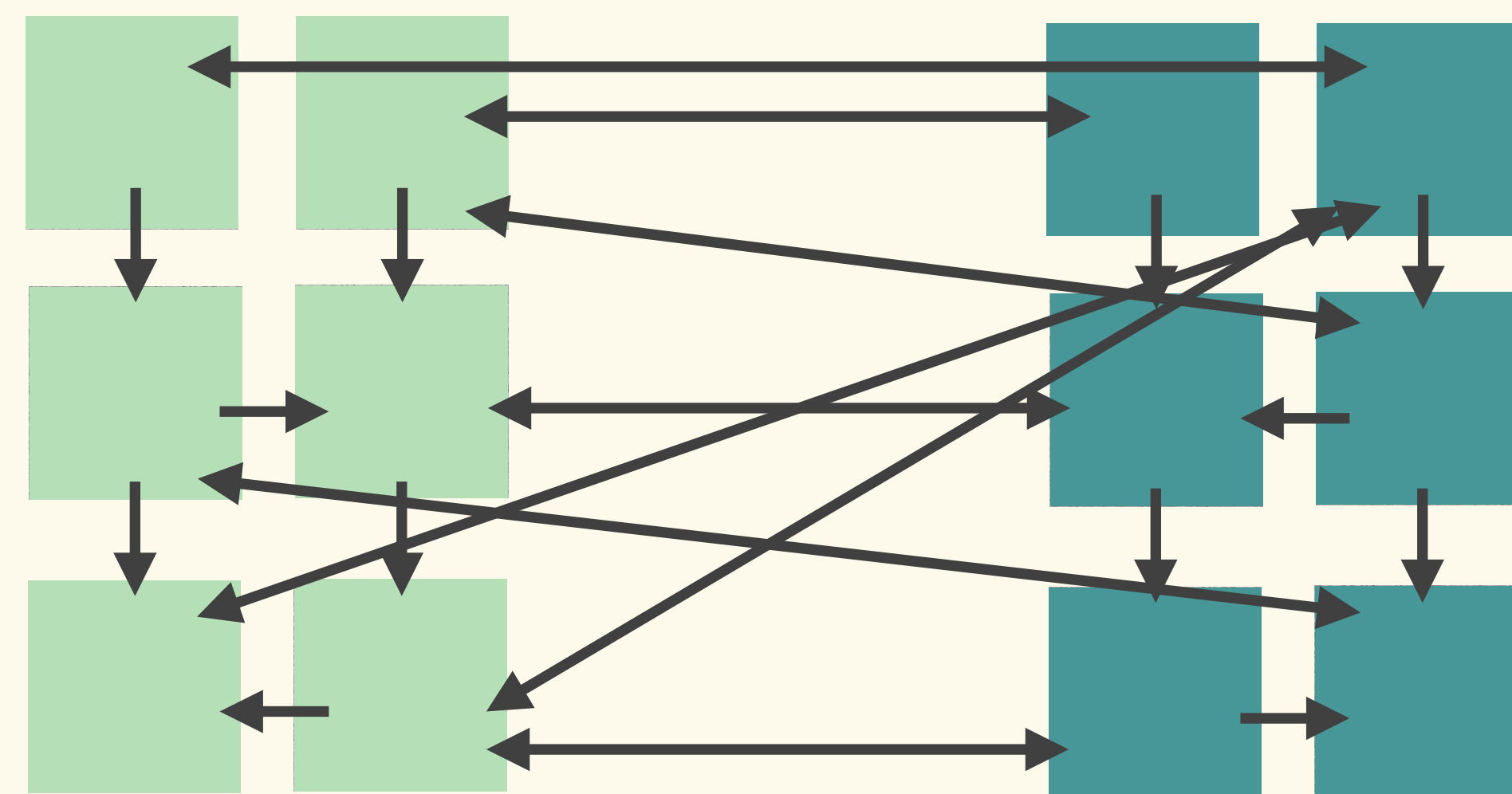
A good interface works at one level of abstraction

- **Interfaces are better within one level of abstraction**
- **Complexity means more training & experience needed**
- **We can use these *insights* to improve our interfaces!**

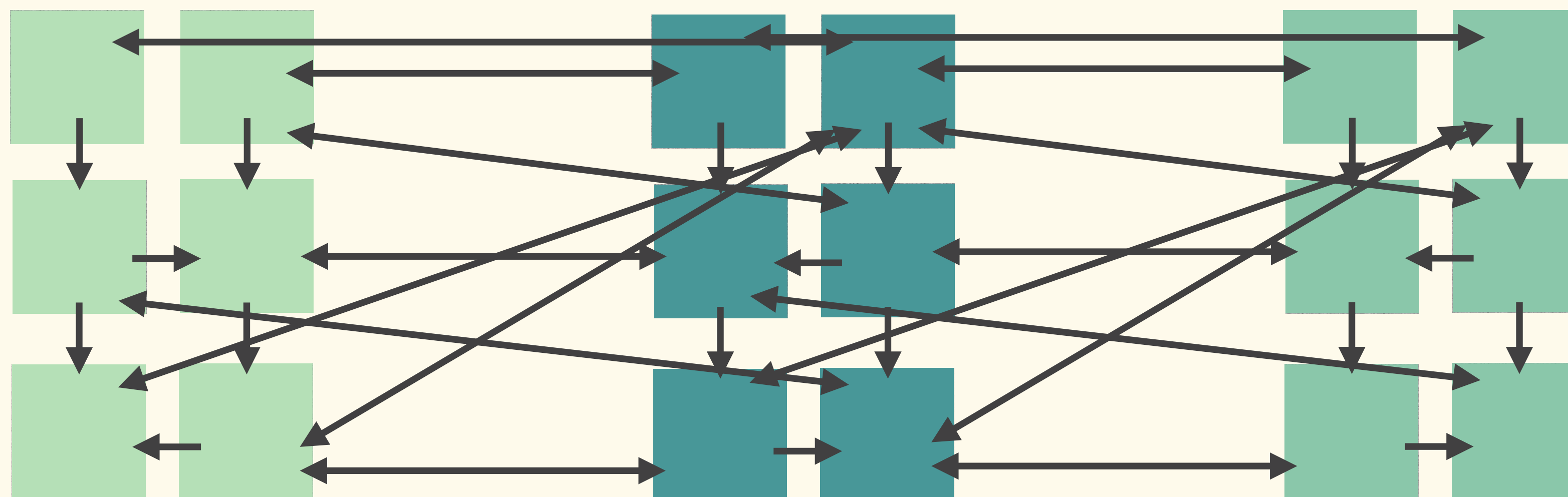
Coupling is expensive
forever



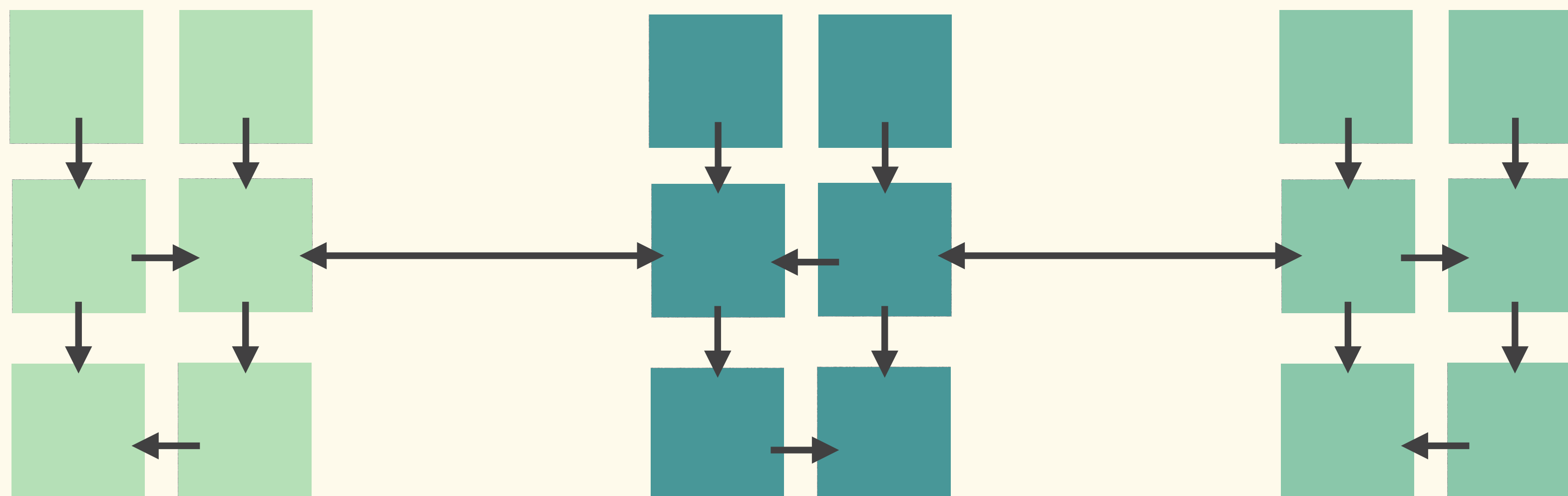




Tight Coupling



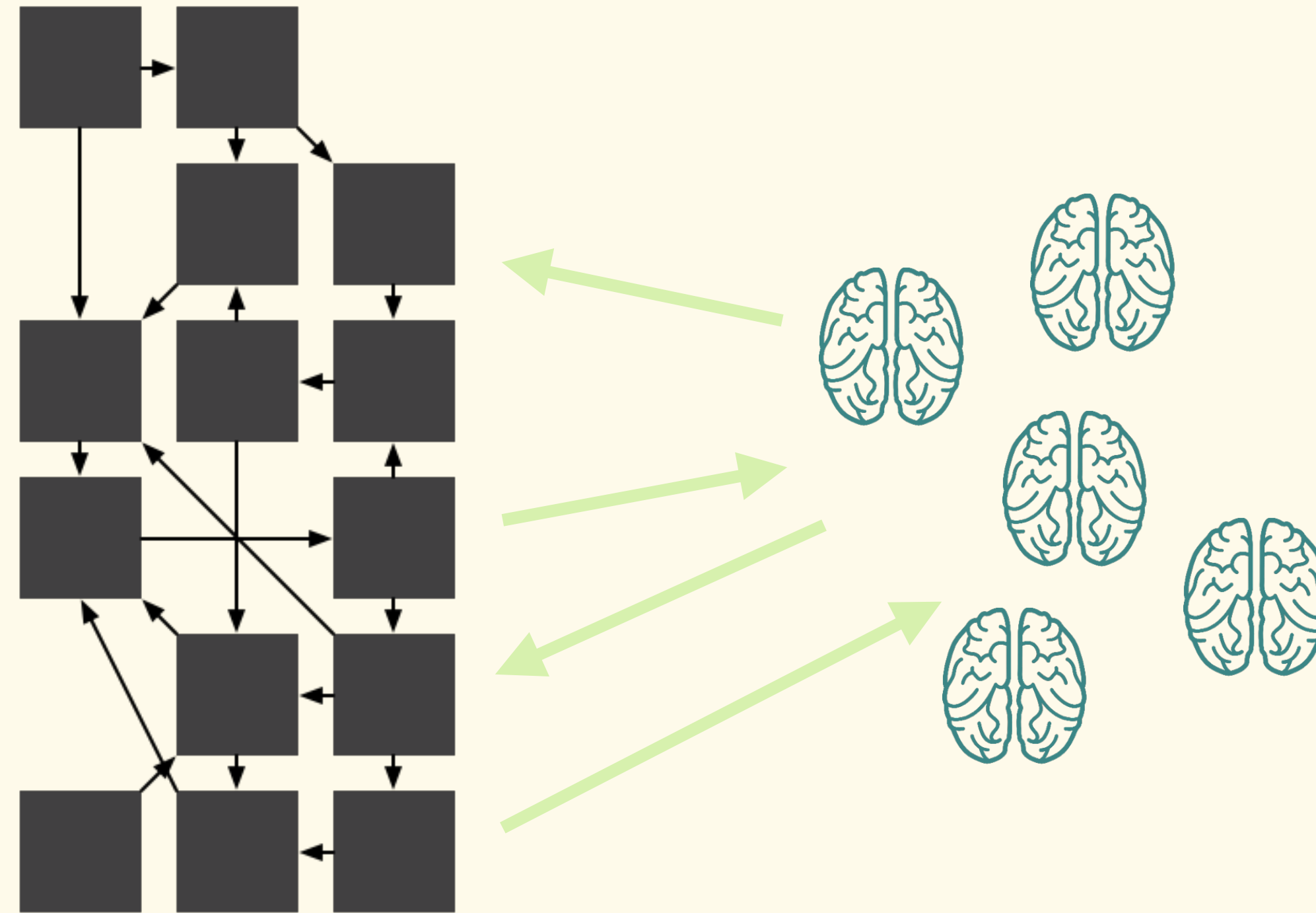
Tight Coupling



Loose Coupling

Systems are for humans
first, machines second

Systems are for humans first, machines second



Systems are for humans first, machines second

- **Attention span**
- **Intensity of focus**
- **Discontinuous attention**
- **Different desires**
- **Mistakes**
- **Shenanigans**

Systems are for humans first, machines second

- **Employ empathy**
- **Strive for simplicity**
- **Think about usability**
- **Plan use cases**
- **Measure task success**
- **Get feedback!**

Systems are for humans first, machines second

- **Hard work**

Feeling Great

Feeling Great

- **Easy to understand**
- **Simple changes have obvious results**
- **Builds confidence in tending the world**
- **Inspires happiness & empowerment**

Easy to understand

Easy to understand

- **Who are these people? What do they do?**
- **How do things work in this world?**
- **What's going on right now in the story?**

Easy to understand

- **Task: Add a new chapter to the middle of the story**
(Hint: Modify a part of the working system)
 - **Read the intro guide**
(Hint: Read the documentation)
 - **Understand the scene**
 - **Try changing something**
 - **Repeat until it works**

Easy to understand

- **Good documentation**
 - **Concise**
 - **Common use cases considered**
 - **Good examples with links to further reading**
 - **Clearly defined & explained jargon**

Easy to understand

- **Simple mechanics**
- **Allows focus on one thing at a time**
- **Doesn't hurt my brain**
- **Fits with its domain**

“But some stories, small, simple ones
about setting out on adventures or
people doing wonders, tales of miracles
and monsters, have outlasted all the
people who told them...”

—Johnny Appleseed

Neil Gaiman

Easy to understand

- Simple mechanics
- Allows focus on one thing at a time
- Doesn't hurt my brain
- Fits with its domain

Easy to understand

```
func Pipe() (*PipeReader, *PipeWriter)
```

```
client := &http.Client{  
    CheckRedirect: redirectPolicyFunc,  
}
```

```
resp, err := client.Get("http://example.com")
```

Three green arrows originate from the bottom left area. One arrow points diagonally upwards and to the right, ending near the 'client' variable. A second arrow points diagonally upwards and to the right, ending near the '*PipeReader' parameter. A third arrow points diagonally upwards and to the right, ending near the '*PipeWriter' parameter.

Easy to understand

- Simple mechanics
- Allows focus on one thing at a time
- Doesn't hurt my brain
- Fits with its domain

Naming

Naming

```
FileUtils.mkdir_p(dir, opts)
FileUtils.rmdir(dir, opts)
FileUtils.ln_s(old, new, opts)
FileUtils.cp_r(src, dest, opts)
FileUtils.rm_rf(list, opts)
FileUtils.chmod_R(mode, list, opts)
FileUtils.chown_R(user, group, list, opts)
```

Fitness for domain

Naming

- **Nouns**
 - **Directories**
 - **Links**
 - **Files**
 - **File Attributes**
- **Verbs**
 - **Create**
 - **Copy**
 - **Move**
 - **Delete**

Fitness for domain

Naming

Directories

```
FileUtils.copy_directory(...)
FileUtils.create_directory(...)
FileUtils.delete_directory(...)
FileUtils.move_directory(...)
```

Links

```
FileUtils.create_link(...)
FileUtils.delete_link(...)
```

Files

```
FileUtils.copy_file(...)
FileUtils.move_file(...)
FileUtils.delete_file(...)
```

File Attributes

```
FileUtils.change_file_attributes(...)
FileUtils.change_file_owner(...)
```

Fitness for domain

Naming

Directories

```
Directory.create(...)
Directory.copy(...)
Directory.move(...)
Directory.delete(...)
```

Files

```
File.copy(...)
File.move(...)
File.delete(...)
```

Links

```
Link.create(...)
Link.delete(...)
```

File Attributes

```
File.change_attributes(...)
File.change_owner(...)
```

Fitness for domain

Naming

Directory.**new**(...)
Directory.**copy**(...)
Directory.**move**(...)
Directory.**delete**(...)

Link.**create**(...)
File.**duplicate**(...)
File.**transmogrify**(...)
Link.**destroy**(...)
File.**dismantle**(...)

Coherence

Naming

Sad things

Directory.new_directory(...)

User.process(...)

store_order.user_id # => “DAL-312348”

employee.user_id # => “aa7eb462-af83”

Descriptiveness

Naming

POST /authors

GET /authors/123/articles

POST /articles

PATCH /articles/321

Organization

Simple changes have
obvious results

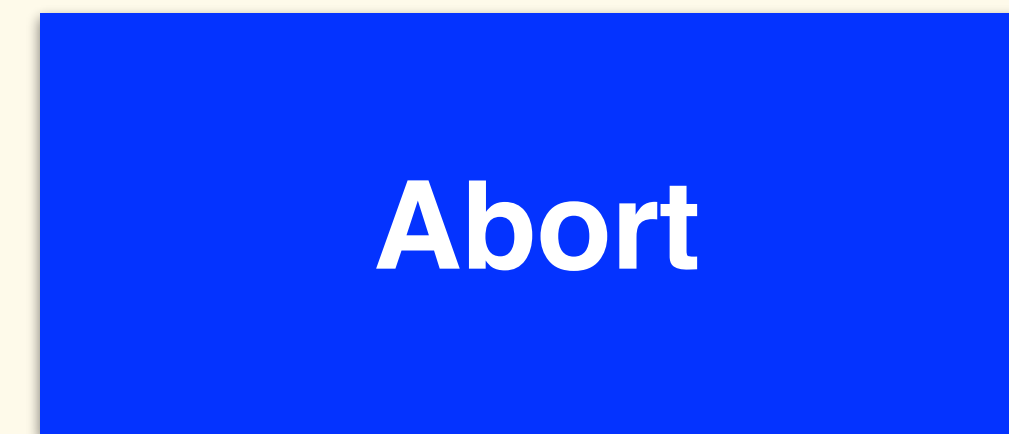
Simple changes have obvious results

```
button {  
  background: red  
}
```




Simple changes have obvious results

```
button {  
  background: blue  
}
```



Simple changes have obvious results



The web framework for perfectionists with deadlines.


OVERVIEWDOWNLOADDOCUMENTATIONNEWSCOMMUNITYCODEABOUT♥ DONATE

Django makes it easier to build better Web apps more quickly and with less code.

Get started with Django


Meet Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.



Ridiculously fast.

Django was designed to help developers take applications from concept to completion as quickly as possible.




Reassuringly secure.

Django takes security seriously and helps developers avoid many common security mistakes.

Download latest release: 1.10

DJANGO DOCUMENTATION ›

Support Django!



Augusto Destrero donated to the Django Software Foundation to support Django development. Donate today!

Latest news

Django 1.10 released

Django 1.10 has been released!

Posted by Tim Graham on August 1, 2016

Really great documentation

A shortcut: `get_object_or_404()`

It's a very common idiom to use `get()` and raise `Http404` if the object doesn't exist. Django provides a shortcut. Here's the `detail()` view, rewritten:

```
polls/views.py
```

```
from django.shortcuts import get_object_or_404, render
```



Philosophy

Why do we use a helper function `get_object_or_404()` instead of automatically catching the `ObjectDoesNotExist` exceptions at a higher level, or having the model API raise `Http404` instead of `ObjectDoesNotExist`?

Because that would couple the model layer to the view layer. One of the foremost design goals of Django is to maintain loose coupling. Some controlled coupling is introduced in the `django.shortcuts` module.

Because that would couple the model layer to the view layer. One of the foremost design goals of Django is to maintain loose coupling. Some controlled coupling is introduced in the `django.shortcuts` module.

There's also a `get_list_or_404()` function, which works just as `get_object_or_404()` – except using `filter()` instead of `get()`. It raises `Http404` if the list is empty.

Really great documentation

stripe

DocumentationSupportSign In

DEVELOPMENT

Getting Started

Embedded Form

Custom Forms

Mobile Apps

Charging Cards

Testing

Security

Fraud Protection

Webhooks

ACH Guide

Bitcoin Guide

Alipay Guide

File Upload Guide

ACCOUNT

Your Account

Getting Paid

Disputes

Submitting Evidence

Dispute Types

Disputes FAQ

Reporting

Integrations

REFERENCES

Checklist

Examples

Stripe.js

Checkout

Recipes

API Libraries

API Upgrades

Full API Reference

SUBSCRIPTIONS

Getting Started

Making Your First Charge

Use Stripe's API and your server-side code to process charges. If you need help after reading this, check out our answers to [common questions](#) or chat live with other developers in [#stripe](#) on freenode.

Once you've securely collected and tokenized your customer's credit card using [Checkout](#) or [Stripe.js](#), you can charge the card. Unlike collection, which occurs in the browser, charge attempts are made from your server, normally using one of our [client libraries](#). If you haven't already, install the library for your favorite language now. This tutorial shows code for Ruby, PHP, Python, and Node, but we also have libraries for Java and Go.

Tutorial requirement

This tutorial assumes you've already implemented a solution for collecting and tokenizing your customers' credit cards. If you haven't, check out our [receiving cards docs](#) before proceeding.

On your server, grab the Stripe token in the POST parameters submitted by your form. From there, it's one simple API call to charge the card:

rubypythonphpnodejava

```
# Set your secret key: remember to change this to your live secret key in production
# See your keys here https://dashboard.stripe.com/account/apikeys
Stripe.api_key = "sk_test_BQokikJOvBiI2HlWgH4oIfQ2"

# Get the credit card details submitted by the form
token = params[:stripeToken]

# Create the charge on Stripe's servers - this will charge the user's card
begin
  charge = Stripe::Charge.create(
    :amount => 1000, # amount in cents, again
    :currency => "usd",
    :source => token,
    :description => "Example charge"
  )
rescue Stripe::CardError => e
  # The card has been declined
end
```

As a convenience, if you're logged in while reading this page, we've pre-filled the example with your *test secret* API key. Only you can see this value. This will authenticate you to Stripe, so keep it secret and keep it safe. Remember to replace the test key with your live key in production. You can get all your keys from [your account page](#).

That's it! If the charge creation request succeeds, the card has been successfully charged. You will automatically receive your money in [two days](#). If the charge attempt fails, we'll return an [error](#) instead.

Synchronicity

Builds confidence in
tending the world

Builds confidence in tending the world

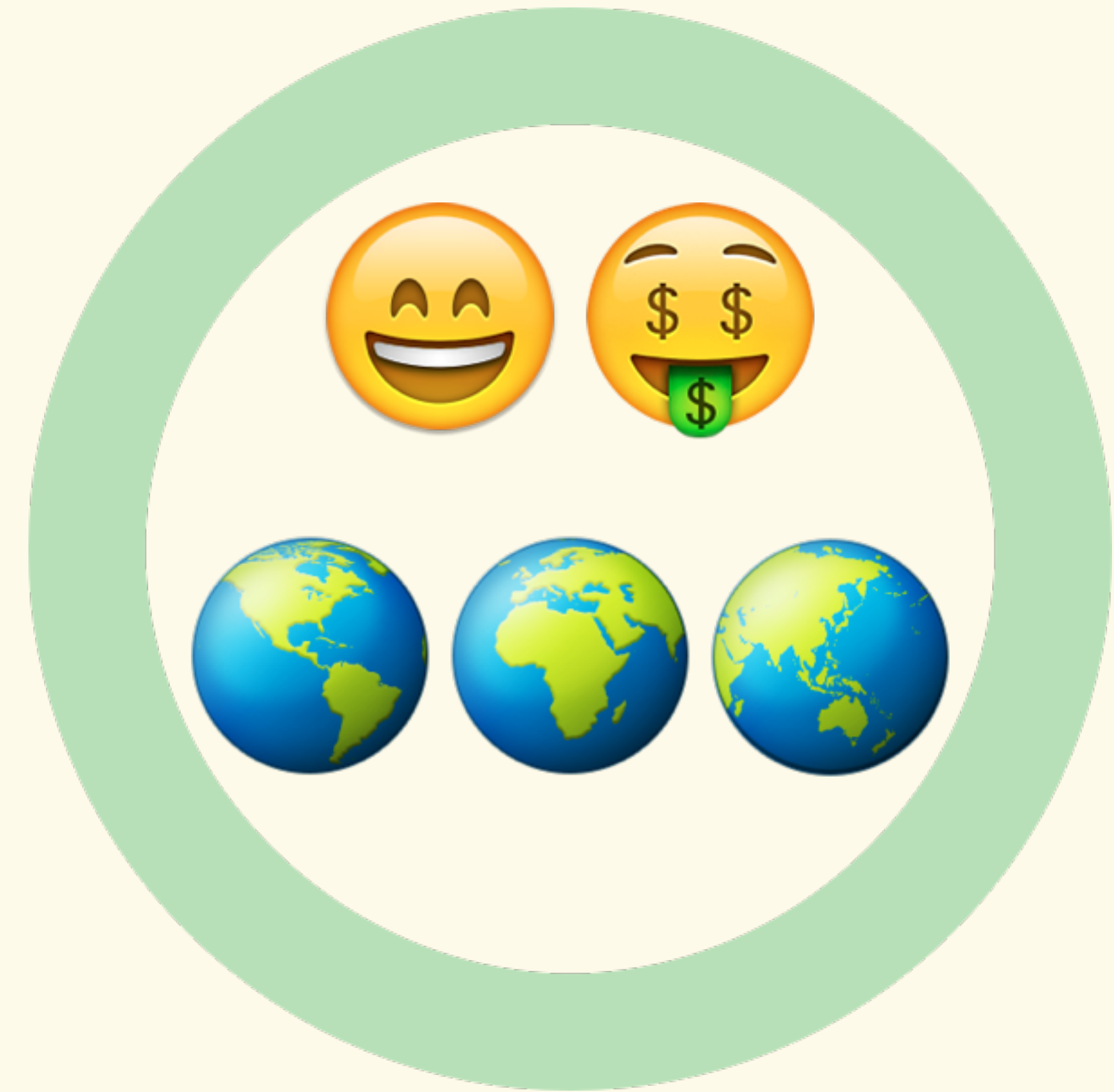
- **Small wins add up, especially when learning**
- **Automated tests are confidence builders**
- **Consistency & testability create a positive feedback loop**

Inspires happiness &
empowerment

Inspires happiness & empowerment

- **Happiness and confidence breed creativity**
- **Empowered developers means nearby worlds improve**
- **The sum of your system's & team's value can grow**
- **You can hire and work with great people!**

Inspires happiness & empowerment



Takeaways

Takeaways

- **Systems have lots of interfaces**
- **Take the time to think about them (give things good names!)**
- **Create your worlds from small, great, consistent stories**
- **Humans first. Humans first. Humans first. Have empathy.**

Bit Operations

| X | Y | X&Y | X Y | X^Y | ~X |
|---|---|-----|-----|-----|----|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |

...and now they have

$(11101001 \wedge 00101011) | (01010111 \& 10111000) \ll 32$

problems...

Thank You

Credits & Colophon

- “Brain” by 뇌진소 from the Noun Project, purchased for use
- “B-52 lower deck” from Wikimedia Commons by Desertsy85450 [Public Domain]
- “Patravi” watch from Wikimedia Commons by Carlbucherer [CC BY-SA 4.0]
- Typefaces are Montserrat & Roboto Slab from Google Fonts [Open Font License]
- Color palette is “Adrift in Dreams” by Skyblue2u on Colour Lovers
- Diagrams built with OmniGraffle & Keynote, images edited in Pixelmator
- This slide deck is released under a Creative Commons BY-SA 4.0 License.

Made in Austin, Texas · July–August, 2016

<https://interfaces.design/>

