Building Worlds

&

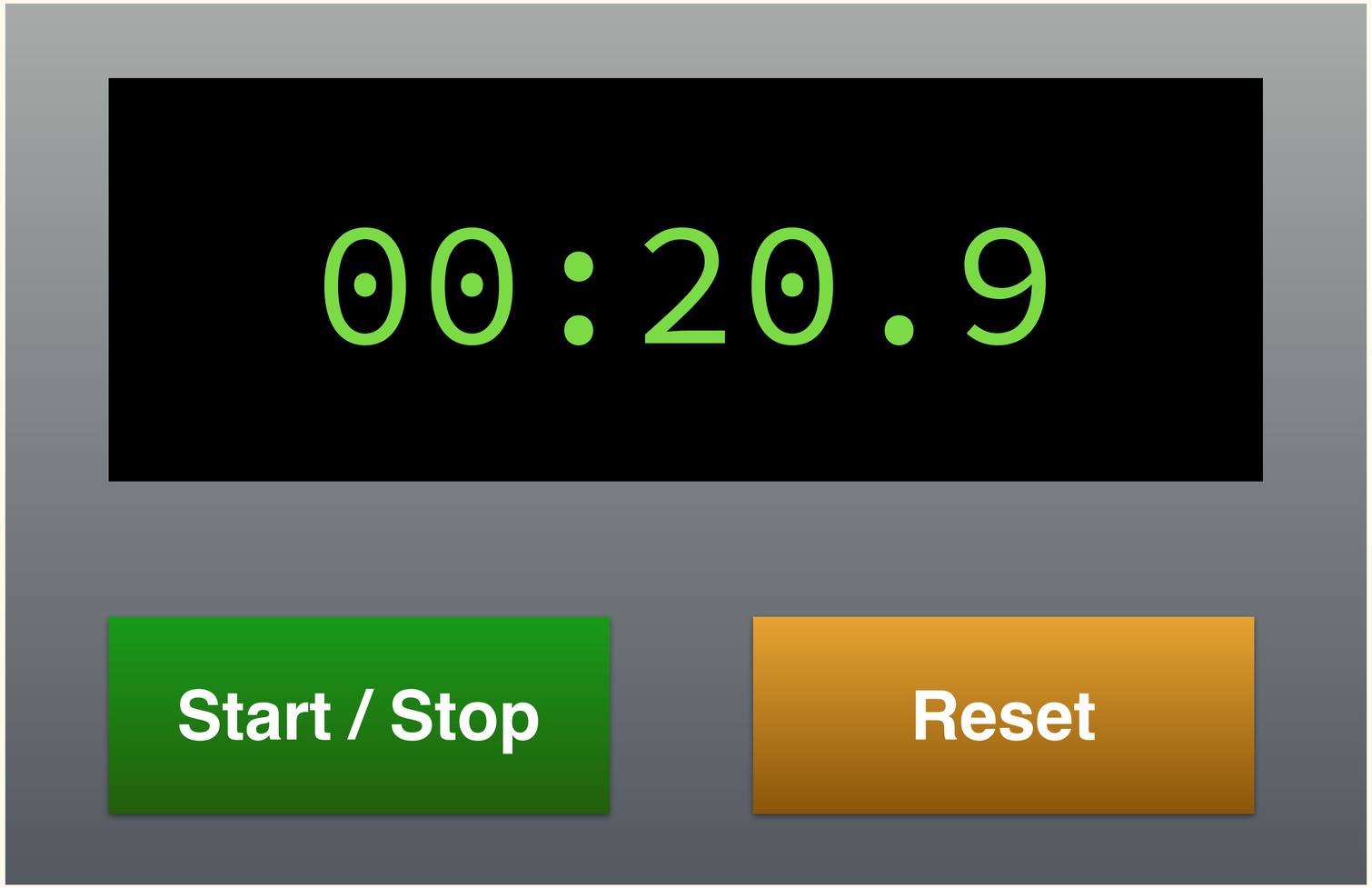
Feeling Great

Brad Fults · @h3h
Head of Engineering Operations
Under Armour

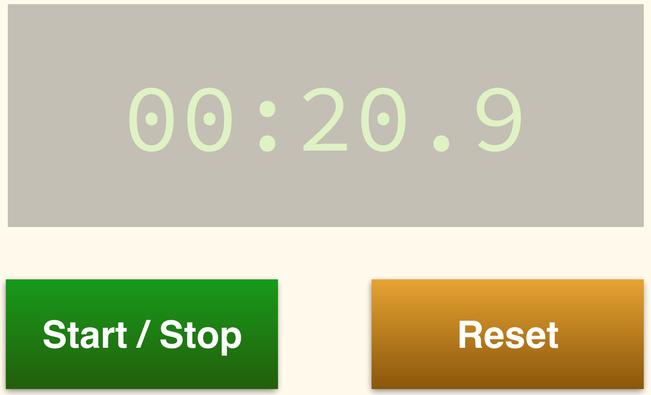
#buildingworlds

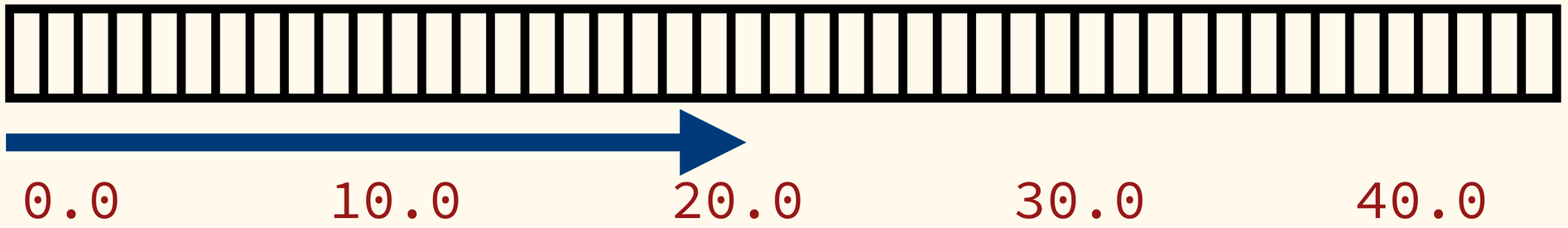
Interfaces

Interfaces

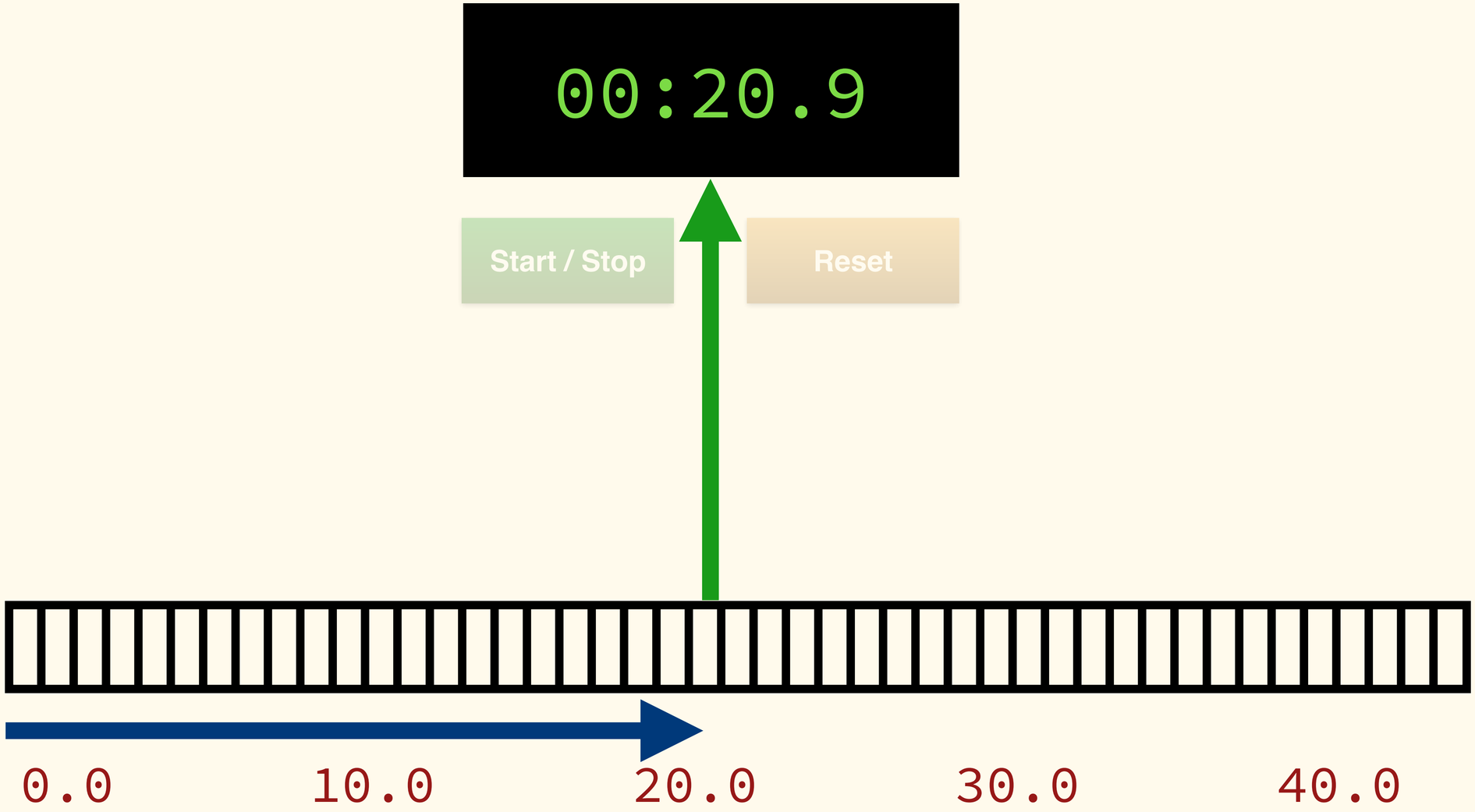


Interfaces

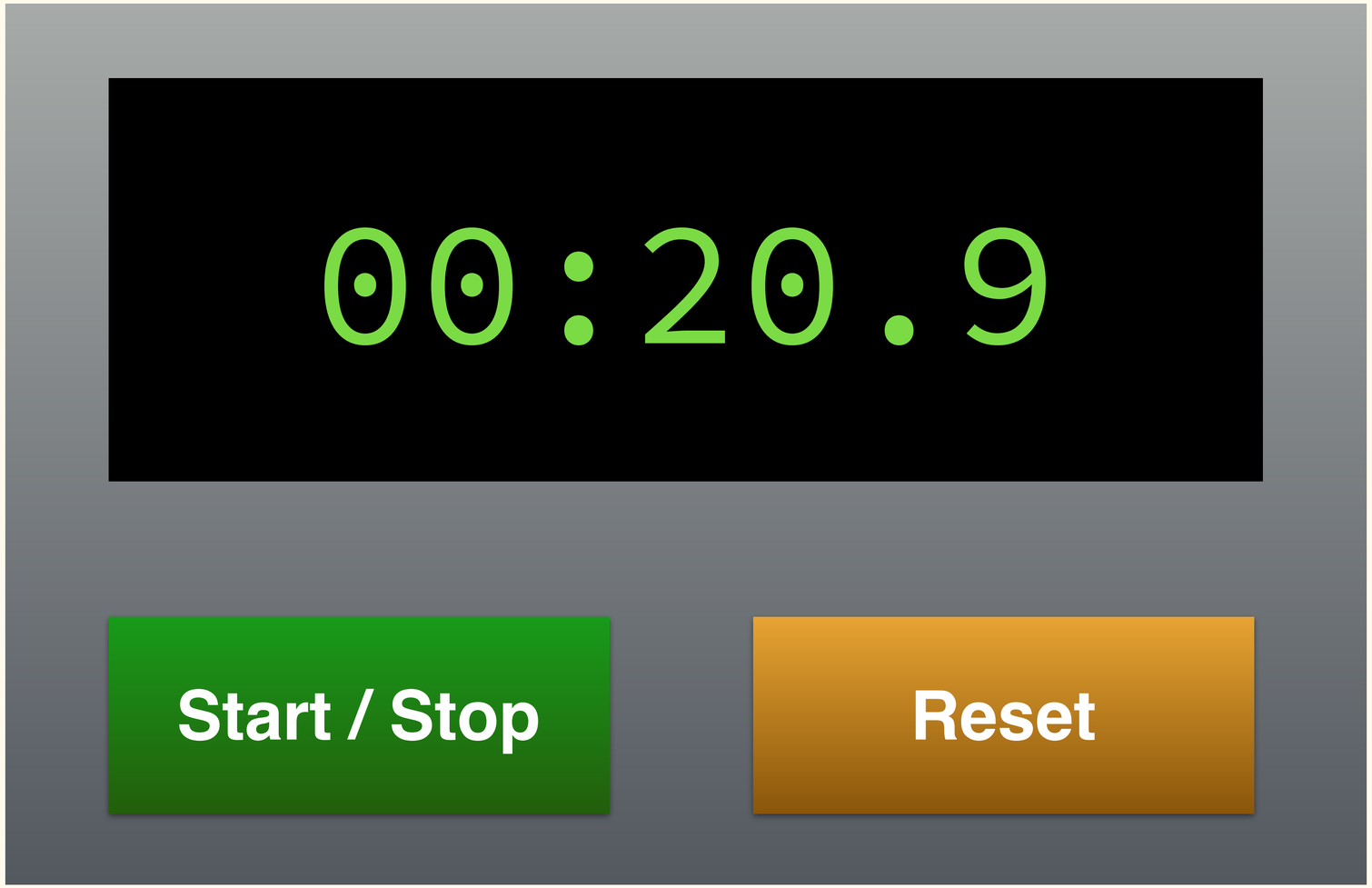




Interfaces



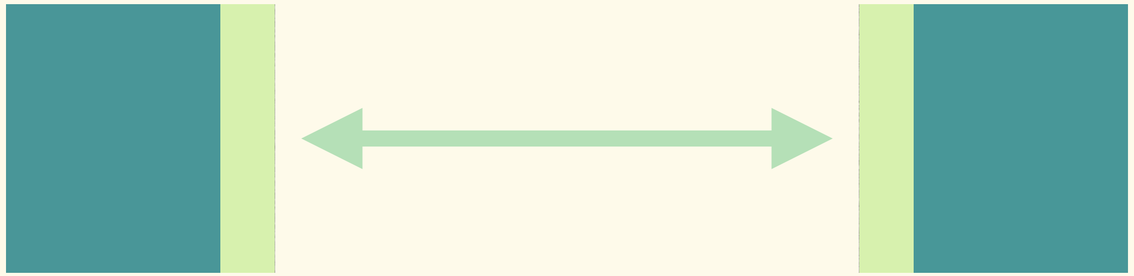
Interfaces



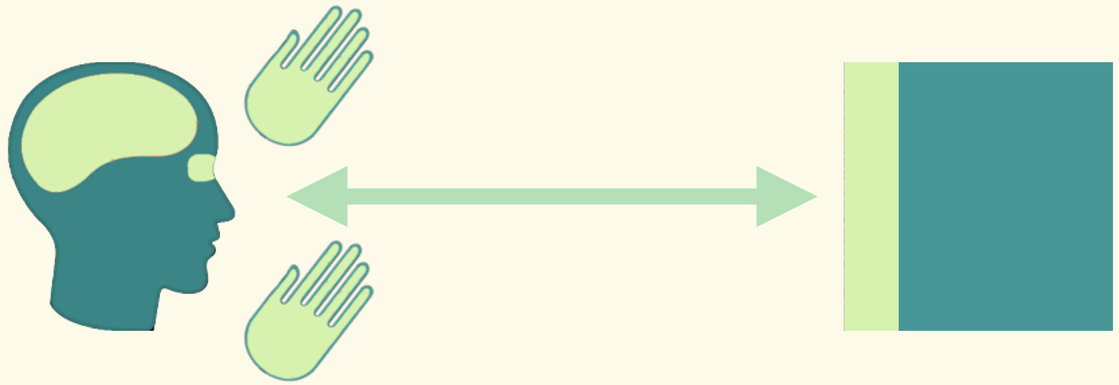
Interfaces

```
Stopwatch

- start
- stop
- reset

- watch_time
- counting?
```

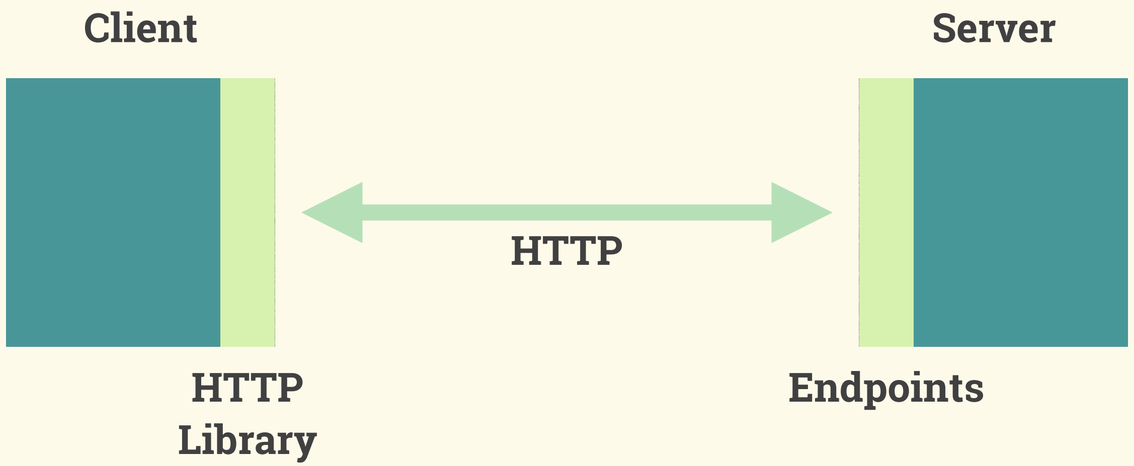
Interfaces



Interfaces

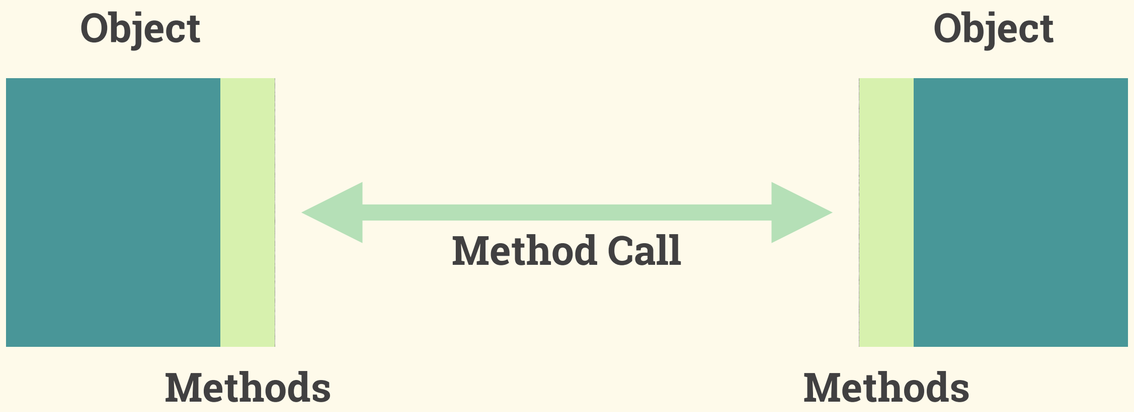


Interfaces



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

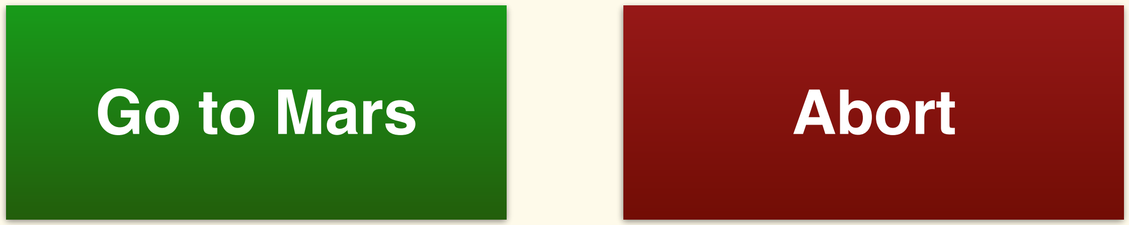
```
class Person

  def first_initial
    first_name.first.upcase
  end

  def last_initial
    last_name.first.upcase
  end

end
```

Interface

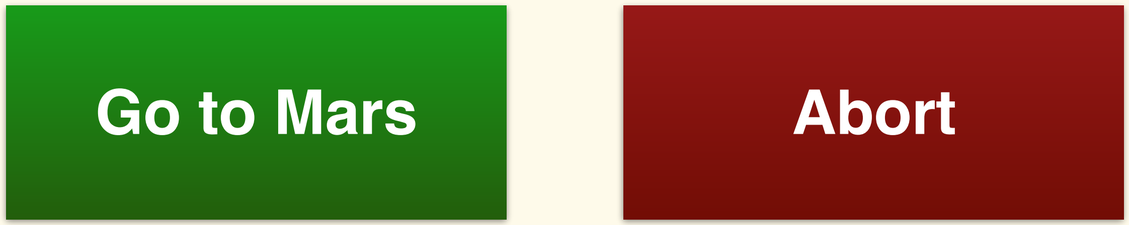
```
Person

- first_initial
- last_initial
```

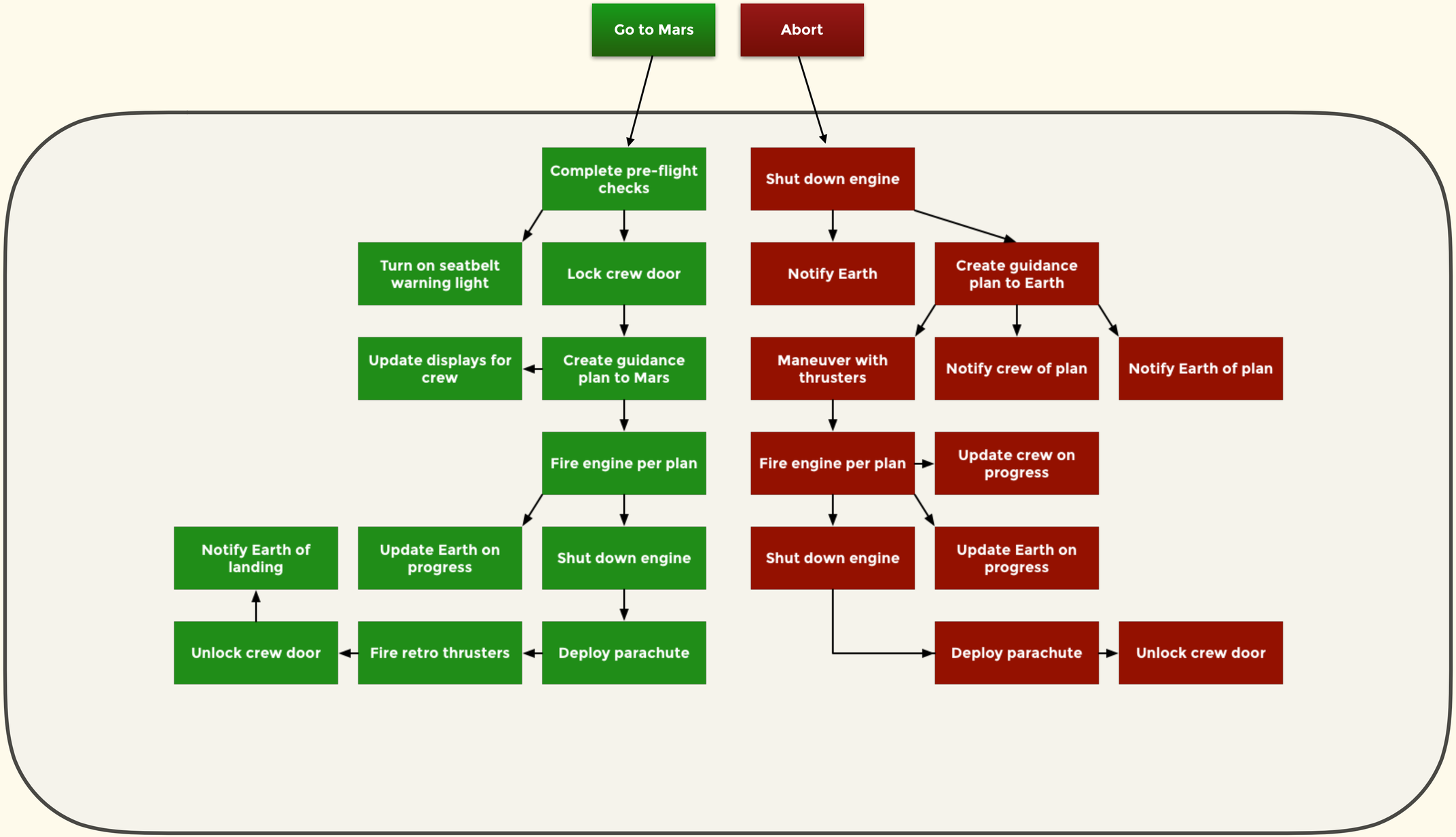
Levels of Abstractions.io

Levels of Abstraction



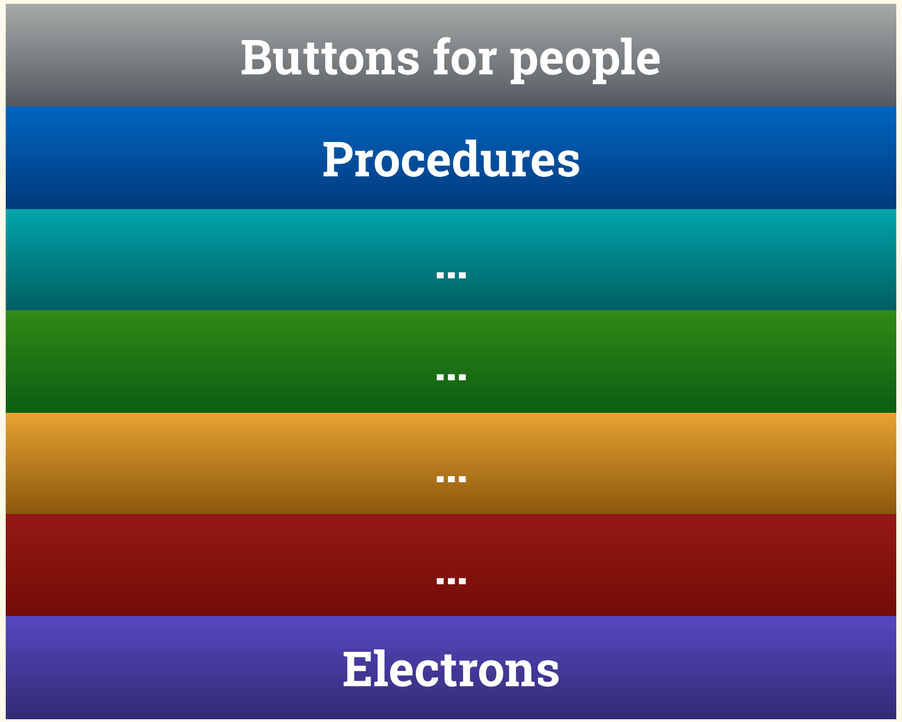
Levels of Abstraction



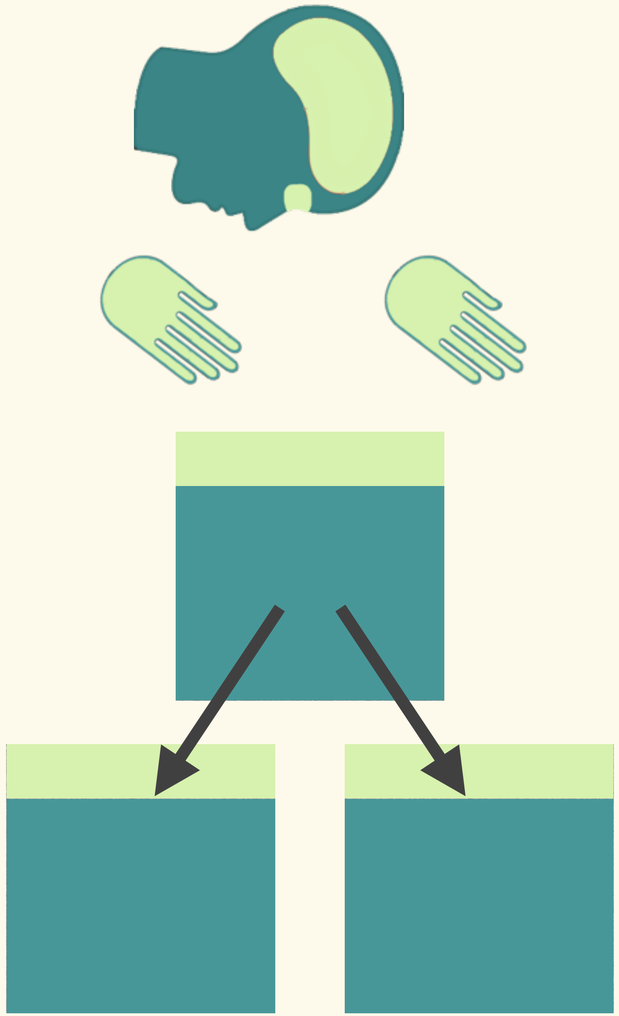

Go to Mars
Abort
Complete pre-flight checks
Turn on seatbelt warning light
Lock crew door
Update displays for crew
Create guidance plan to Mars
Fire engine per plan
Notify Earth of landing
Update Earth on progress
Shut down engine
Unlock crew door
Fire retro thrusters
Deploy parachute
Shut down engine
Notify Earth
Create guidance plan to Earth
Maneuver with thrusters
Notify crew of plan
Notify Earth of plan
Fire engine per plan
Update crew on progress
Shut down engine
Update Earth on progress
Deploy parachute
Unlock crew door

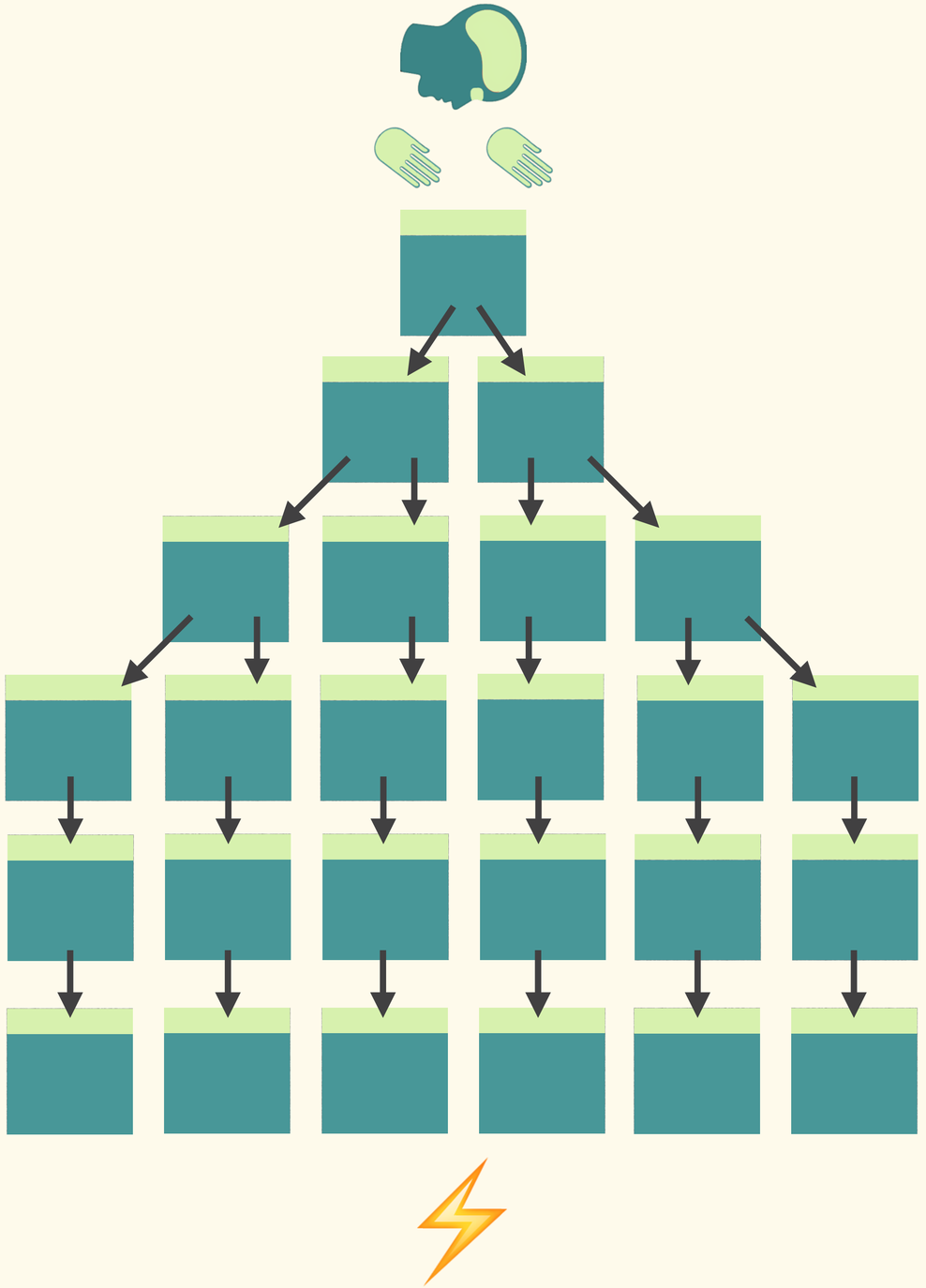
Levels of Abstraction

Higher

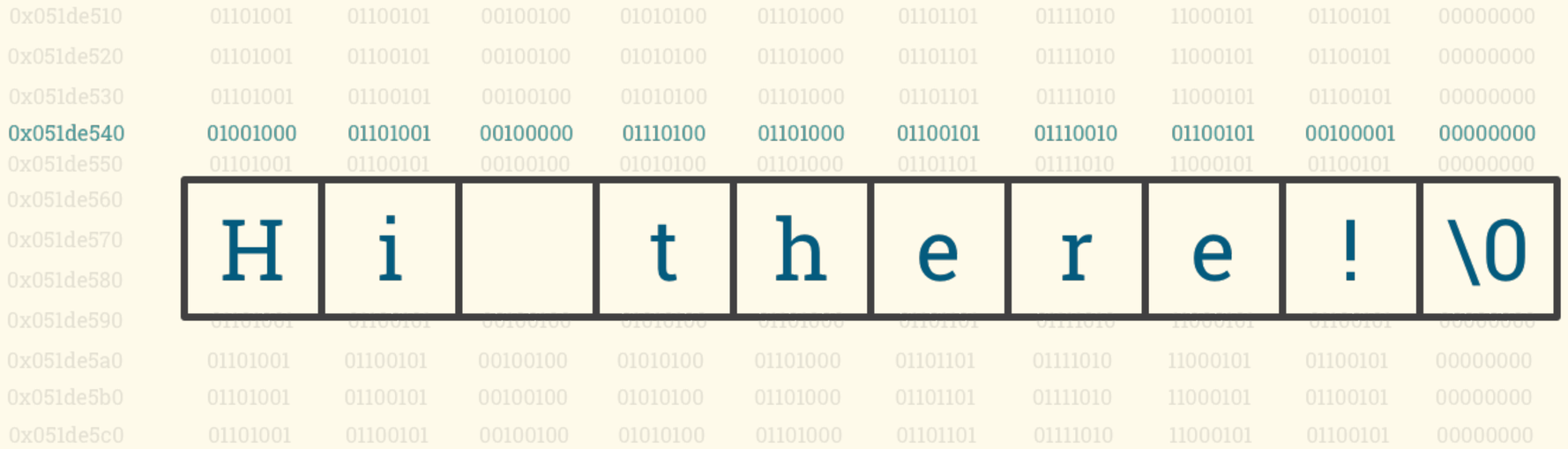


Lower





Levels of Abstraction




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

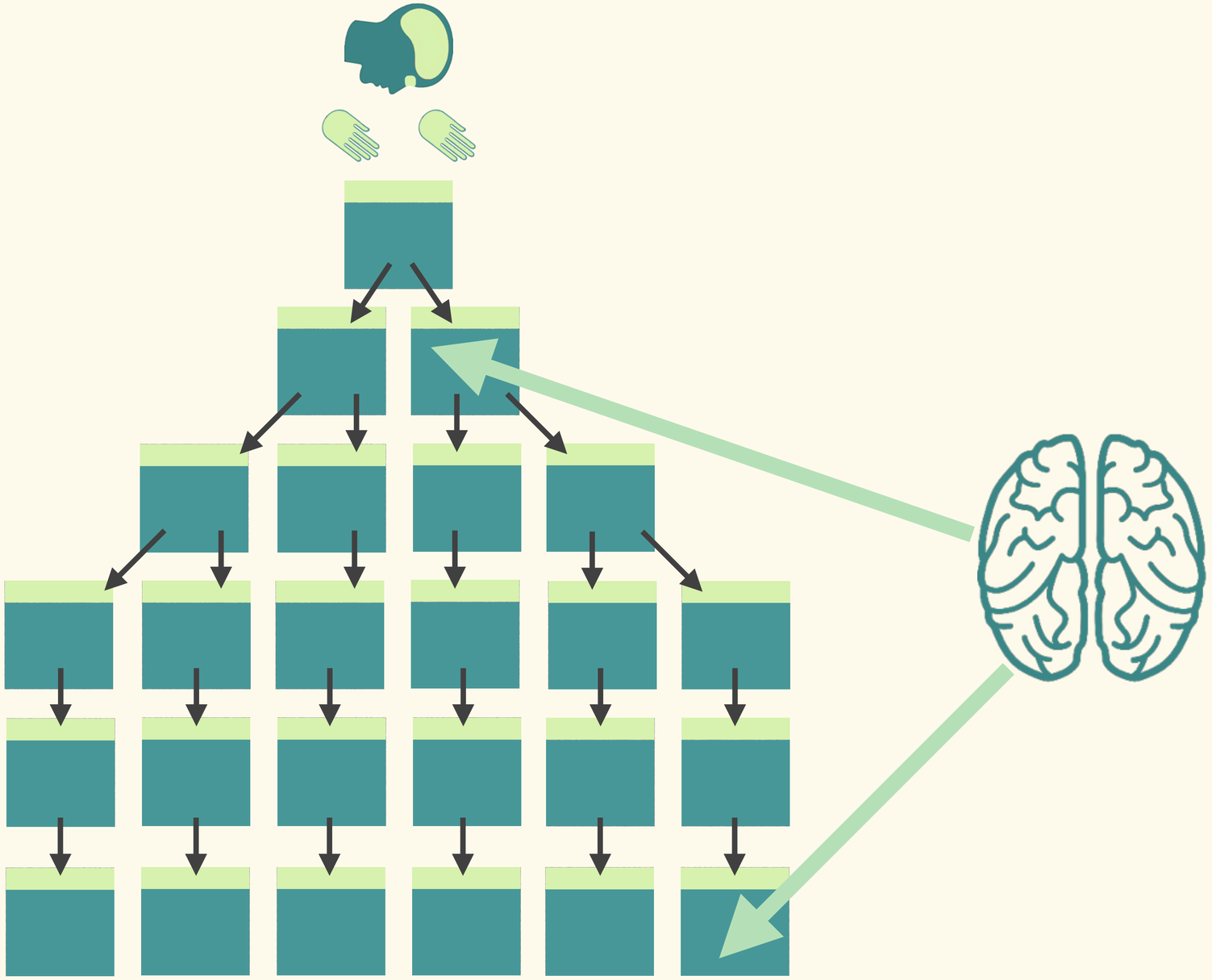
Levels of Abstraction

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

Higher

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

Lower

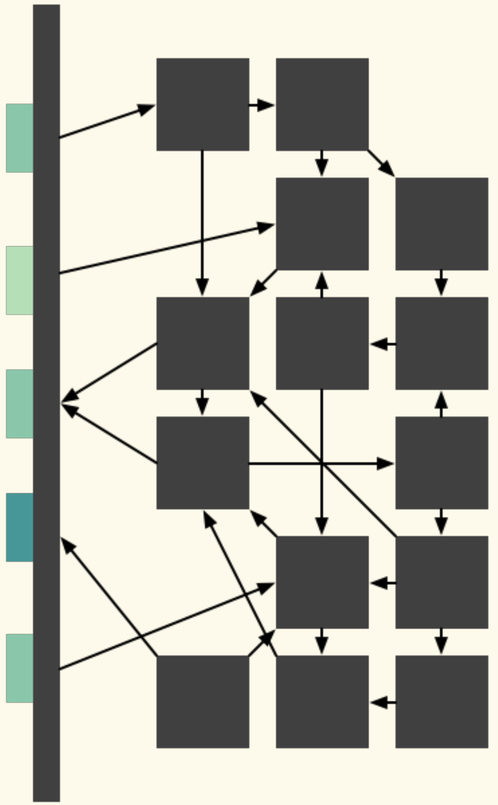


Building Worlds

Building Worlds

Complexity

Difficulty



Interface World

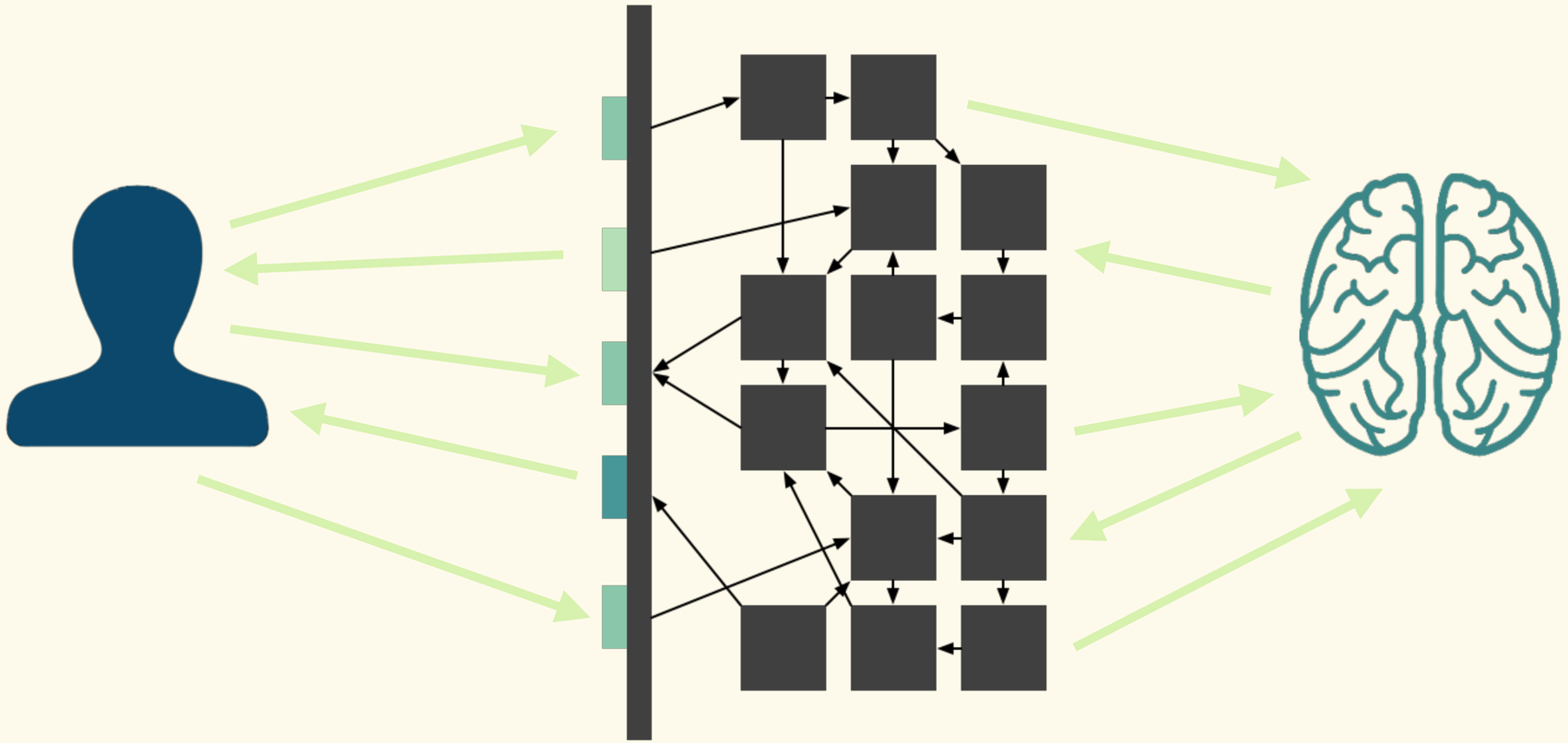
Interfaces

Implementation

```
class Person

  def first_initial
    first_name.first.upcase
  end

  def last_initial
    last_name.first.upcase
  end

end
```

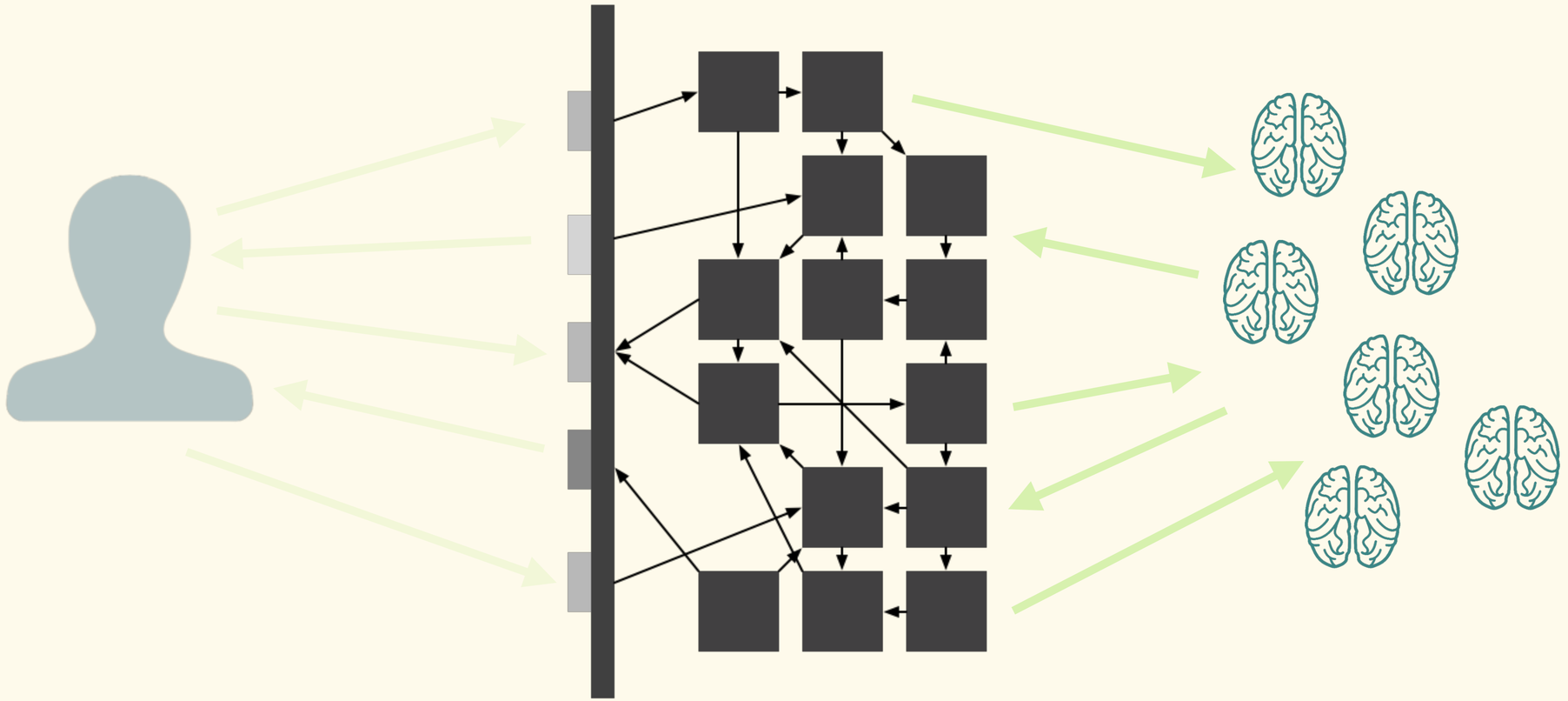
Interface

```
Person

- first_initial
- last_initial
```



User Interface Design

World **Building**



User Interface Design

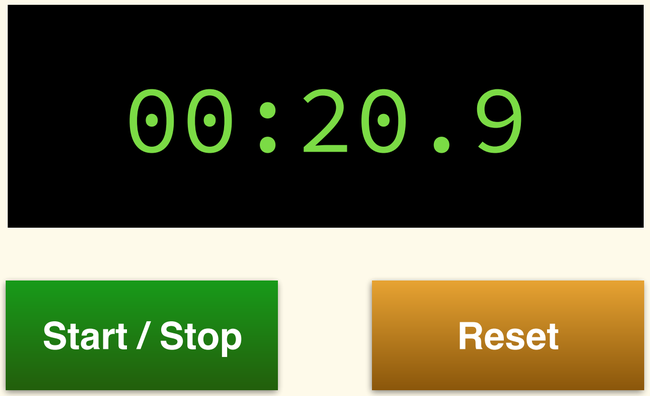
World Building

Building Worlds

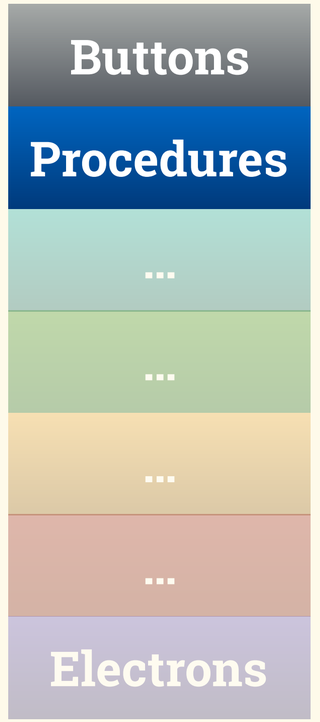
1. It’s interfaces all the way down
2. A good interface works at one level of abstraction
3. Coupling is expensive forever
4. Systems are for humans first, machines second

It’s interfaces all
the way down

It’s interfaces all the way down




```
Stopwatch

- start
- stop
- reset

- watch_time
- counting?
```

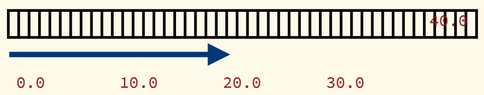


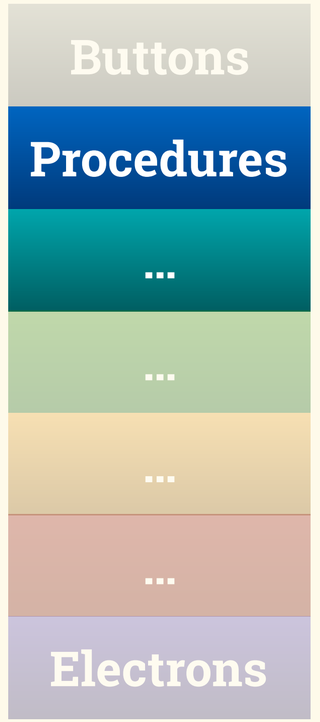
It’s interfaces all the way down

```
Stopwatch

- start
- stop
- reset

- watch_time
- counting?
```



```
Clock

- tick

- value
```



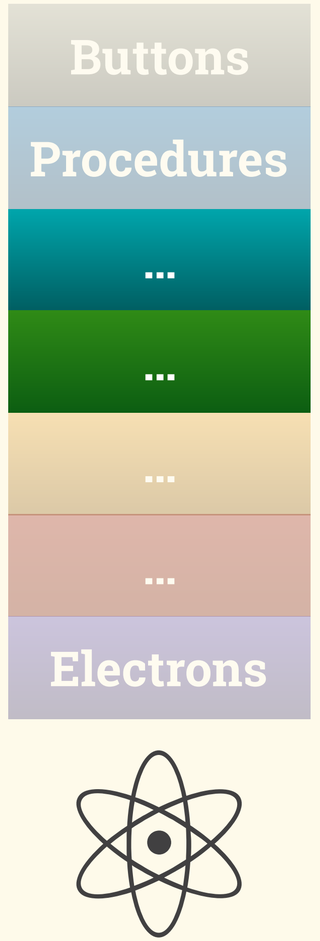


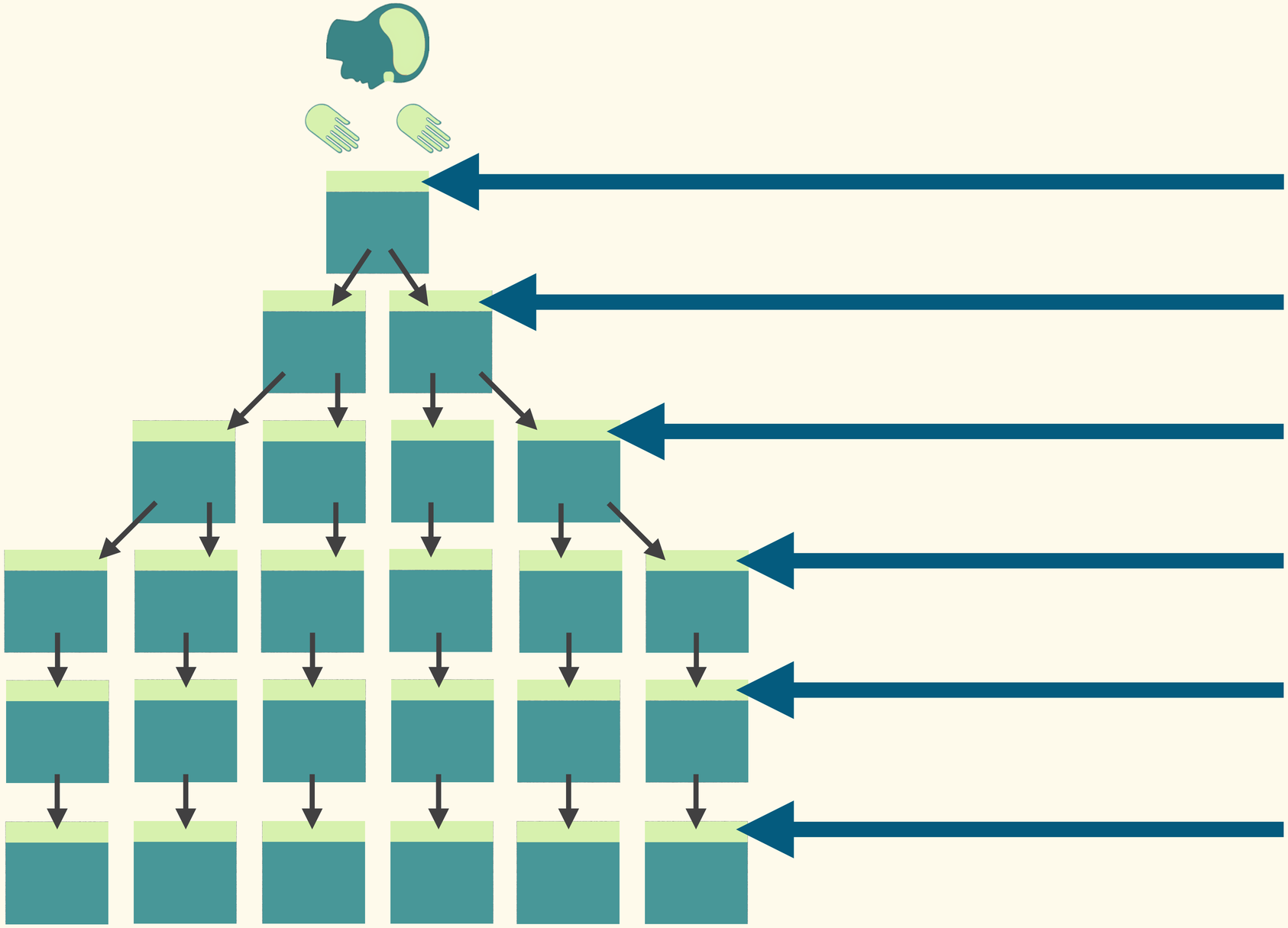
It’s interfaces all the way down

```
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);
```



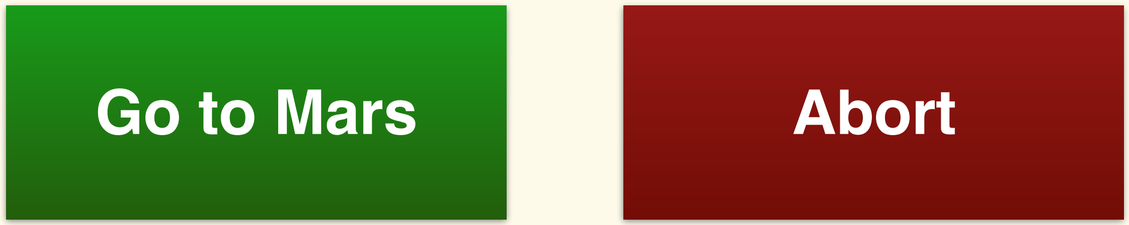


It’s interfaces all the way down

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

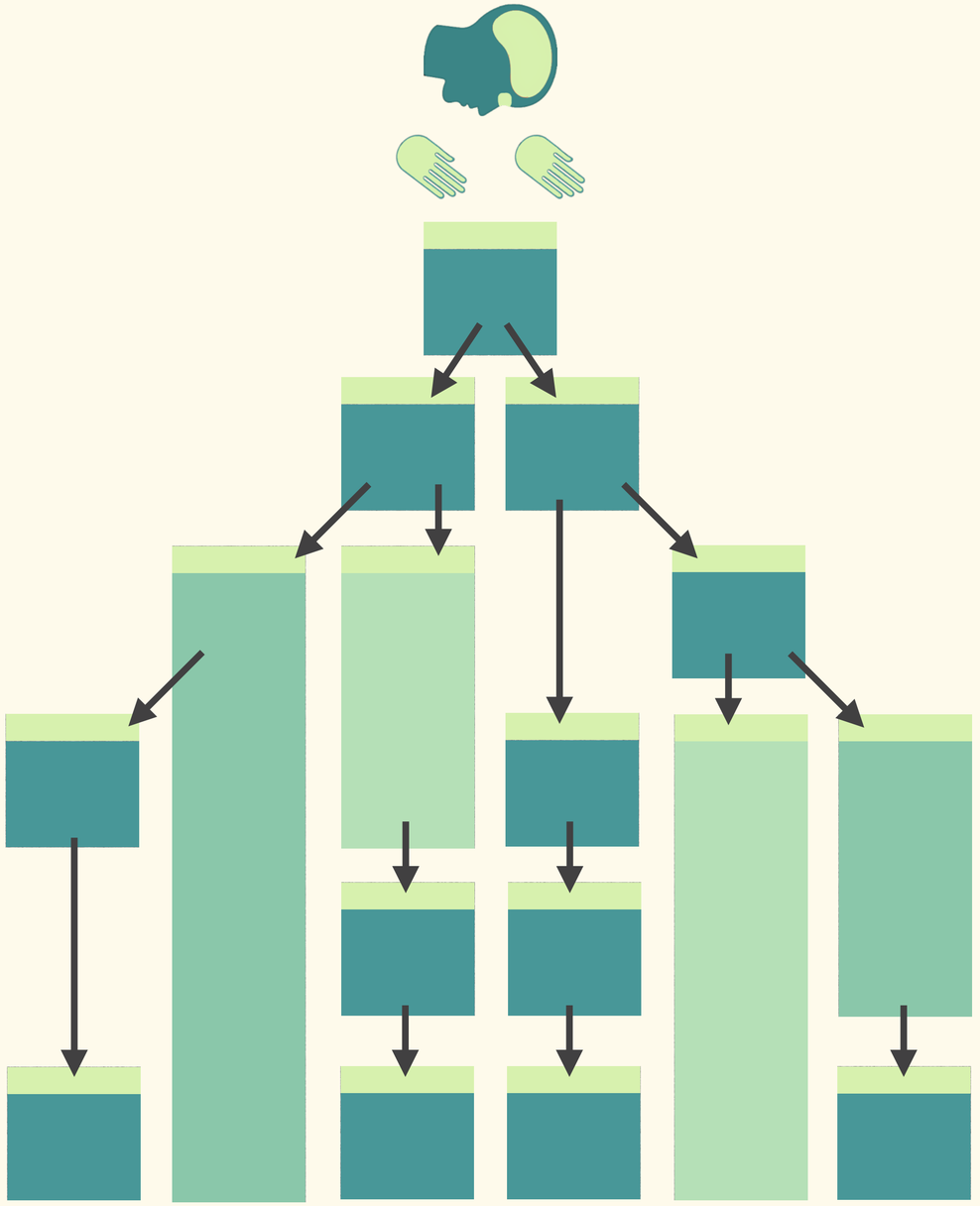
A good interface works at one level of abstraction

A good interface works at one level of abstraction



A good interface works at one level of abstraction

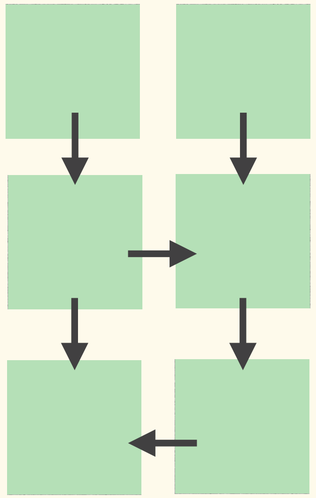
A good interface works at one level of abstraction

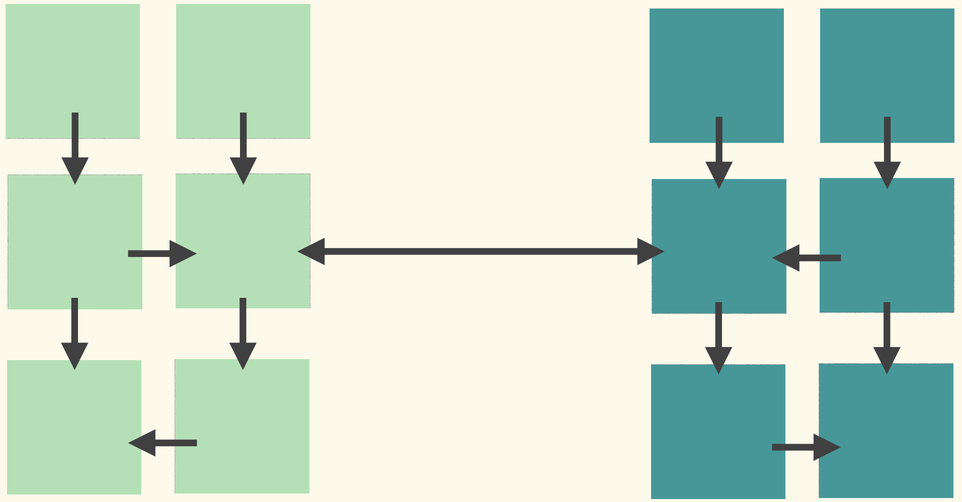


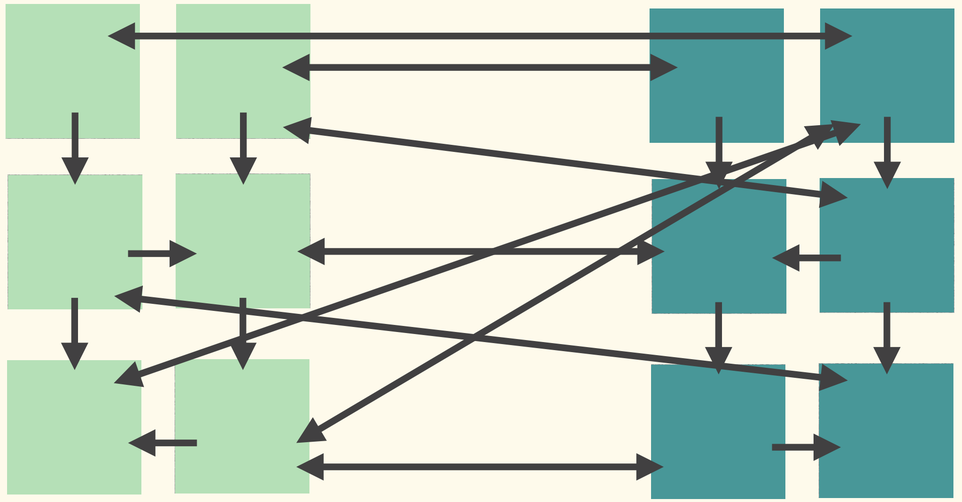
A good interface works at one level of abstraction

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

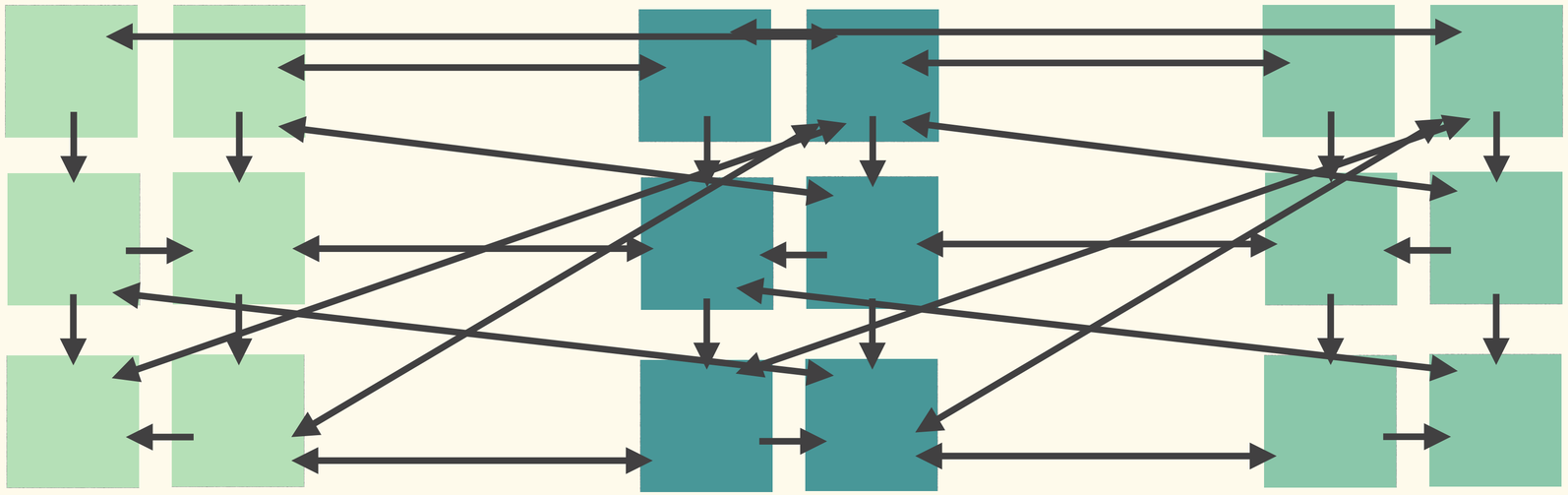
Coupling is expensive forever



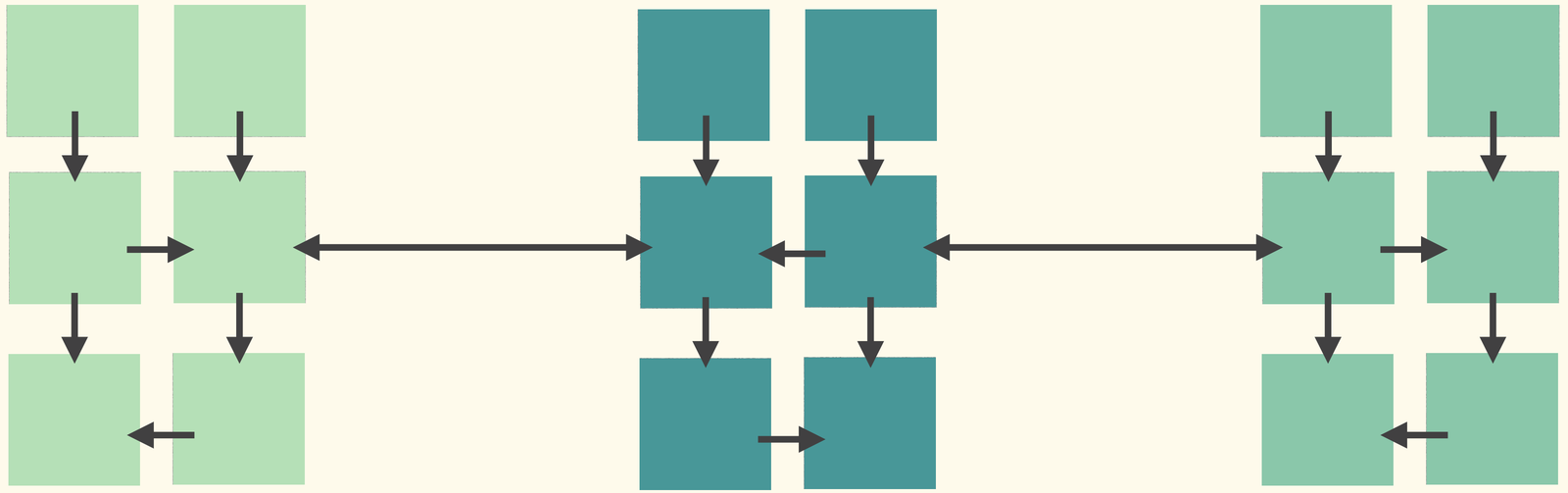




Tight Coupling



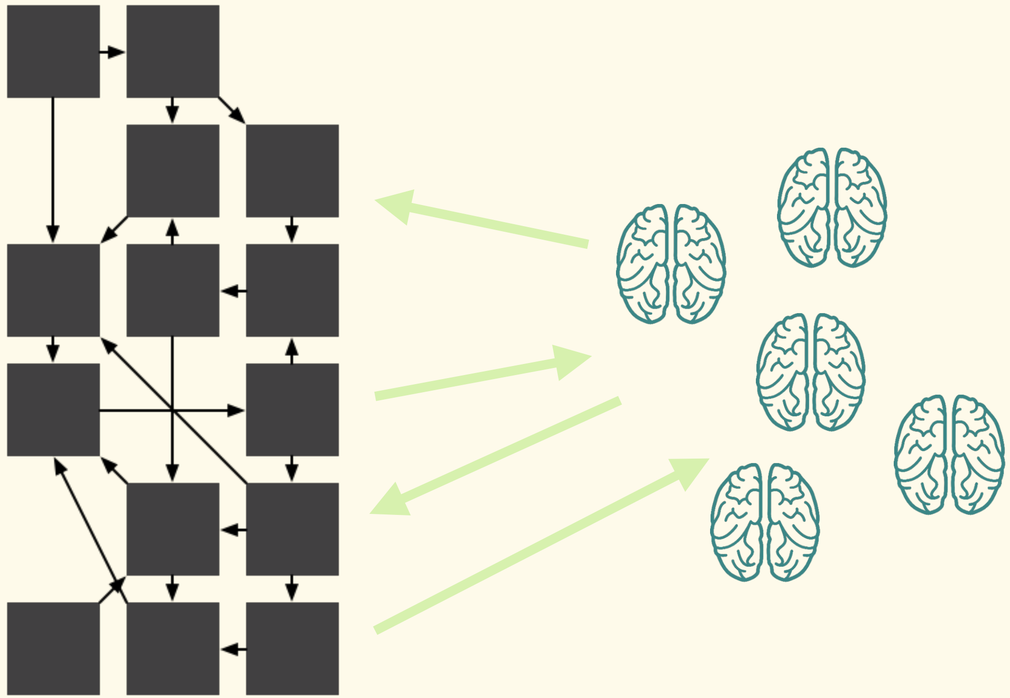
Tight Coupling



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

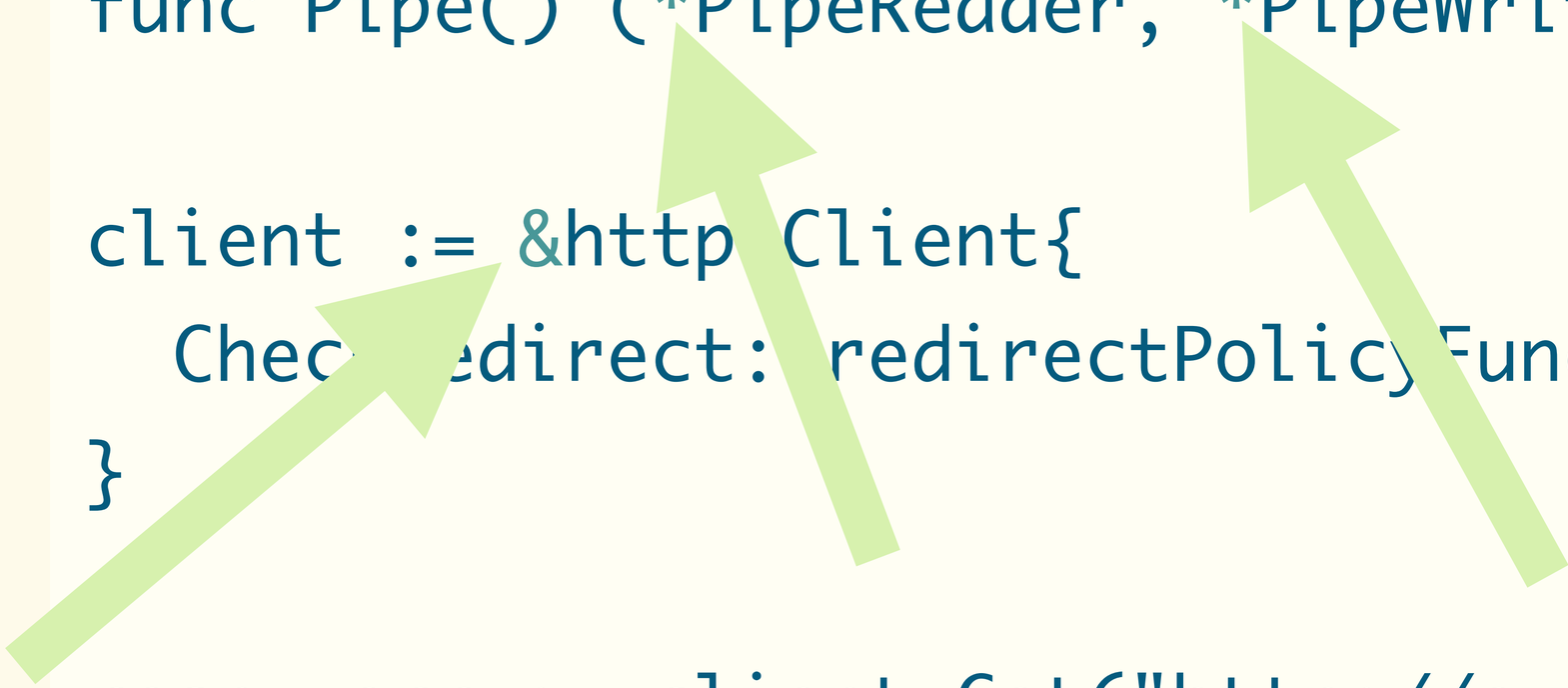
Neil Gaiman

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")
```



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(…)
```

Files

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

Links

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

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

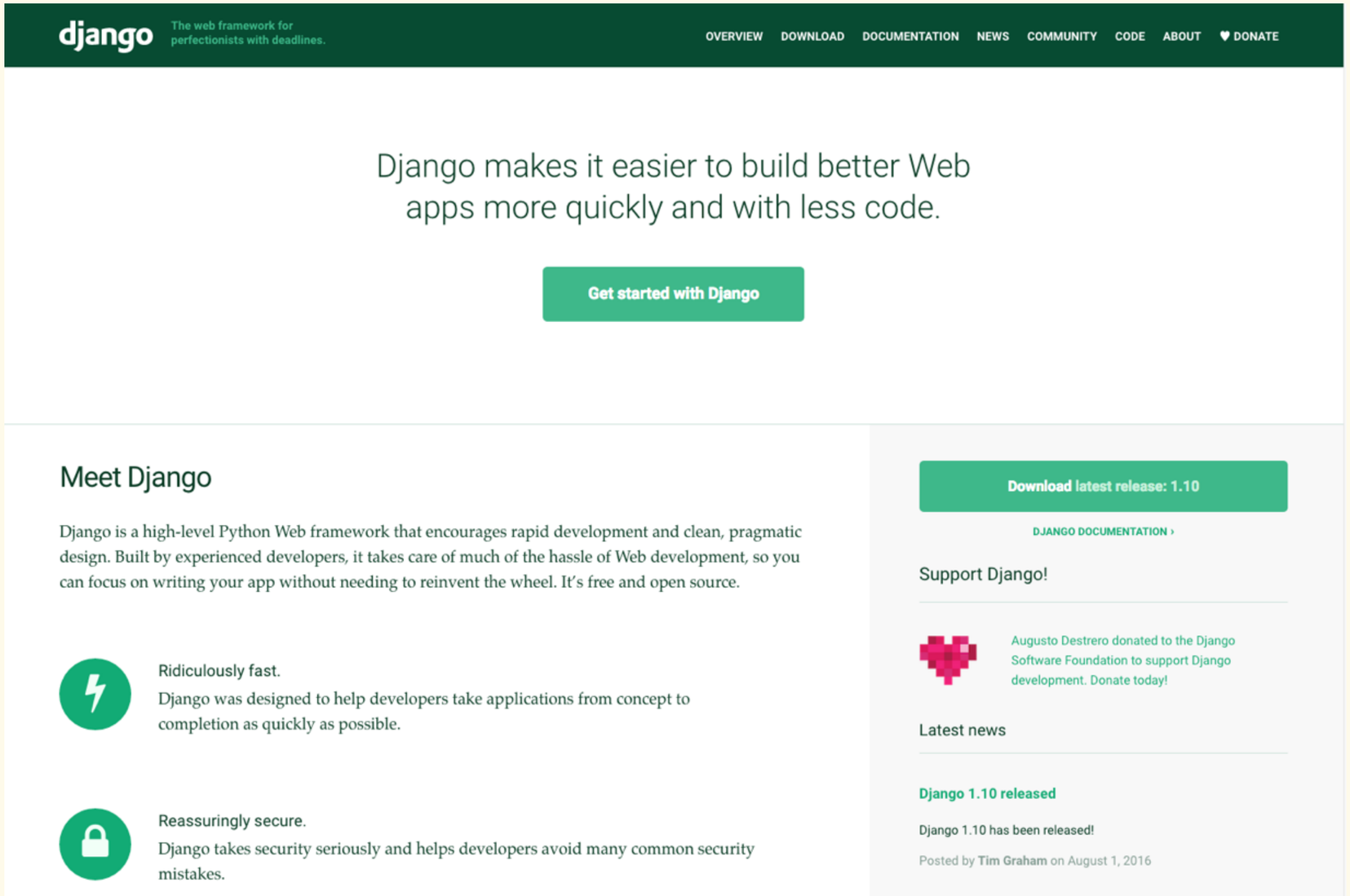
Simple changes have obvious results

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

Simple changes have obvious results

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

Simple changes have obvious results



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:

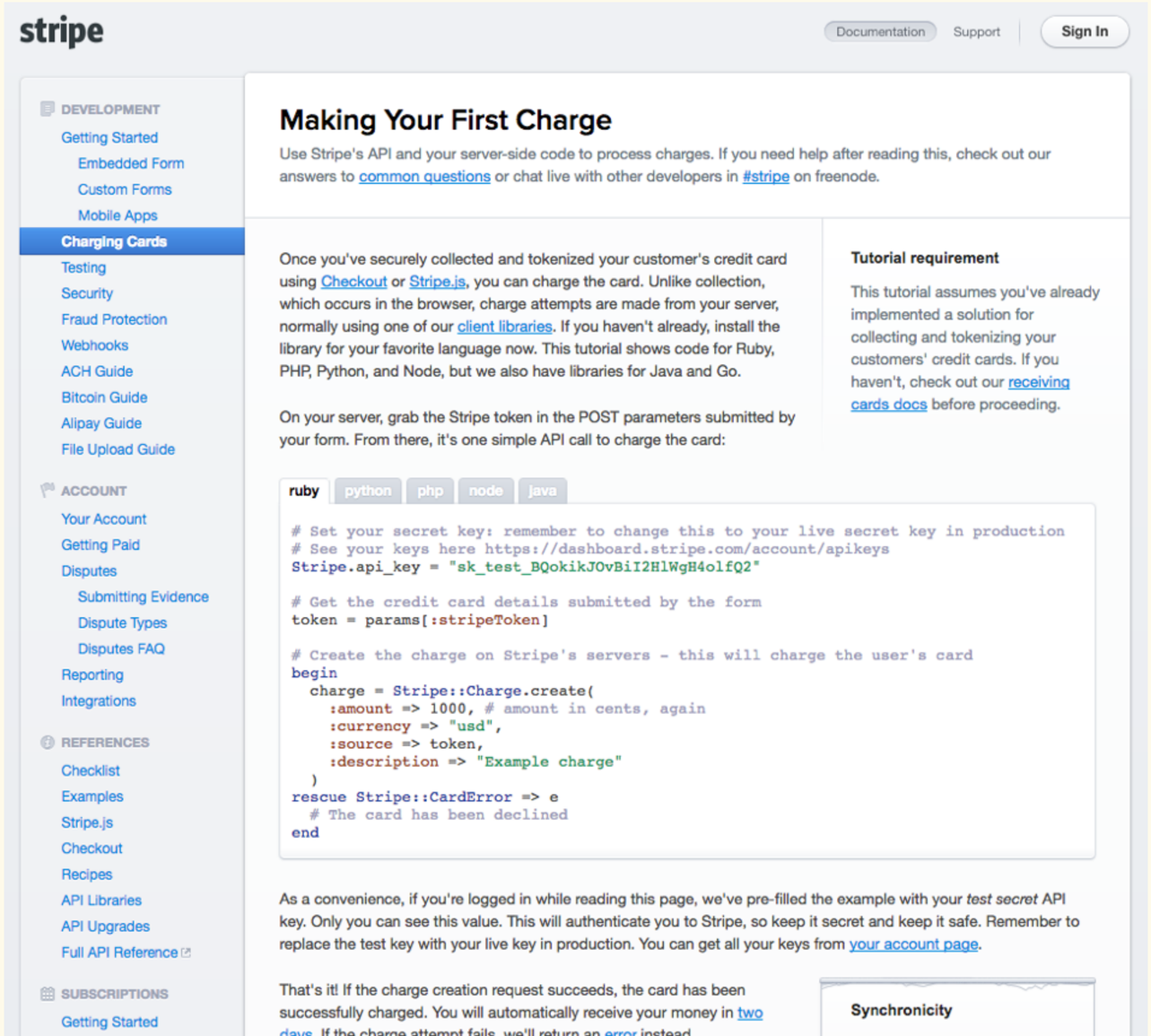
```
polls/views.py

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.

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



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 ^ 00101011) | (01010111 & 10111000)) << 32

problems...

Thank You

Credits & Colophon

- “Brain” by 녕진소 from the Noun Project, purchased for use
- “B-52 lower deck” from Wikimedia Commons by Desertsky85450 [Public Domain]
- “Patravi” watch from Wikimedia Commons by Carlfbucherer [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

Made in Austin, Texas · July–August, 2016

https://interfaces.design/