

How To Learn Rust

5/19/2016





talks.edunham.net/oscon2016
oscon2016@edunham.net



Two Talks In One

Thanks

@anjuan
@Azuxul
@bulba_zord
@echorand
@EnamsuoBarry
@FranklinWaller
@gansai9

@glasnt
@gmearthe
@hectorjcorrea
@iamed2
@IanBertolacci
@Ignoreintuition
@itsAhmedWay

@jackyboen
@joaomello
@jsookha
@l.d.walker
@llogiq
@macaronique
@mayaskme

@megamatman
@Mike_Fal
@proogey
@robn
@sundayayandokun
@Vinatorul
<http://agares.info/>



Today's Audience



The background of the slide is a close-up photograph of ancient Egyptian hieroglyphs. The hieroglyphs are carved into a dark, textured stone surface, likely limestone or granite. They are arranged in horizontal rows, separated by thin, recessed lines. The lighting is dramatic, coming from the side, which creates strong highlights on the raised surfaces of the hieroglyphs and deep shadows in the recessed areas, emphasizing their three-dimensional quality. The colors range from dark brown to a slightly lighter, yellowish-brown where the light hits.

Your Next Language



Curiosity

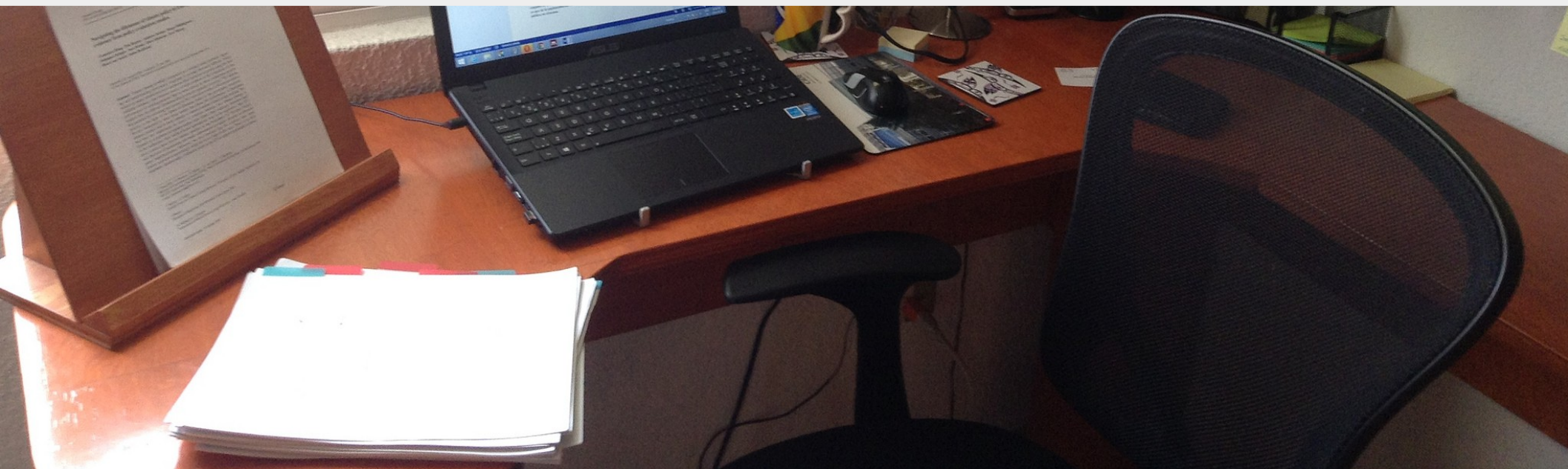


Two golden retriever puppies are looking over a white banner that covers the middle of the image. The puppy on the right is looking slightly to the right, while the one on the left is looking towards the camera.

Features



Work



The background of the slide is a blurred photograph. In the upper half, a black pencil sharpener is visible, slightly out of focus. In the lower half, a sharpened pencil with a reddish-brown body and a white eraser is lying horizontally. The pencil's lead tip is sharp and pointed. The overall scene is set against a light, neutral background.

Why?



Pattern Recognition





***Your* Next Language**



Learning Styles





Tasks You Enjoy

vs

Tasks You Despise



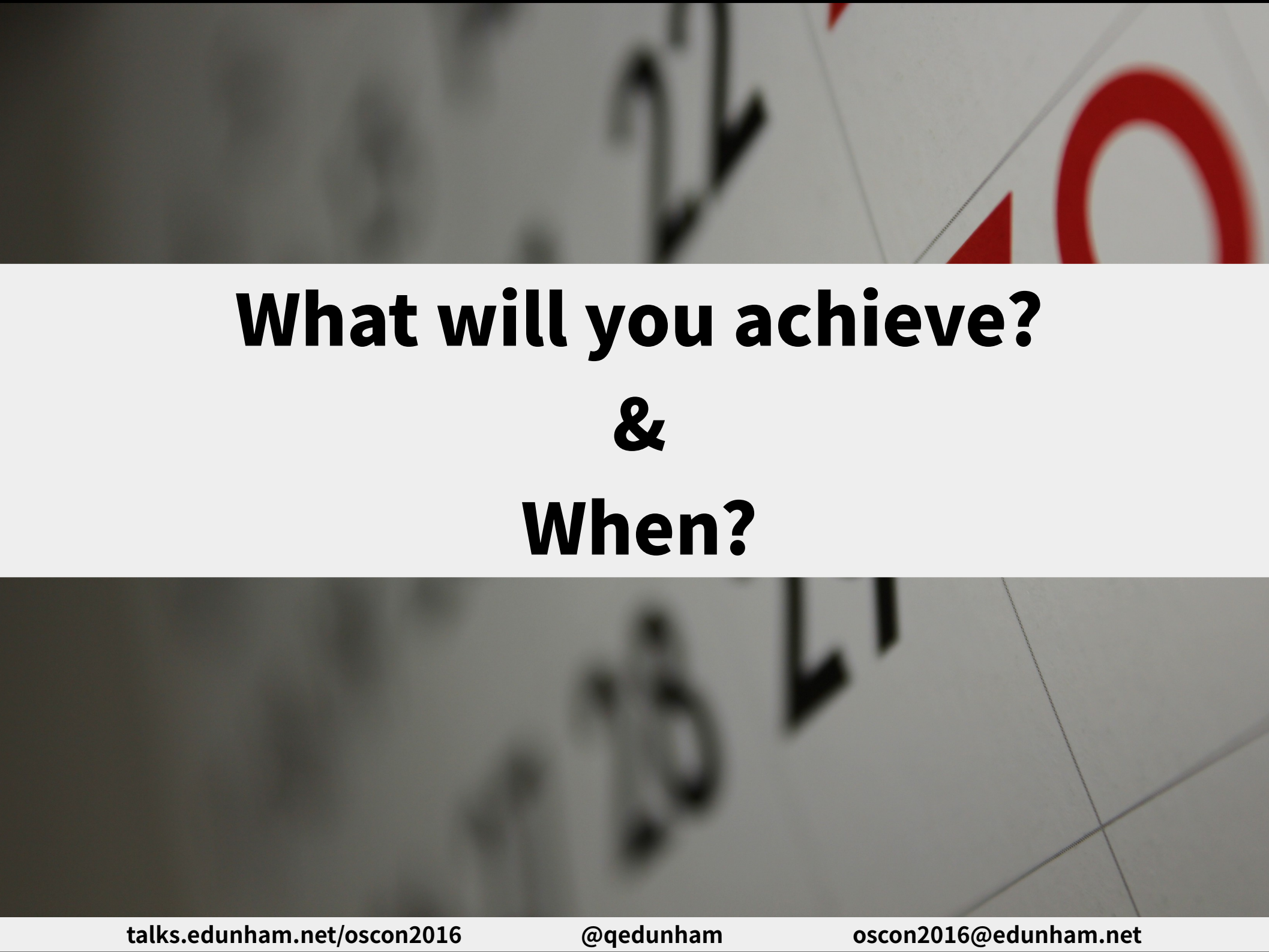


Successful Projects

vs

Unsuccessful Projects





What will you achieve? & When?



Which Language?





Popularity





Community





Resources



Features





Old & New Concepts





“My next language _____”





It's worth taking time from...

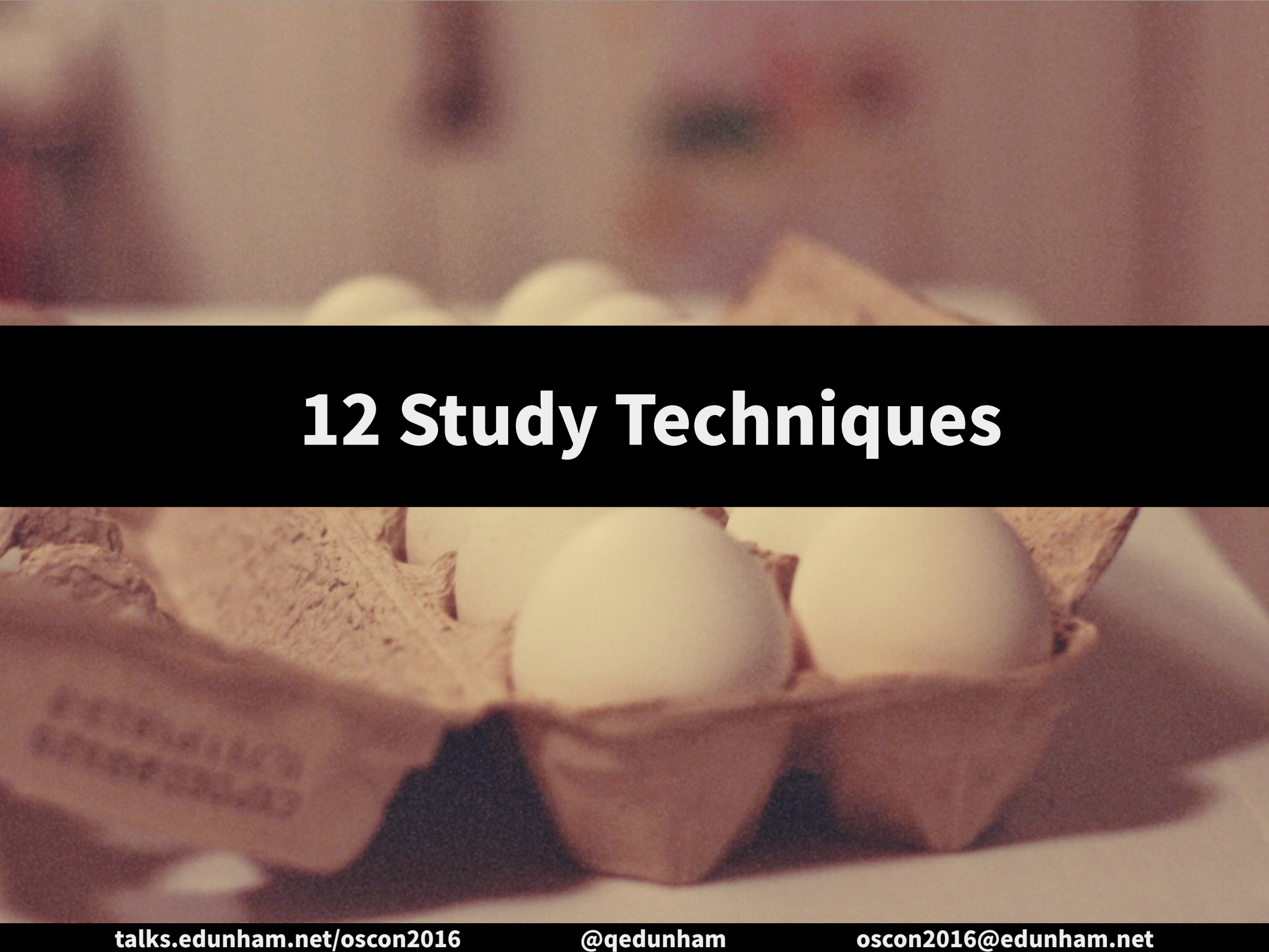




Family? Hobbies? Sleep?







12 Study Techniques



1) Language Docs





2) Similar Languages





3) Community Support





4) Write Real Code





5) Read The Books



6) Examine Examples





7) Find Good Tools





8) Read Real Code





9) Write Toy Programs





10) Google & Stackoverflow



11) Watch Lectures & Courses





12) Maintain Enthusiasm

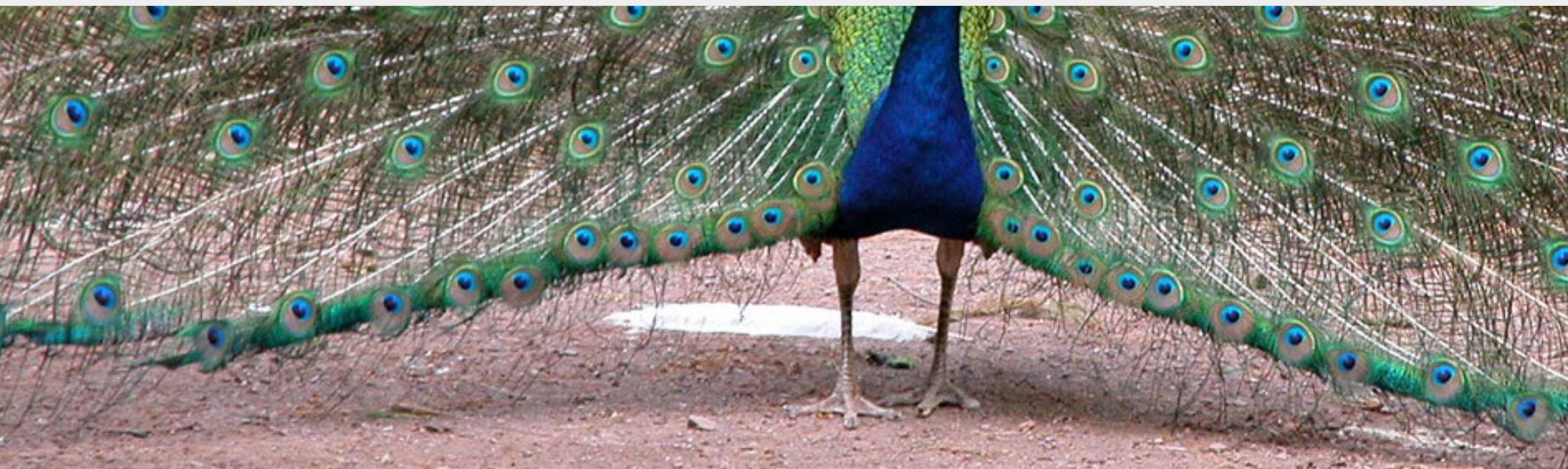




Troubleshooting



Community Interaction





Time





Language features





Advice



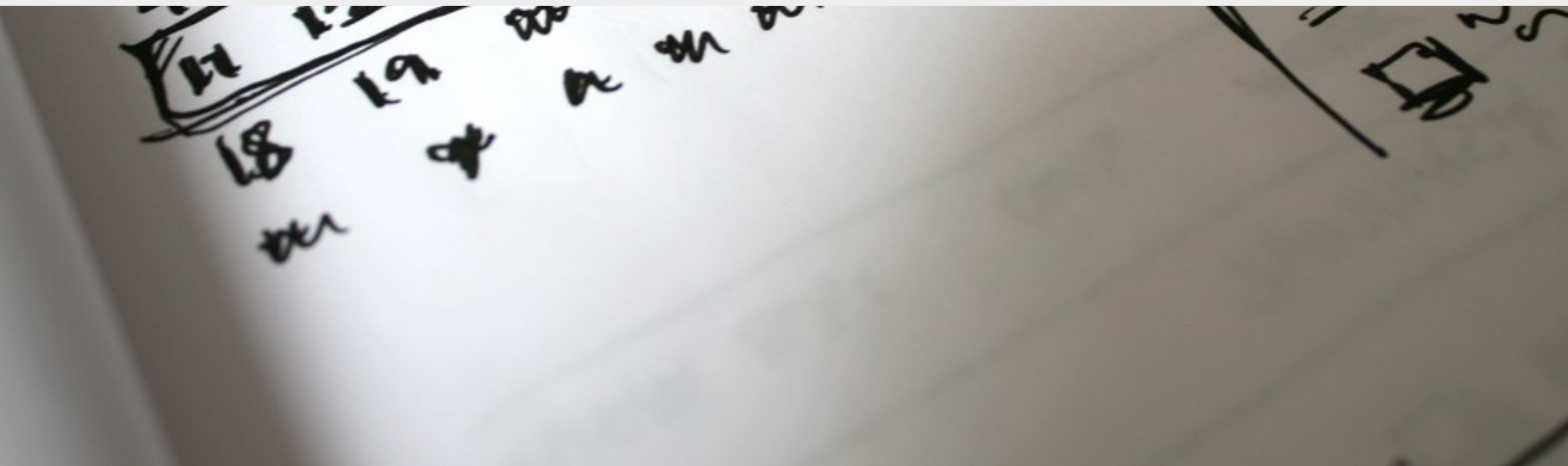


Learn by Doing





Practice Consistently





Start with the Basics



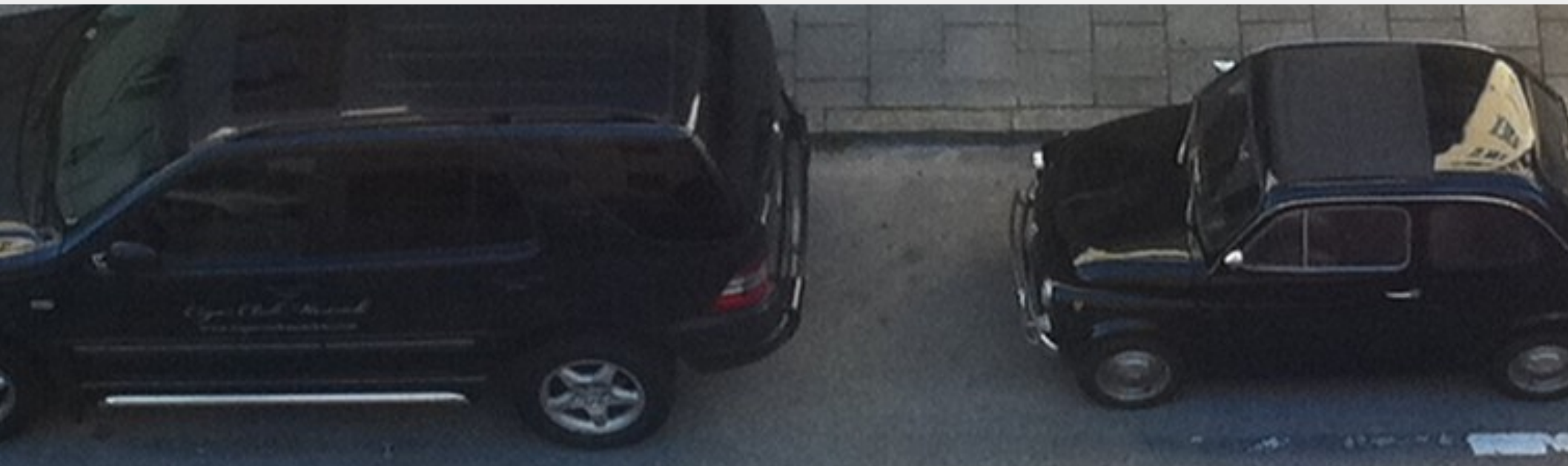


Build Something You Love





Respect Languages' Differences





It Gets Better





Learning Rust!





Are You Ready?



www.rust-lang.org
1.0 on May 15, 2015





Systems Programming. Safe, Concurrent, Fast.





Memory Safety, no GC. Ownership + Lifetimes. Types & Traits.



“I am learning @rustlang to ____”



1) Language Docs

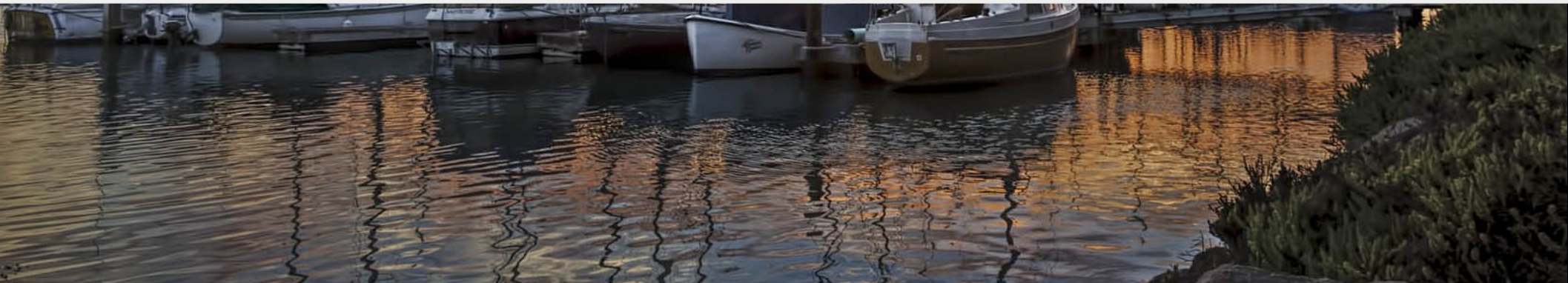
doc.rust-lang.org





2) Similar Languages

github.com/ctjhoa/rust-learning/





3) Community Support

www.rust-lang.org/community.html

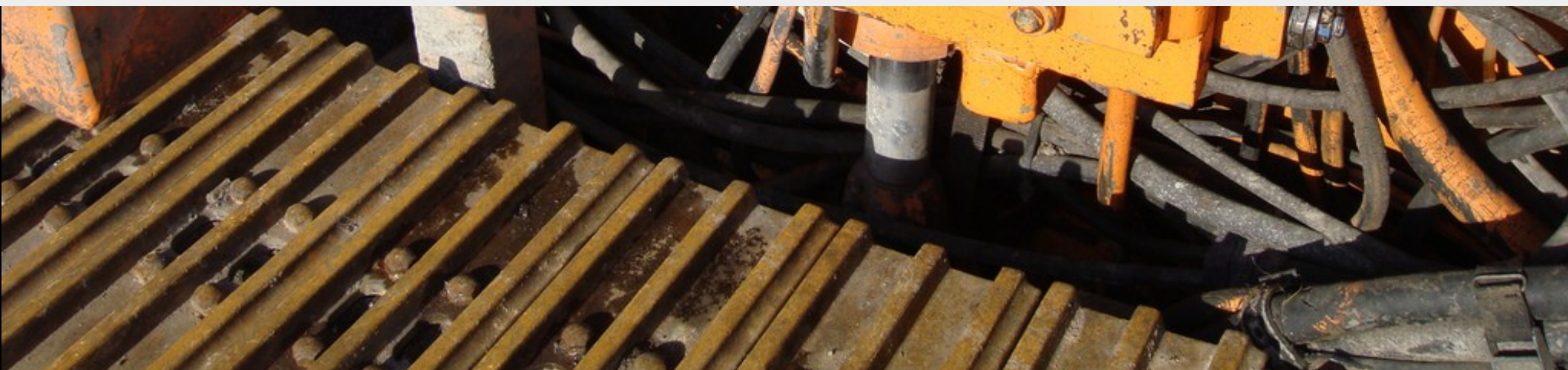
users.rust-lang.org

#rust-beginners on irc.mozilla.org



4) Write Real Code

**GitHub search “is:open is:issue language:rust”
starters.servo.org**





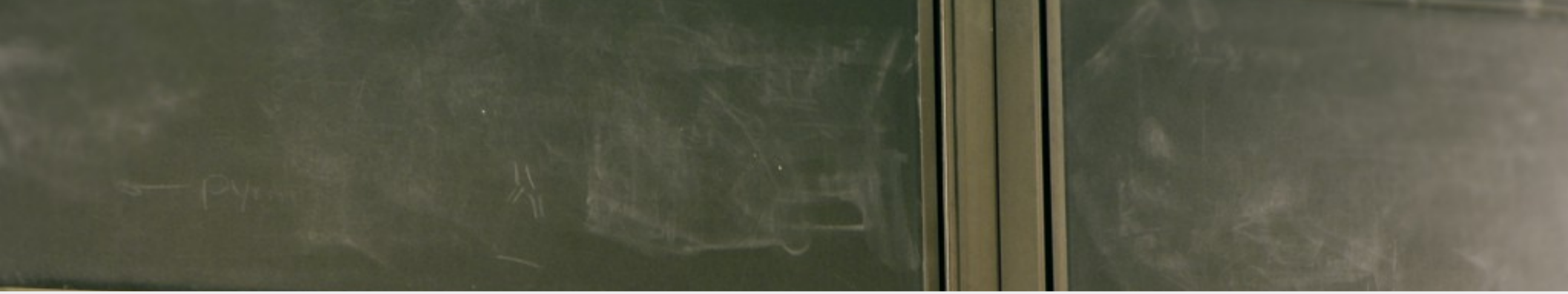
5) Read The Books

doc.rust-lang.org/book

O'Reilly, Programming Rust

Packt Publishing, Rust Essentials





6) Examine Examples

rustbyexample.com

Carol's rustlings

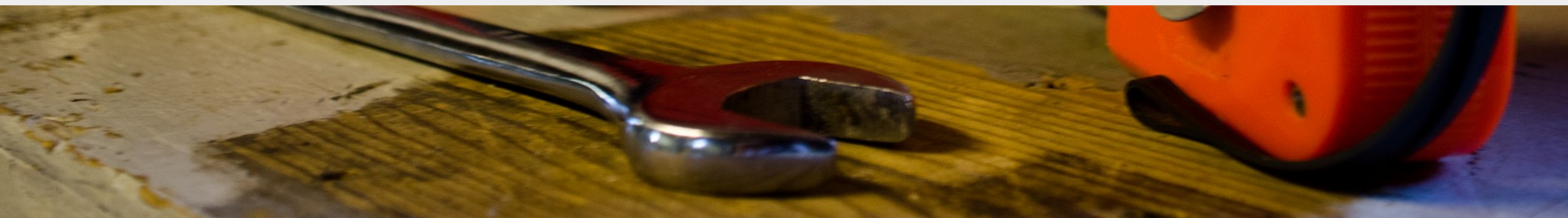
Rosetta Code





7) Find Good Tools

play.rust-lang.org
Clippy and Rustfmt
IDE support



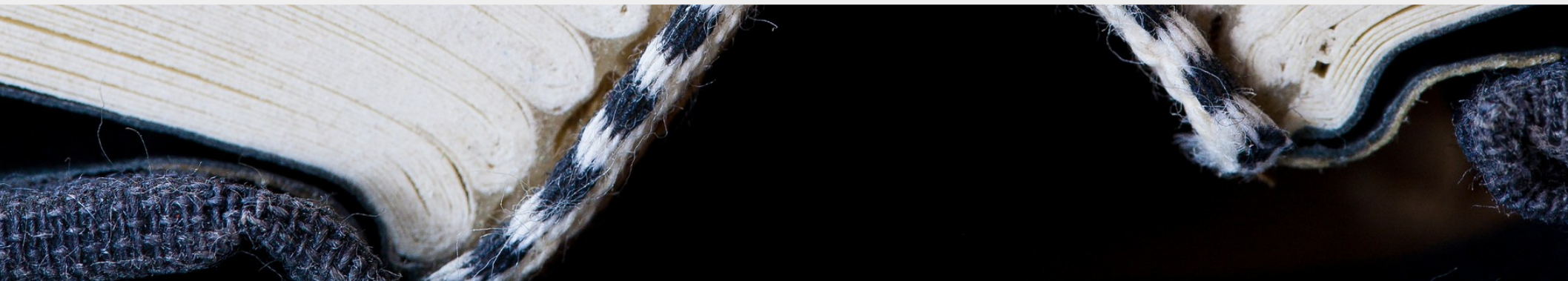


8) Read Real Code

crates.io popular libraries

The Rust Compiler

Rust in Production





9) Write Toy Programs

Rust by Example

exercism.io

Project Euler

Hackerrank



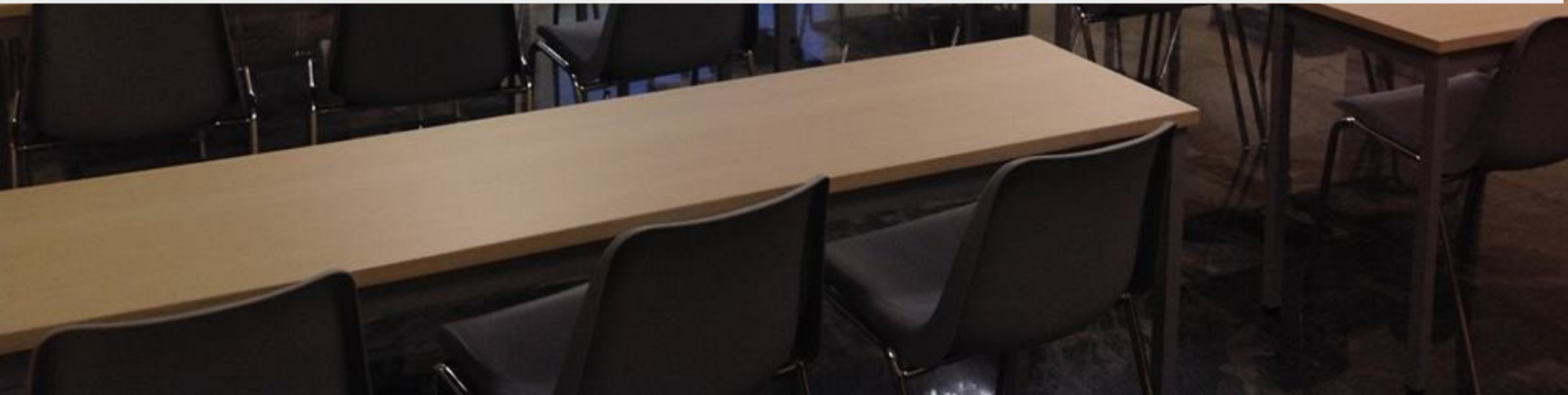
10) Google & Stackoverflow

Stackoverflow Rust tag



11) Watch Lectures & Courses

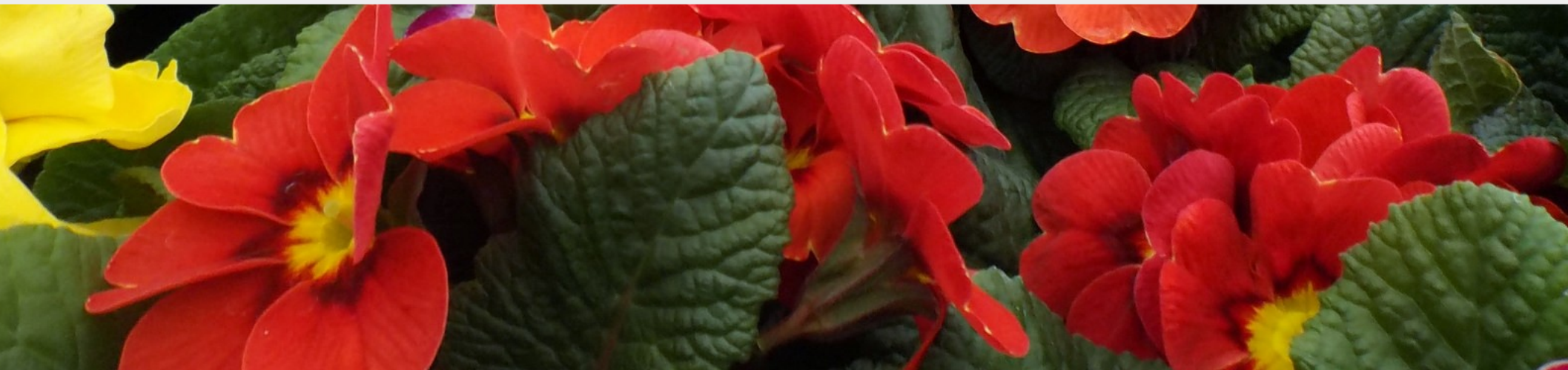
rust-learning list

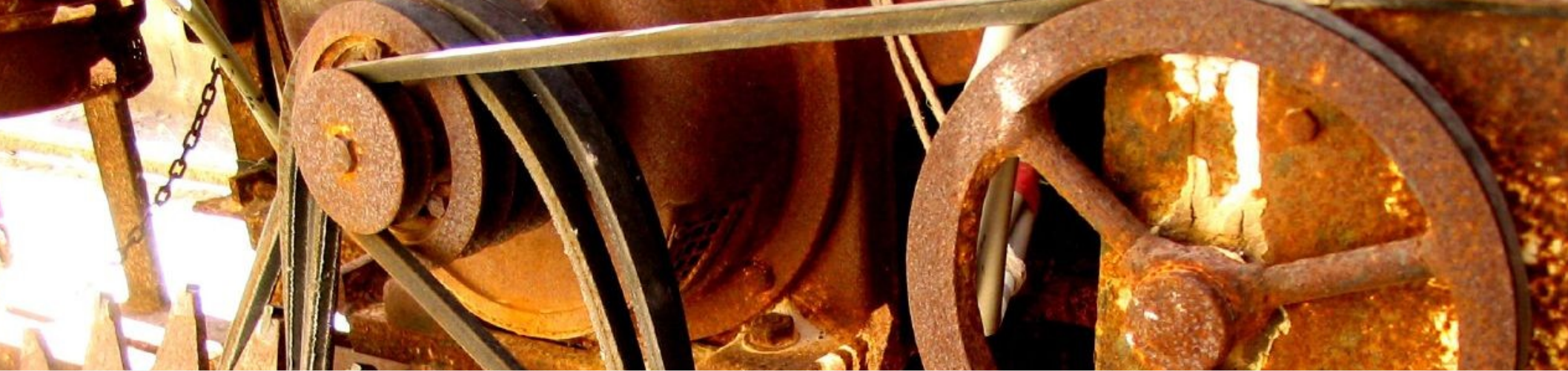




12) Maintain Enthusiasm

**Rust subreddit
This Week In Rust**





Errors are here to help



Basic Syntax

```
1 // Main takes no arguments and returns nothing
2 fn main(){
3     // The function body is the *scope* inside these curly brace
4     // Create a variable. It owns a string.
5     let what_to_say = "Hello World";
6     // Meet print syntax
7     println!("This program says {}", what_to_say);
8 }
```

This program says Hello World



Program ended.

Scope Errors!

```
1 ▾ fn not_main(){  
2     let what_to_say = "Hello World";  
3 }  
4 ▾ fn main(){  
5     println!("This program says {}", what_to_say);  
6 }
```

<anon>:5:38: 5:49 **error: unresolved name** `what\_to\_say` [\[E0425\]](#) ✕

<anon>:5 println!("This program says {}", what_to_say);
 ^~~~~~

Punctuation Errors

```
1 ▾ fn main(){  
2     let what_to_say = "Hello World"  
3     println!("This program says {}", what_to_say);  
4 }
```

```
<anon>:3:5: 3:12 error: expected one of `.`, `;`, or an  
operator, found `println`  
<anon>:3      println!("This program says {}", what_to_say);
```


Unused Variables

```
1 fn main(){  
2     let what_to_say = "Hello World";  
3     println!("Hello");  
4 }
```

```
<anon>:2:9: 2:20 warning: unused variable: `what_to_say`,  
#[warn(unused_variables)] on by default  
<anon>:2      let what_to_say = "Hello World";
```


Hey, Pythonistas!

```
1 ▾ fn main(){let what_to_say="Hello World";println!  
2  ("This program says {}",what_to_say);}
```

This program says Hello World

Hey, Pythonistas!

```
1 fn
2 main
3 (
4   ){
5   let what_to_say
6       =
7       "Hello World"
8   ;println
9   ! (
10      "This program says {}"
11      , what_to_say
12  ) ;      }
```

This program says Hello World

Program ended.

Primitive types (built-in)

- `bool`
- `char`
- `i8, i16, i32, i64`
- `u8, u16, u32, u64`

- `f32, f64`
- `isize, usize`
- `str`
- `tuple`

<https://doc.rust-lang.org/book/primitive-types.html>

Function Type Signatures

```
fn function name  
  ( name : type ,  
    name : type )  
  -> result type {  
    ...  
  }
```


Function with a type signature

```
1 ▾ fn sum(a: i32, b: i32, c: i32) -> i32{  
2     a + b + c  
3 }  
4 |  
5 ▾ fn main(){  
6     println!("A sum is {}", sum(1, 2, 3))  
7 }
```

A sum is 6

Program ended.

Synonymous return

```
1 ▾ fn sum(a: i32, b: i32, c: i32) -> i32{  
2     return a + b + c;  
3 }  
4  
5 ▾ fn main(){  
6     println!("A sum is {}", sum(1, 2, 3))  
7 }
```

A sum is 6

Program ended.

Type Errors

```
1 ▾ fn sum(a: i32, b: i32, c: i32) -> i32{  
2     return a + b + c;  
3 }  
4  
5 ▾ fn main(){  
6     println!("A sum is {}", sum(1.1, 2.3, 3.2))  
7 }
```

```
<anon>:6:33: 6:36 error: mismatched types:  
    expected `i32`,  
    found `_`  
(expected i32,  
    found floating-point variable) [E0308]
```

Anything you can add...

<https://doc.rust-lang.org/std/ops/trait.Add.html>



std::ops

Structs

Range

RangeFrom

RangeFull

RangeTo

Traits

Click or press 'S' to search, '?' for more options...

Trait std::ops::Add

[\[-\]](#) [\[src\]](#)

```
pub trait Add<RHS = Self> {  
    type Output;  
    fn add(self, rhs: RHS) -> Self::Output;  
}
```

[\[+\]](#) Expand description

Associated Types

[\[-\]](#) type Output

The resulting type after applying the `+` operator

Traits

```
1 use std::ops::Add;
2
3 fn sum<T: Add<Output=T>>(a: T, b: T, c: T) -> T{
4     return a + b + c;
5 }
6
7 fn main(){
8     println!("A sum is {}", sum(1.1, 2.3, 3.2))
9 }
```

A sum is 6.6

Program ended.

Additional Resources

- <https://doc.rust-lang.org/book/traits.html>
- <http://blog.rust-lang.org/2015/05/11/traits.html>
- <http://rustbyexample.com/trait.html>
- <http://pcwalton.github.io/blog/2012/08/08/a-gentle-introduction-to-traits-in-rust/>

Ownership Rules

- No borrow may outlive value's owner
- Exactly 1 mutable reference (&mut T)
 - OR
- As many read-only references (&T) as you want

Giving Away a Value

```
1 ▾ fn main() {  
2     let mystring = "I'm immutable!".to_string();  
3     println!("{}", mystring);  
4     let mut yarn = mystring; //yarn now owns the value  
5     println!("{}", yarn);  
6     yarn = "I have been mutated".to_string();  
7     println!("{}", yarn);  
8 }  
9 |
```

I'm immutable!
I'm immutable!
I have been mutated

Program ended.

Given away means gone

```
1 ▾ fn main() {  
2     let mystring = "I'm immutable!".to_string();  
3     println!("{}", mystring);  
4     let mut yarn = mystring; //yarn now owns the value  
5     println!("{}", mystring);  
6 }  
7
```

<anon>:5:20: 5:28 **error**: use of moved value: `mystring` [E0382] ×

<anon>:5 println!("{}", mystring);
 ^{^~~~~~}

<std macros>:2:25: 2:56 **note**: in this expansion of format_args!

<std macros>:3:1: 3:54 **note**: in this expansion of print!

Borrow the value

```
1 fn main() {  
2     let mystring = "I'm immutable!".to_string();  
3     println!("{}", mystring);  
4     let yarn = &mystring;  
5     println!("{}", mystring);  
6     println!("{}", yarn);  
7 }  
8
```

I'm immutable!
I'm immutable!
I'm immutable!

Program ended.

Make a mutable copy

```
1 ▾ fn main() {  
2     let mystring = "I'm immutable!".to_string();  
3     println!("{}", mystring);  
4     let mut yarn = mystring.clone();  
5     println!("{}", mystring);  
6     println!("{}", yarn);  
7     yarn = "I have been mutated".to_string();  
8     println!("{}", yarn);  
9 }
```

```
I'm immutable!  
I'm immutable!  
I'm immutable!  
I have been mutated
```

Review

- Only owner can access value
- 1 mutable reference or unlimited read-only references to each value
- Borrow may not outlive owner

Additional Resources

- <http://doc.rust-lang.org/stable/book/ownership.html>
- <http://doc.rust-lang.org/stable/book/references-and-borrowing.html>
- Why the `to_string()`?
<http://hermanradtke.com/2015/05/03/string-vs-str-in-rust-functions.html>
- <http://rustbyexample.com/trait/clone.html>
- https://www.reddit.com/r/rust/comments/2xxjda/when_should_my_type_be_copy/



It Gets Better





Thank You

talks.edunham.net/oscon2016

Photos at <https://www.flickr.com/photos/143305168@N08/favorites>

