

2023.08.26 LinkedIn Course – Eve Porcello – Rust for JavaScript Developers (~1 hr 17 mins)

Saturday, August 26th, 2023, 8:04am

Originally released on Friday, July 7th, 2023 and available on LinkedIn at <https://www.linkedin.com/learning/rust-for-javascript-developers/learning-rust-from-scratch>

The screenshot shows the LinkedIn Learning interface. The left sidebar displays a tree view of course contents under 'Rust for JavaScript Developers'. The main content area shows the 'Overview' tab for the course, featuring a bio of the instructor, Eve Porcello, a 'Following on LinkedIn' button, and course details like duration (1h 17m), level (Intermediate), and release date (7/7/2023). A video player at the bottom of the overview page shows the video is 0:00 / 0:48.

- Contents
 - Introduction
 - Learning Rust from scratch (48s)
 - Why learn Rust? (2m 16s)
 - Setting up VSCode (1m 54s)
 - Chapter Quiz (2 questions)
 - 1. Creating Rust Programs
 - Installing Rust on a Mac (1m 44s)
 - Installing Rust on a PC (1m 8s)
 - Using Cargo (3m 30s)
 - Chapter Quiz (2 questions)
 - 2. Rust Primitives
 - Creating strings (3m 30s)
 - Assigning variables (5m 21s)
 - Working with constants (2m 14s)
 - Writing comments (1m 59s)
 - Using booleans (1m 59s)

Introduction

Learning Rust from scratch

Why learn Rust?

Setting up VSCode

Chapter Quiz

Rust for JavaScript Developers
Chapter Quiz

Up next
Installing Rust on a Mac
1m 44s

Question 1 of 2
Why might you choose to work with Rust?

Flexibility
 all of these answers
 Correct
 Performance
 Small file sizes

Next question

Rust for JavaScript Developers
Chapter Quiz

Question 2 of 2
Where can you find VSCode themes?

Styles
 Window
 View
 Settings
 Correct

Next

Creating Rust Programs

Installing Rust on a Mac

Install Rust globally on your machine:

```
% curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
% source "$HOME/.cargo/env"
```

Verify Rust has been installed successfully:

```
% rustc --version  
rustc 1.72.0 (5680fa18f 2023-08-23)
```

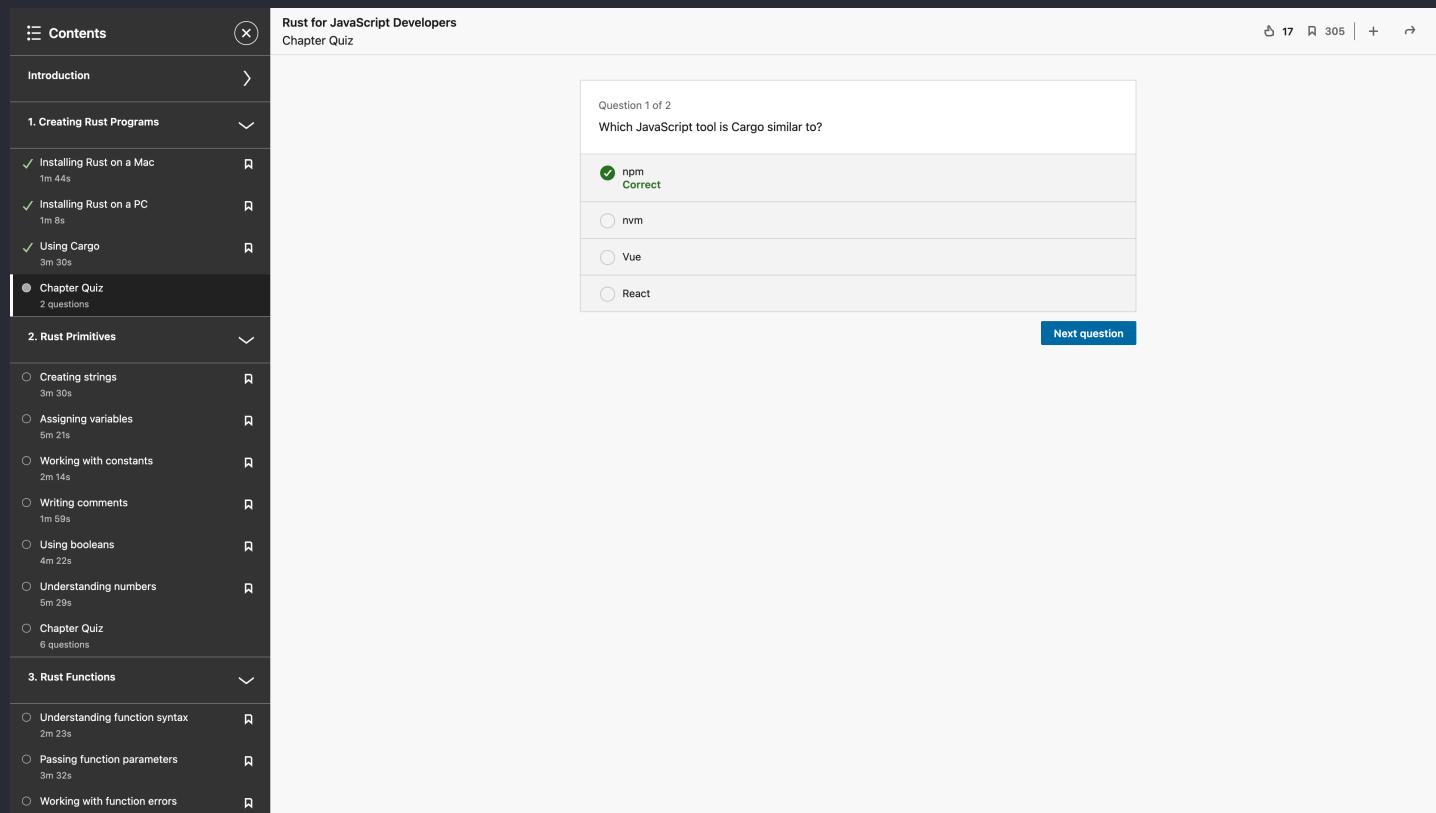
Installing Rust on a PC

Using Cargo

Think of **Cargo** as somewhat comparable to **NPM** in JavaScript land.

```
# Create a new Rust application using cargo  
% cargo new camping-app  
  
# Navigate to the source file directory for our app  
% cd camping-app/src  
  
# Note that we have a main.rs file.  
## Compile it using rustc - the Rust compiler  
% rustc main.rs  
## Run the binary  
% ./main  
Hello, world!
```

Chapter Quiz



The screenshot shows a dark-themed user interface for a chapter quiz. On the left is a sidebar with a tree-like navigation structure:

- Contents** (with a close button)
- Introduction**
- 1. Creating Rust Programs** (expanded)
 - ✓ **Installing Rust on a Mac** (1m 44s)
 - ✓ **Installing Rust on a PC** (1m 8s)
 - ✓ **Using Cargo** (3m 30s)
- Chapter Quiz** (selected, 2 questions)
- 2. Rust Primitives** (expanded)
 - **Creating strings** (3m 30s)
 - **Assigning variables** (5m 21s)
 - **Working with constants** (2m 14s)
 - **Writing comments** (1m 59s)
 - **Using booleans** (4m 22s)
 - **Understanding numbers** (5m 29s)
- 3. Rust Functions** (expanded)
 - **Understanding function syntax** (2m 23s)
 - **Passing function parameters** (3m 32s)
 - **Working with function errors** (2m 23s)

The main area displays the first question of the quiz:

Question 1 of 2
Which JavaScript tool is Cargo similar to?

npm
Correct

nvm

Vue

React

Next question

Contents (X)

Rust for JavaScript Developers

Chapter Quiz

Introduction >

1. Creating Rust Programs < ▾

- ✓ Installing Rust on a Mac 1m 44s
- ✓ Installing Rust on a PC 1m 8s
- ✓ Using Cargo 3m 30s
- ✓ Chapter Quiz 2 questions

2. Rust Primitives < ▾

- Creating strings 3m 30s
- Assigning variables 5m 21s
- Working with constants 2m 14s
- Writing comments 1m 59s
- Using booleans 4m 22s
- Understanding numbers 5m 29s
- Chapter Quiz 6 questions

3. Rust Functions < ▾

- Understanding function syntax 2m 23s
- Passing function parameters 3m 32s
- Working with function errors 5m 10s

Question 2 of 2
What is the tool that you use to install Rust?

rust-get

rusting

rustup **Correct**

rust-down

Next

Rust Primitives

Creating strings

```
fn main() {
    println!("Hello, world!");
    println!("Hey it's all happening!");
}
```

Assigning variables

```
fn main() {
    let mut destination = "Crater Lake";
    println!("{} is the deepest lake in the US", destination);
    destination = "Sparks Lake";
    println!("{} is also a lake in the US", destination);
}
```

If you want to have a multi-word variable name in Rust, use snake case:

```
fn main() {
    let mut destination_lake = "Crater Lake"
}
```

Working with constants

Note that Rust will warn us to use capitalized names for constants and to make sure we have a type defined for our constant value:

```
fn main() {
    const DESTINATION: &str = "Crater Lake";
    println!("{}", DESTINATION);
}
```

Writing comments

```
/*
In this file, we set a destination!
*/

/// This function is used throughout the entire project to state where the destination is.
fn main() {
    const DESTINATION: &str = "Crater Lake";
    println!("{}", DESTINATION);
}
```

Generate documentation based on your comments

We can use `cargo doc` to generate documentation for our application based on our comments.

```
% cargo doc
Documenting camping-app v0.1.0 (/Users/rob/repos/linkedin-course-rust-for-javascript-developers-eve-porcello/Exercise
Files/Ch02/02_04/finished/camping-app)
Finished dev [unoptimized + debuginfo] target(s) in 0.69s
```

We can see the generated HTML documentation underneath the `target/doc` directory of our Rust application - e.g., `target/doc/index.html`

Using booleans

Notice that we didn't add a type annotation here. Rust will infer the variable as a boolean if it is assigned `true` or `false`.

The screenshot shows the Visual Studio Code interface. On the left is the sidebar with a tree view of course content. In the center is the code editor with the following code:

```
fn main() {
    let summer = true;
    let winter = true;
    println!("Summer is the best season: {}", summer);
    println!("Winter is also good: {}", winter);
}
```

Below the code editor is the terminal window showing the output of running the program:

```
eveporcello src $ rustc main.rs
eveporcello src $ ./main
Summer is the best season: true
Winter is also good: true
eveporcello src $
```

At the bottom, there is a snippet of code from the course notes:

```
fn main() {
    let summer: bool = false;
    let winter: bool = false;

    if summer {
        println!("go biking!");
    } else if winter {
        println!("go skiing");
    } else {
        println!("do something else");
    }
}
```

```
fn main() {
    let summer: bool = false;
    let winter: bool = false;

    if summer {
        println!("go biking!");
    } else if winter {
        println!("go skiing");
    } else {
        println!("do something else");
    }
}
```

```
}
```

Understanding numbers

The screenshot shows a browser window with the URL doc.rust-lang.org/std/index.html#primitives. The page title is "Primitive Types". On the left, there is a sidebar with a tree view of Rust documentation categories. The "std" crate is selected. The main content area displays detailed information about various primitive types like never, Experimental, array, bool, char, f32, f64, fn, i8, i16, i32, i64, i128, isize, pointer, reference, slice, str, tuple, u8, u16, u32, u64, and u128. A note about isize states: "A dynamically-sized view into a contiguous sequence, [T]. Contiguous here means that elements are laid out so that every element is the same distance from its neighbors. String slices." A "LinkedIn Learning" logo is visible in the bottom right corner.

```
fn main() {
    # A 64-bit unsigned integer
    let days: u64 = 5;
    println!("{} days", days);

    # A 32-bit floating point number
    let backpack_weight: f32 = 28.5;
    println!(
        "hiking for {} days with a {} lb. pack",
        days, backpack_weight
    );

    let meals = days * 3;
    println!("{} meals", meals);
}
```

Chapter Quiz

Contents X

Rust for JavaScript Developers
Chapter Quiz

Introduction >

1. Creating Rust Programs >

2. Rust Primitives ▾

- ✓ Creating strings 3m 30s
- ✓ Assigning variables 5m 21s
- ✓ Working with constants 2m 14s
- ✓ Writing comments 1m 59s
- ✓ Using booleans 4m 22s
- ✓ Understanding numbers 5m 29s

Chapter Quiz 6 questions

3. Rust Functions ▾

- Understanding function syntax 2m 23s
- Passing function parameters 3m 32s
- Working with function errors 4m 48s
- Returning values from functions 1m 29s
- Creating arrays 3m 32s
- Looping through arrays 3m 10s
- Chapter Quiz 6 questions

Question 1 of 6

What is the type annotation for a boolean?

b

true/false

bool Correct

boolean

Next question

Contents X

Rust for JavaScript Developers
Chapter Quiz

Introduction >

1. Creating Rust Programs >

2. Rust Primitives ▾

- ✓ Creating strings 3m 30s
- ✓ Assigning variables 5m 21s
- ✓ Working with constants 2m 14s
- ✓ Writing comments 1m 59s
- ✓ Using booleans 4m 22s
- ✓ Understanding numbers 5m 29s

Chapter Quiz 6 questions

3. Rust Functions ▾

- Understanding function syntax 2m 23s
- Passing function parameters 3m 32s
- Working with function errors 4m 48s
- Returning values from functions 1m 29s
- Creating arrays 3m 32s
- Looping through arrays 3m 10s
- Chapter Quiz 6 questions

Question 2 of 6

How do you create documentation comments in Rust?

???

/?

/// Correct

/*

Next question

Contents (X)

Rust for JavaScript Developers
Chapter Quiz

17 305 | + ↗

- Introduction >
- 1. Creating Rust Programs >
- 2. Rust Primitives >
- ✓ Creating strings 3m 30s <input checked="" type="radio"/>
- ✓ Assigning variables 5m 21s <input type="radio"/>
- ✓ Working with constants 2m 14s <input type="radio"/>
- ✓ Writing comments 1m 59s <input type="radio"/>
- ✓ Using booleans 4m 22s <input type="radio"/>
- ✓ Understanding numbers 5m 29s <input type="radio"/>
- Chapter Quiz 6 questions
- 3. Rust Functions >
- <input type="radio"/> Understanding function syntax 2m 23s <input checked="" type="radio"/>
- <input type="radio"/> Passing function parameters 3m 32s <input checked="" type="radio"/>
- <input type="radio"/> Working with function errors 4m 48s <input checked="" type="radio"/>
- <input type="radio"/> Returning values from functions 1m 29s <input checked="" type="radio"/>
- <input type="radio"/> Creating arrays 3m 32s <input checked="" type="radio"/>
- <input type="radio"/> Looping through arrays 3m 10s <input checked="" type="radio"/>
- <input type="radio"/> Chapter Quiz 6 questions

Question 3 of 6

How can you tell the difference between a constant and a regular variable in Rust?

Using camel case

ALL CAPS
Correct

Believing strongly enough that it will remain constant

all lowercase

[Next question](#)

Contents (X)

Rust for JavaScript Developers
Chapter Quiz

17 305 | + ↗

- Introduction >
- 1. Creating Rust Programs >
- 2. Rust Primitives >
- ✓ Creating strings 3m 30s <input checked="" type="radio"/>
- ✓ Assigning variables 5m 21s <input type="radio"/>
- ✓ Working with constants 2m 14s <input type="radio"/>
- ✓ Writing comments 1m 59s <input type="radio"/>
- ✓ Using booleans 4m 22s <input type="radio"/>
- ✓ Understanding numbers 5m 29s <input type="radio"/>
- Chapter Quiz 6 questions
- 3. Rust Functions >
- <input type="radio"/> Understanding function syntax 2m 23s <input checked="" type="radio"/>
- <input type="radio"/> Passing function parameters 3m 32s <input checked="" type="radio"/>
- <input type="radio"/> Working with function errors 4m 48s <input checked="" type="radio"/>
- <input type="radio"/> Returning values from functions 1m 29s <input checked="" type="radio"/>
- <input type="radio"/> Creating arrays 3m 32s <input checked="" type="radio"/>
- <input type="radio"/> Looping through arrays 3m 10s <input checked="" type="radio"/>
- <input type="radio"/> Chapter Quiz 6 questions

Question 4 of 6

How do you make a variable mutable?

Add the function keyword.

Convert to a string.

Add the mut keyword.
Correct

You cannot make a variable mutable in Rust.

[Next question](#)

Contents

- Introduction
- 1. Creating Rust Programs
 - 2. Rust Primitives
 - ✓ Creating strings 3m 30s
 - ✓ Assigning variables 5m 21s
 - ✓ Working with constants 2m 14s
 - ✓ Writing comments 1m 59s
 - ✓ Using booleans 4m 22s
 - ✓ Understanding numbers 5m 29s
 - Chapter Quiz 6 questions
- 3. Rust Functions
 - Understanding function syntax 2m 23s
 - Passing function parameters 3m 32s
 - Working with function errors 4m 48s
 - Returning values from functions 1m 29s
 - Creating arrays 3m 32s
 - Looping through arrays 3m 10s
 - Chapter Quiz 6 questions

Rust for JavaScript Developers
Chapter Quiz

Question 5 of 6
Which character do you use to denote a string?

Backticks

Double quotes
Correct

Exclamation marks

Single quotes

[Next question](#)

Contents

- Introduction
- 1. Creating Rust Programs
 - 2. Rust Primitives
 - ✓ Creating strings 3m 30s
 - ✓ Assigning variables 5m 21s
 - ✓ Working with constants 2m 14s
 - ✓ Writing comments 1m 59s
 - ✓ Using booleans 4m 22s
 - ✓ Understanding numbers 5m 29s
 - Chapter Quiz 6 questions
- 3. Rust Functions
 - Understanding function syntax 2m 23s
 - Passing function parameters 3m 32s
 - Working with function errors 4m 48s
 - Returning values from functions 1m 29s
 - Creating arrays 3m 32s
 - Looping through arrays 3m 10s
 - Chapter Quiz 6 questions

Rust for JavaScript Developers
Chapter Quiz

Question 6 of 6
What does a u stand for when annotating a value's type?

unsupported

underlined

unbelievable

unsigned
Correct

[Next](#)

Introduction >

1. Creating Rust Programs >

2. Rust Primitives ▾

✓ Creating strings 3m 30s

✓ Assigning variables 5m 21s

✓ Working with constants 2m 14s

✓ Writing comments 1m 59s

✓ Using booleans 4m 22s

✓ Understanding numbers 5m 29s

✓ Chapter Quiz 6 questions

3. Rust Functions ▾

○ Understanding function syntax 2m 23s

○ Passing function parameters 3m 32s

○ Working with function errors 4m 48s

○ Returning values from functions 1m 29s

○ Creating arrays 3m 32s

○ Looping through arrays 3m 10s

○ Chapter Quiz



You answered 6 of 6 questions correctly.
You successfully completed all questions in this quiz.

[Review all answers](#)[Continue watching](#)

Rust Functions

Understanding function syntax

```
fn calculate_distance() {
    let days: u64 = 5;
    println!("{} days", days);
    let distance: u64 = 10;
    println!("{} miles total distance", distance * days);
}

fn main() {
    calculate_distance();
}
```

Passing function parameters

```
fn calculate_distance(days: u64, distance: u64) -> u64 {
    let total_miles = days * distance;
    return total_miles;
}

fn main() {
    let result = calculate_distance(5, 10);
    println!("{} miles", result);
}
```

Working with function errors

```
# We are returning a result that will either be an unsigned 64-bit integer or a string in the event of an error or unexpected data
fn calculate_distance(days: u64, distance: u64) -> Result<u64, String> {
    if days == 0 {
        Err("Cannot go on a zero-day hike".to_string())
    } else {
        Ok(days * distance)
    }
}
```

```

fn main() {
    let result = calculate_distance(0, 10);

    # Invoke the appropriate println! statement based on the result we receive
    match result {
        Ok(miles) => println!("Miles: {}", miles),
        Err(error) => println!("Error: {}", error),
    }
}

```

Returning values from functions

Creating arrays

```

fn main() {
    let years: [i32; 3] = [1996, 2002, 2023];
    println!("{} years", years.len());
}

```

The screenshot shows a code editor interface with a sidebar on the left containing a table of contents for a Rust course. The main area displays a file named `main.rs` with the following code:

```

fn main() {
    let years: [i32; 3] = [1996, 2002, 2023];
    println!("{} years", years.len());
}

```

The terminal window below shows the output of running the program:

```

eveporcello src $ ./main
[1996, 2002, 2023]
eveporcello src $

```

The sidebar contains the following course structure:

Contents	
Introduction	>
1. Creating Rust Programs	>
2. Rust Primitives	⌄
✓ Creating strings	3m 30s
✓ Assigning variables	5m 21s
✓ Working with constants	2m 14s
✓ Writing comments	1m 59s
✓ Using booleans	4m 22s
✓ Understanding numbers	5m 29s
✓ Chapter Quiz	6 questions
3. Rust Functions	⌄
✓ Understanding function syntax	2m 23s
✓ Passing function parameters	3m 32s
✓ Working with function errors	4m 48s
✓ Returning values from functions	1m 29s
● Creating arrays	3m 32s
○ Looping through arrays	3m 10s
○ Chapter Quiz	...

At the bottom right of the sidebar, there is a "LinkedIn Learning" logo.

Contents

- Introduction
- 1. Creating Rust Programs
 - 2. Rust Primitives
 - Creating strings 3m 30s
 - Assigning variables 5m 21s
 - Working with constants 2m 14s
 - Writing comments 1m 59s
 - Using booleans 4m 22s
 - Understanding numbers 5m 29s
 - Chapter Quiz 6 questions
- 3. Rust Functions
 - Understanding function syntax 2m 23s
 - Passing function parameters 3m 32s
 - Working with function errors 4m 48s
 - Returning values from functions 1m 29s
 - Creating arrays 3m 32s
 - Looping through arrays 3m 10s
 - Chapter Quiz 6 questions

main.rs Ch03/03_05/start/camping-app/src/main.rs/ main

```
1 fn main() {  
2     let years: [i32; 3] = [1996, 2002, 2023];  
3     println!("{}", years[0]);  
4 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

zsh - src + ×

```
• eveporcello src $ rustc main.rs  
• eveporcello src $ ./main  
1996  
○ eveporcello src $ █
```

LinkedIn Learning

Contents

- Introduction
- 1. Creating Rust Programs
- 2. Rust Primitives
 - Creating strings 3m 30s
 - Assigning variables 5m 21s
 - Working with constants 2m 14s
 - Writing comments 1m 59s
 - Using booleans 4m 22s
 - Understanding numbers 5m 29s
 - Chapter Quiz 6 questions
- 3. Rust Functions
 - Understanding function syntax 2m 23s
 - Passing function parameters 3m 32s
 - Working with function errors 4m 48s
 - Returning values from functions 1m 29s
 - Creating arrays 3m 32s
 - Looping through arrays 3m 10s
 - Chapter Quiz 6 questions

main.rs Ch03/03_05/start/camping-app/src/main.rs/ main

```
1 fn main() {  
2     let years: [i32; 3] = [1996, 2002, 2023];  
3     println!("{}", years[0]);  
4 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

zsh - src + ×

```
• eveporcello src $ rustc main.rs  
• eveporcello src $ ./main  
1996  
● eveporcello src $ rustc main.rs  
● eveporcello src $ ./main  
1996  
○ eveporcello src $ █
```

LinkedIn Learning

Looping through arrays

The screenshot shows a dark-themed IDE interface. On the left, a sidebar lists course contents under 'Ch03/03_06/start/camping-app/src/main.rs'. The main area displays the code for 'main.rs':

```
1 fn main() {  
2     let years: [i32; 3] = [1996, 2002, 2023];  
3     for year in years.iter() {  
4         println!("The year is {}", year);  
5     }  
6 }  
7
```

Below the code, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab shows the following command-line session:

- eveporcello src \$ rustc main.rs
- eveporcello src \$./main
1996
- eveporcello src \$ rustc main.rs
- eveporcello src \$./main
The year is 1996
The year is 2002
The year is 2023
- eveporcello src \$ █

In the bottom right corner, there is a 'LinkedIn Learning' logo.

BONUS: Create a countdown using a while loop

```
fn main() {  
    let mut new_year_countdown = 10;  
  
    while new_year_countdown > 0 {  
        println!("{}new_year_countdown");  
        new_year_countdown -= 1;  
    }  
}
```

Chapter Quiz

Rust for JavaScript Developers

Chapter Quiz

Question 1 of 6

Where does an index start in Rust?

2

0 **Correct**

1

-1

Next question

Contents

- Introduction
- 1. Creating Rust Programs
- 2. Rust Primitives
- 3. Rust Functions
 - ✓ Understanding function syntax 2m 23s
 - ✓ Passing function parameters 3m 32s
 - ✓ Working with function errors 4m 48s
 - ✓ Returning values from functions 1m 29s
 - ✓ Creating arrays 3m 32s
 - ✓ Looping through arrays 3m 10s
- Chapter Quiz 6 questions

4. Complex Data Types in Rust

- Building tuples 3m 28s
- Creating vectors 2m 31s
- Understanding enums 3m 17s
- Working with structs 2m 38s
- Chapter Quiz 4 questions

5. Files and Patterns

Rust for JavaScript Developers

Chapter Quiz

Question 2 of 6

Arrays can hold:

the same type **Correct**

only numbers

different types

only strings

Next question

Contents

- Introduction
- 1. Creating Rust Programs
- 2. Rust Primitives
- 3. Rust Functions
 - ✓ Understanding function syntax 2m 23s
 - ✓ Passing function parameters 3m 32s
 - ✓ Working with function errors 4m 48s
 - ✓ Returning values from functions 1m 29s
 - ✓ Creating arrays 3m 32s
 - ✓ Looping through arrays 3m 10s
- Chapter Quiz 6 questions

4. Complex Data Types in Rust

- Building tuples 3m 28s
- Creating vectors 2m 31s
- Understanding enums 3m 17s
- Working with structs 2m 38s
- Chapter Quiz 4 questions

5. Files and Patterns

Rust for JavaScript Developers

Chapter Quiz

Question 3 of 6

What is a function that returns an error in Rust?

Error()

Exception()

Err()
Correct

E()

[Next question](#)

Contents

- Introduction
- 1. Creating Rust Programs
- 2. Rust Primitives
- 3. Rust Functions
 - ✓ Understanding function syntax 2m 23s
 - ✓ Passing function parameters 3m 32s
 - ✓ Working with function errors 4m 48s
 - ✓ Returning values from functions 1m 29s
 - ✓ Creating arrays 3m 32s
 - ✓ Looping through arrays 3m 10s
- Chapter Quiz 6 questions
- 4. Complex Data Types in Rust
 - Building tuples 3m 28s
 - Creating vectors 2m 31s
 - Understanding enums 3m 17s
 - Working with structs 2m 38s
- Chapter Quiz 4 questions
- 5. Files and Patterns

Rust for JavaScript Developers

Chapter Quiz

Question 4 of 6

When you see `println!` in a Rust application, it is a:

cry for help

function

return

macro
Correct

[Next question](#)

Contents

- Introduction
- 1. Creating Rust Programs
- 2. Rust Primitives
- 3. Rust Functions
 - ✓ Understanding function syntax 2m 23s
 - ✓ Passing function parameters 3m 32s
 - ✓ Working with function errors 4m 48s
 - ✓ Returning values from functions 1m 29s
 - ✓ Creating arrays 3m 32s
 - ✓ Looping through arrays 3m 10s
- Chapter Quiz 6 questions
- 4. Complex Data Types in Rust
 - Building tuples 3m 28s
 - Creating vectors 2m 31s
 - Understanding enums 3m 17s
 - Working with structs 2m 38s
- Chapter Quiz 4 questions
- 5. Files and Patterns

Rust for JavaScript Developers
Chapter Quiz

Question 5 of 6
What does the `->` mean in Rust?

a lambda function
 defines the type of the return value
Correct
 an inline function
 defines the parameters

Next question

Contents X

- Introduction >
- 1. Creating Rust Programs >
- 2. Rust Primitives >
- 3. Rust Functions >
 - ✓ Understanding function syntax 2m 23s
 - ✓ Passing function parameters 3m 32s
 - ✓ Working with function errors 4m 48s
 - ✓ Returning values from functions 1m 29s
 - ✓ Creating arrays 3m 32s
 - ✓ Looping through arrays 3m 10s
- Chapter Quiz 6 questions
- 4. Complex Data Types in Rust >
 - Building tuples 3m 2s
 - Creating vectors 2m 31s
 - Understanding enums 3m 17s
 - Working with structs 2m 38s
- Chapter Quiz 4 questions
- 5. Files and Patterns >

Rust for JavaScript Developers
Chapter Quiz

Question 6 of 6
How do you declare a function in Rust?

fun keyword
 function keyword
 function! keyword
 fn keyword
Correct

Next

Contents X

- Introduction >
- 1. Creating Rust Programs >
- 2. Rust Primitives >
- 3. Rust Functions >
 - ✓ Understanding function syntax 2m 23s
 - ✓ Passing function parameters 3m 32s
 - ✓ Working with function errors 4m 48s
 - ✓ Returning values from functions 1m 29s
 - ✓ Creating arrays 3m 32s
 - ✓ Looping through arrays 3m 10s
- ✓ Chapter Quiz 6 questions
- 4. Complex Data Types in Rust >
 - Building tuples 3m 2s
 - Creating vectors 2m 31s
 - Understanding enums 3m 17s
 - Working with structs 2m 38s
- Chapter Quiz 4 questions
- 5. Files and Patterns >

Contents

Rust for JavaScript Developers
Chapter Quiz

Introduction >

1. Creating Rust Programs >

2. Rust Primitives >

3. Rust Functions >

- ✓ Understanding function syntax 2m 23s
- ✓ Passing function parameters 3m 32s
- ✓ Working with function errors 4m 48s
- ✓ Returning values from functions 1m 29s
- ✓ Creating arrays 3m 32s
- ✓ Looping through arrays 3m 10s
- ✓ Chapter Quiz 6 questions

4. Complex Data Types in Rust >

- Building tuples 3m 2s
- Creating vectors 2m 31s
- Understanding enums 3m 17s
- Working with structs 2m 38s
- Chapter Quiz 4 questions

5. Files and Patterns



You answered 6 of 6 questions correctly.
You successfully completed all questions in this quiz.

[Review all answers](#) [Continue watching](#)

Complex Data Types in Rust

Building tuples

Contents

1m 29s

- ✓ Creating arrays 3m 32s
- ✓ Looping through arrays 3m 10s
- ✓ Chapter Quiz 6 questions

4. Complex Data Types in Rust >

- Building tuples 3m 2s
- Creating vectors 2m 31s
- Understanding enums 3m 17s
- Working with structs 2m 38s
- Chapter Quiz 4 questions

5. Files and Patterns

○ Reading files 3m 5s

○ Writing files 3m 29s

○ Using match expressions 3m 12s

○ Destructuring assignment 2m 31s

○ Chapter Quiz 4 questions

Conclusion

○ Building for the future with Rust 1m

main.rs Ch04/04_01/start/camping-app/src/main.rs/...

```

1 fn main() {
2     let person_info = ("Eve", 38, "P");
3     println!("{} is {}, and her last initial is {}.", 
4         person_info.0, person_info.1, person_info.2
5     );
6 }
7
8

```

Debug Console (⌃⌘Y)

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL
			zsh - src + ✖ ... ^ ×

```

● eveporcello src $ rustc main.rs
● eveporcello src $ ./main
  Eve is 38, and her last initial is P.
○ eveporcello src $ 

```

LinkedIn Learning

The screenshot shows a video player interface with a sidebar containing a course outline and a main area with a code editor and terminal window.

Course Outline:

- 1m 29s
- ✓ Creating arrays 3m 32s
- ✓ Looping through arrays 3m 10s
- ✓ Chapter Quiz 6 questions
- 4. Complex Data Types in Rust ▼
- ✓ Building tuples 3m 2s
 - Creating vectors 2m 31s
 - Understanding enums 3m 17s
 - Working with structs 2m 38s
 - Chapter Quiz 4 questions
- 5. Files and Patterns ▼
- Reading files 3m 5s
- Writing files 3m 29s
- Using match expressions 3m 12s
- Destructuring assignment 2m 31s
- Chapter Quiz 4 questions
- Conclusion ▼
- Building for the future with Rust 1m

Main Area:

File: main.rs Ch04/04_01/start/camping-app/src/main.rs sentence_builder

```
1 fn sentence_builder(person_info: (&str, u64, &str)) {  
2     println!(  
3         "{} is {}, and her last initial is {}.",  
4         person_info.0, person_info.1, person_info.2  
5     );  
6 }  
7  
8 fn main() {  
9     let values = ("Eve", 38, "P");  
10    sentence_builder(values)  
11 }  
12
```

Terminal:

```
eveporcello src $ rustc main.rs  
eveporcello src $ ./main  
Eve is 38, and her last initial is P.  
eveporcello src $
```

Bottom Right: LinkedIn Learning

```
fn sentence_builder(person_info: (&str, u64, &str)) {  
    println!(  
        "{} is {}, and her last initial is {}.",  
        person_info.0, person_info.1, person_info.2  
    );  
}  
  
fn main() {  
    let values = ("Eve", 38, "P");  
    sentence_builder(values);  
}
```

Creating vectors

What are vectors? "Vecs" – or vectors – are dynamically sized lists.

The screenshot shows a code editor interface with a sidebar containing a table of contents for a course. The main area displays a file named `main.rs` with the following code:

```
fn main() {
    let mut packing_list = Vec::new();
    packing_list.push("sunglasses");
    packing_list.push("sunscreen");
    packing_list.push("hat");
    println!("{}:?", packing_list);
}
```

Below the code editor is a terminal window showing the output of running the program:

```
eveporcello src $ rustc main.rs
eveporcello src $ ./main
["sunglasses", "sunscreen", "hat"]
eveporcello src $
```

The sidebar contains sections for chapters 4 and 5, along with quizzes and exercises. The footer of the interface shows the LinkedIn Learning logo.

If you're not excited about that constructor syntax, you can use the `vec!` macro instead:

```
fn main() {
    let packing_list = vec!["sunglasses", "sunscreen", "hat"];
    println!("{}:?", packing_list);
}
```

Understanding enums

```
#[derive(Debug)]
enum Steepness {
    Easy,
    Moderate,
    Difficult,
}

fn main() {
    let _calm_trail = Steepness::Easy;
    let _fun_trail = Steepness::Moderate;
    let prickly_peak_trail = Steepness::Difficult;
    println!("Steepness is {:?}", prickly_peak_trail);
}
```

Working with structs

```
struct Hiker {
    name: String,
    miles_hiked: u64,
}

fn main() {
    let jennifer = Hiker {
        name: String::from("Jennifer"),
        miles_hiked: 49,
    };
    println!("{} has hiked {} miles", jennifer.name, jennifer.miles_hiked);
}
```

Chapter Quiz

Contents

- 1. Creating Rust Programs
- 2. Rust Primitives
- 3. Rust Functions
- 4. Complex Data Types in Rust
 - ✓ Building tuples 3m 2s
 - ✓ Creating vectors 2m 31s
 - ✓ Understanding enums 3m 17s
 - ✓ Working with structs 2m 38s
- Chapter Quiz 4 questions
- 5. Files and Patterns
 - Reading files 3m 5s
 - Writing files 3m 29s
 - Using match expressions 3m 12s
 - Destructuring assignment 2m 31s
 - Chapter Quiz 4 questions
- Conclusion
- Building for the future with Rust 1m

Rust for JavaScript Developers
Chapter Quiz

Question 1 of 4
What is `Vec::new` called?

a constructor
Correct

a macro

a function

an array

[Next question](#)

Contents

- 1. Creating Rust Programs
- 2. Rust Primitives
- 3. Rust Functions
- 4. Complex Data Types in Rust
 - ✓ Building tuples 3m 2s
 - ✓ Creating vectors 2m 31s
 - ✓ Understanding enums 3m 17s
 - ✓ Working with structs 2m 38s
- Chapter Quiz 4 questions
- 5. Files and Patterns
 - Reading files 3m 5s
 - Writing files 3m 29s
 - Using match expressions 3m 12s
 - Destructuring assignment 2m 31s
 - Chapter Quiz 4 questions
- Conclusion
- Building for the future with Rust 1m

Rust for JavaScript Developers
Chapter Quiz

Question 2 of 4
Tuples are useful when _____.

returning types
Incorrect

calling functions
Incorrect

passing arguments
This was the correct answer

iterating through arrays
Incorrect

Review this video
Building tuples
3m 2s

[Next question](#)

Contents (X)

Rust for JavaScript Developers
Chapter Quiz

1. Creating Rust Programs >

2. Rust Primitives >

3. Rust Functions >

4. Complex Data Types in Rust ▼

- ✓ Building tuples 3m 2s
- ✓ Creating vectors 2m 31s
- ✓ Understanding enums 3m 17s
- ✓ Working with structs 2m 38s

Chapter Quiz 4 questions

5. Files and Patterns ▼

- Reading files 3m 5s
- Writing files 3m 29s
- Using match expressions 3m 12s
- Destructuring assignment 2m 31s

Chapter Quiz 4 questions

Conclusion ▼

○ Building for the future with Rust 1m

Question 3 of 4

A struct in Rust is most similar to what in JavaScript?

a Node package

a number

an object

Correct

an array

Next question

Contents (X)

Rust for JavaScript Developers
Chapter Quiz

1. Creating Rust Programs >

2. Rust Primitives >

3. Rust Functions >

4. Complex Data Types in Rust ▼

- ✓ Building tuples 3m 2s
- ✓ Creating vectors 2m 31s
- ✓ Understanding enums 3m 17s
- ✓ Working with structs 2m 38s

Chapter Quiz 4 questions

5. Files and Patterns ▼

- Reading files 3m 5s
- Writing files 3m 29s
- Using match expressions 3m 12s
- Destructuring assignment 2m 31s

Chapter Quiz 4 questions

Conclusion ▼

○ Building for the future with Rust 1m

Question 4 of 4

What are enums best for?

setting types for an integer

storing a fixed set of values

Correct

returning values from a function

storing a collection of arrays

Next

Contents (X)

Rust for JavaScript Developers
Chapter Quiz

1. Creating Rust Programs >

2. Rust Primitives >

3. Rust Functions >

4. Complex Data Types in Rust ▾

- ✓ Building tuples 3m 2s
- ✓ Creating vectors 2m 31s
- ✓ Understanding enums 3m 17s
- ✓ Working with structs 2m 38s

✓ Chapter Quiz 4 questions

5. Files and Patterns ▾

- Reading files 3m 5s
- Writing files 3m 29s
- Using match expressions 3m 12s
- Destructuring assignment 2m 31s
- Chapter Quiz 4 questions

Conclusion ▾

○ Building for the future with Rust 1m



You answered 3 of 4 questions correctly.
Keep practicing! Review your answers and retake the quiz.

[Review all answers](#) [Continue watching](#)

Files and Patterns

Reading files

```
# Similar to imports in JavaScript land - we will import the filesystem module from the standard Rust library
use std::fs;

fn main() {
    let text = fs::read_to_string("my_file.txt").expect("Something went wrong reading the file");

    println!("What is in this file:\n{}", text);
}
```

Writing files

```
use std::fs::OpenOptions; // We need OpenOptions so we can open a file
use std::io::Write; // Let's add the ability to write to a file

fn main() {
    let mut file = OpenOptions::new()
        .append(true) // Add to our file unless we want to overwrite the existing contents of the file
        .open("my_file.txt")
        .expect("Something went awry with the file");
    let text = "We're making it happen!";
    file.write_all(text.as_bytes())
        .expect("something went wrong");
}
```

Using match expressions

```
fn main() {
    let destination = "Snow Lake";

    match destination {
        "Long Lake" => println!("We're heading to Long Lake!"),
        "Mammoth Lakes" => println!("We're heading to Mammoth!"),
        "Bowman Lake" => println!("We're heading to Bowman Lake!"),
        _ => println!("We're heading anywhere else"),
    }
}
```

```
}
```

Destructuring assignment

```
fn main() {
    struct Hiker {
        name: String,
        miles_hiked: u64,
    }

    let billy = Hiker {
        name: "Billy".to_string(),
        miles_hiked: 67,
    };

    let Hiker { name, miles_hiked } = billy;

    println!("name {}, miles: {}", name, miles_hiked);
}
```

Chapter Quiz

The screenshot shows a dark-themed user interface for a chapter quiz. On the left is a sidebar with a 'Contents' section containing links to chapters 1 through 5, file operations, and the current 'Chapter Quiz'. The main area is titled 'Rust for JavaScript Developers Chapter Quiz'. A question box is open, showing 'Question 1 of 4' and the question 'How do you convert a value to a string?'. Three options are listed: 'You can't.', 'Use the .string() method.', and 'Use the .str() method.' The third option is checked with a green checkmark and labeled 'Correct'. A 'Next question' button is visible at the bottom right of the question box.

Rust for JavaScript Developers
Chapter Quiz

Question 1 of 4

How do you convert a value to a string?

You can't.

Use the `.string()` method.

Use the `.to_string()` method.
Correct

Next question

Contents (X)

Rust for JavaScript Developers
Chapter Quiz

Question 2 of 4
How are arrow functions used in Rust?

for fun

for inline function returns
Correct

for profit

for type setting

[Next question](#)

Introduction >

1. Creating Rust Programs >

2. Rust Primitives >

3. Rust Functions >

4. Complex Data Types in Rust >

5. Files and Patterns ▼

- ✓ Reading files 3m 5s
- ✓ Writing files 3m 29s
- ✓ Using match expressions 3m 12s
- ✓ Destructuring assignment 2m 31s

● Chapter Quiz 4 questions

Conclusion ▼

○ Building for the future with Rust 1m

Contents (X)

Rust for JavaScript Developers
Chapter Quiz

Question 3 of 4
When is the .expect() method called in Rust?

when there is an error
Correct

when a file is deleted

when you're out of memory

when a file is opened

[Next question](#)

Introduction >

1. Creating Rust Programs >

2. Rust Primitives >

3. Rust Functions >

4. Complex Data Types in Rust >

5. Files and Patterns ▼

- ✓ Reading files 3m 5s
- ✓ Writing files 3m 29s
- ✓ Using match expressions 3m 12s
- ✓ Destructuring assignment 2m 31s

● Chapter Quiz 4 questions

Conclusion ▼

○ Building for the future with Rust 1m

Contents (X)

Rust for JavaScript Developers
Chapter Quiz

Question 4 of 4
If you wanted to read files from your computer with Rust, you'd use which library?

std::file

file

std::fs
Correct

fs::std

Next

Introduction >

1. Creating Rust Programs >

2. Rust Primitives >

3. Rust Functions >

4. Complex Data Types in Rust >

5. Files and Patterns ▼

- ✓ Reading files 3m 5s
- ✓ Writing files 3m 29s
- ✓ Using match expressions 3m 12s
- ✓ Destructuring assignment 2m 31s

✓ Chapter Quiz 4 questions

Conclusion ▼

Building for the future with Rust 1m

Contents (X)

Rust for JavaScript Developers
Chapter Quiz

You answered 4 of 4 questions correctly.
You successfully completed all questions in this quiz.

Review all answers Continue watching



Introduction >

1. Creating Rust Programs >

2. Rust Primitives >

3. Rust Functions >

4. Complex Data Types in Rust >

5. Files and Patterns ▼

- ✓ Reading files 3m 5s
- ✓ Writing files 3m 29s
- ✓ Using match expressions 3m 12s
- ✓ Destructuring assignment 2m 31s

✓ Chapter Quiz 4 questions

Conclusion ▼

Building for the future with Rust 1m

Conclusion

Building for the future with Rust

Contents

- Introduction
- 1. Creating Rust Programs
- 2. Rust Primitives
- 3. Rust Functions
- 4. Complex Data Types in Rust
- 5. Files and Patterns
 - ✓ Reading files 3m 5s
 - ✓ Writing files 3m 29s
 - ✓ Using match expressions 3m 12s
 - ✓ Destructuring assignment 2m 31s
 - ✓ Chapter Quiz 4 questions
- Conclusion
- Building for the future with Rust 1m

Next Steps

- Rust Essential Training
- Advanced Rust: Managing Projects
- Rust Code Challenges

LinkedIn Learning

Contents

- Introduction
- 1. Creating Rust Programs
- 2. Rust Primitives
- 3. Rust Functions
- 4. Complex Data Types in Rust
- 5. Files and Patterns
 - ✓ Reading files 3m 5s
 - ✓ Writing files 3m 29s
 - ✓ Using match expressions 3m 12s
 - ✓ Destructuring assignment 2m 31s
 - ✓ Chapter Quiz 4 questions
- Conclusion
- Building for the future with Rust 1m

RUN ⚙ DEBUG ⚙ STABLE ⚙ ...

SHARE TOOLS ⚙ CONFIG ⚙

```
fn main() {
    println!("thanks for taking the class!");
}
```

Execution Standard Error

```
Compiling playground v0.0.1 (/playground)
Finished dev [unoptimized + debuginfo] target(s) in 1.25s
Running `target/debug/playground`
```

Standard Output

```
thanks for taking the class!
```

LinkedIn Learning