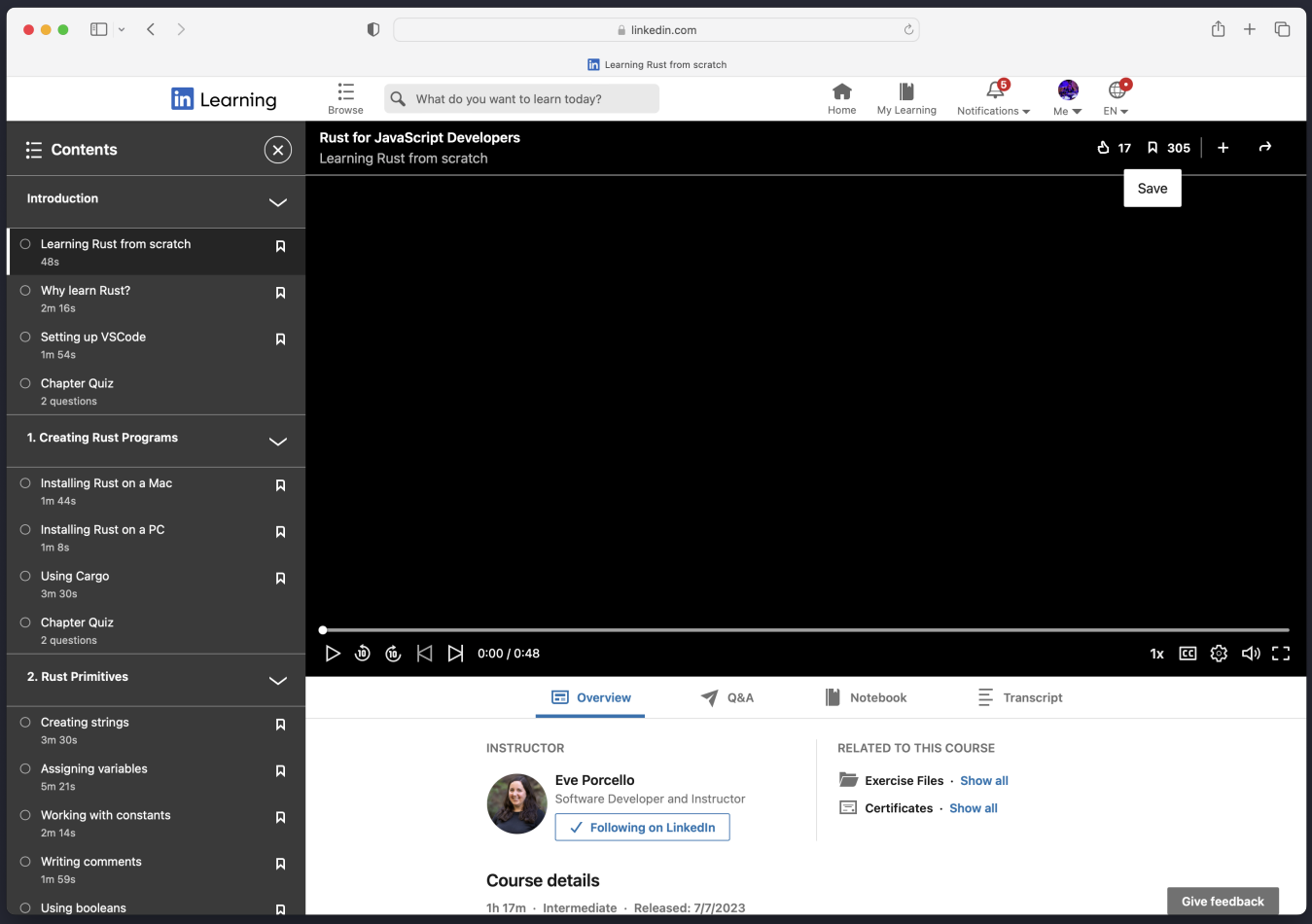
2023.08.26 LinkedIn Course – Eve Porcello – Rust for JavaScript Developers (~1 hr 17 mins)

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*



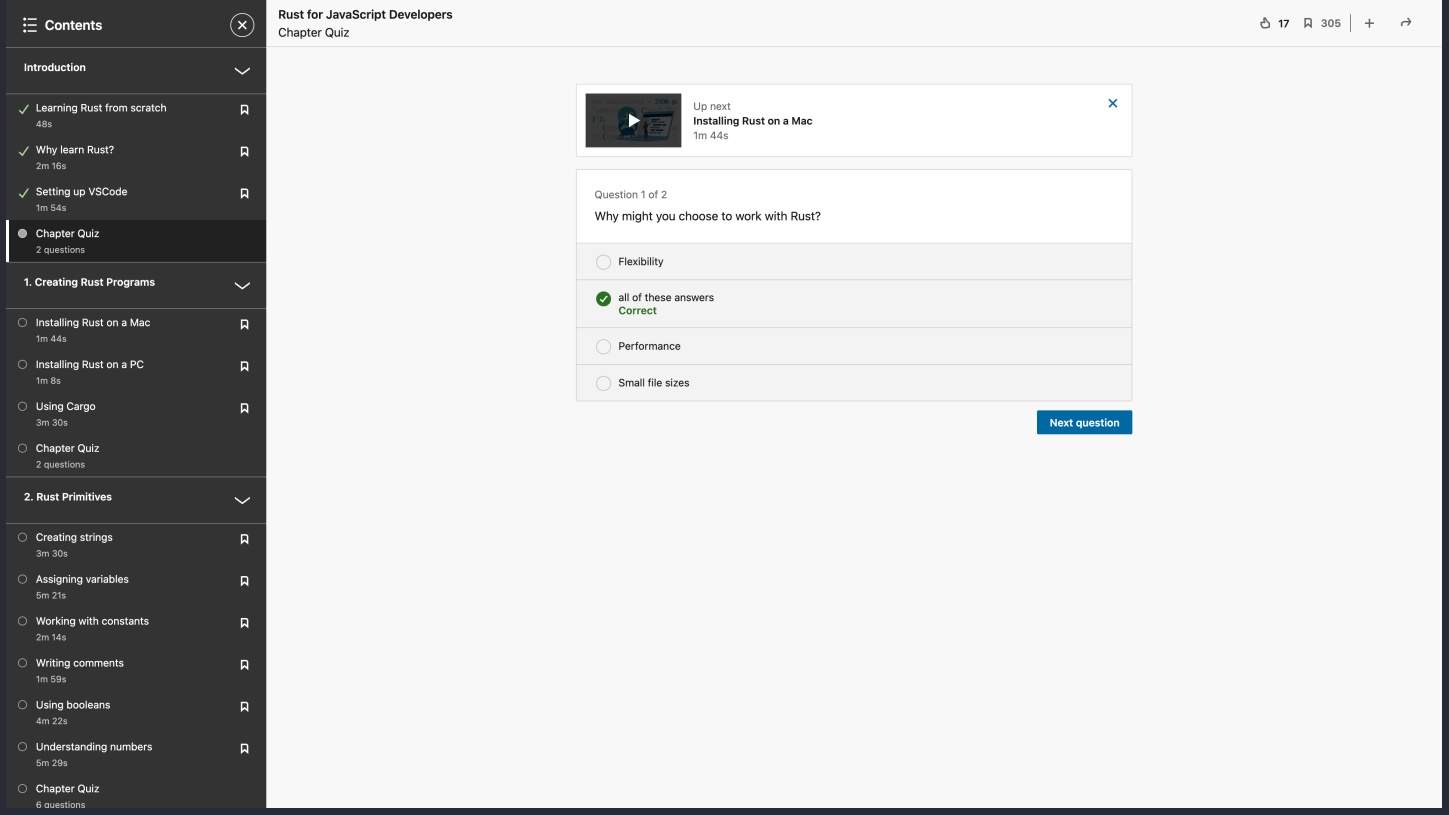
Introduction

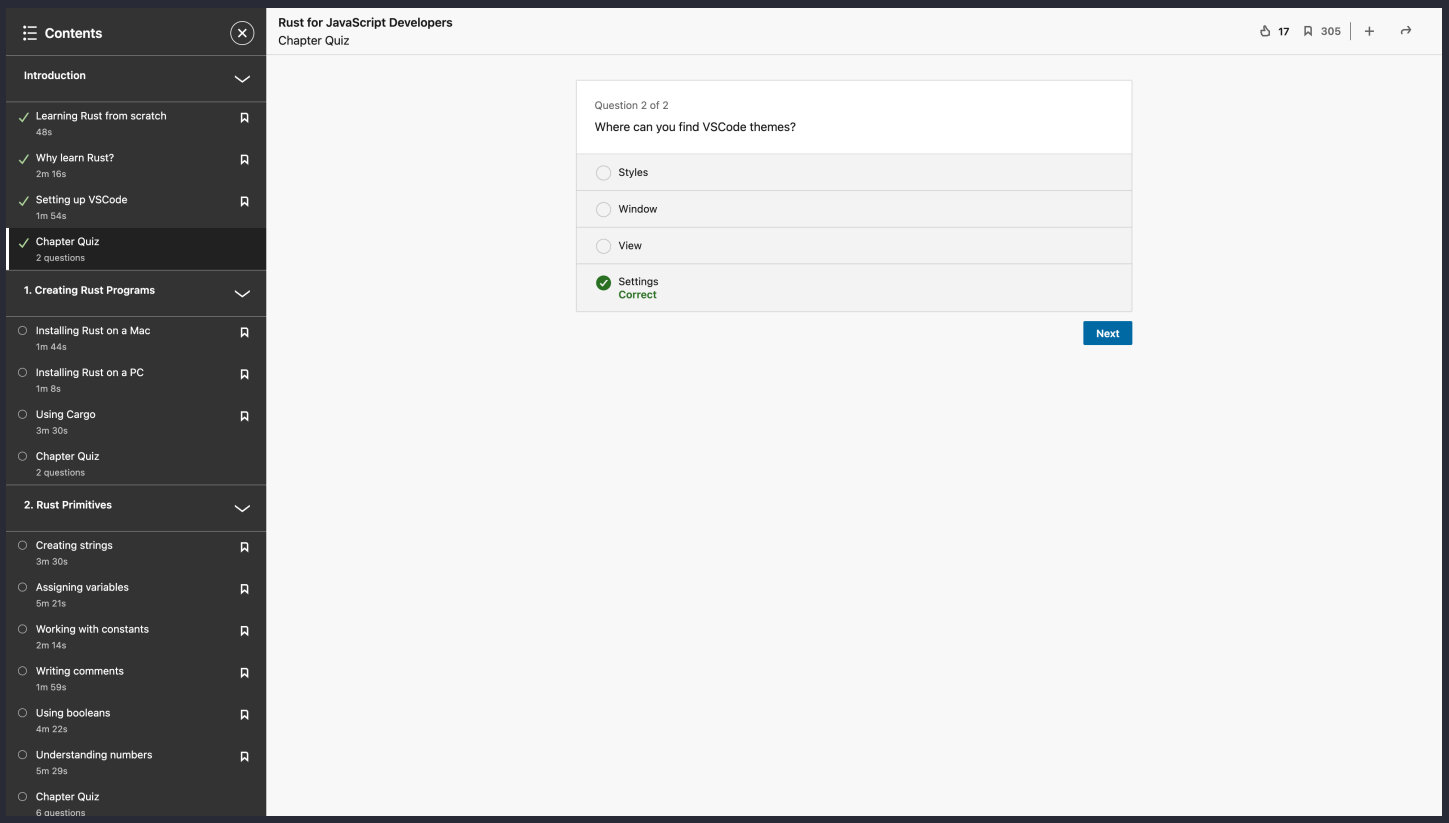
Learning Rust from scratch

Why learn Rust?

Setting up VSCode

Chapter Quiz





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)
```

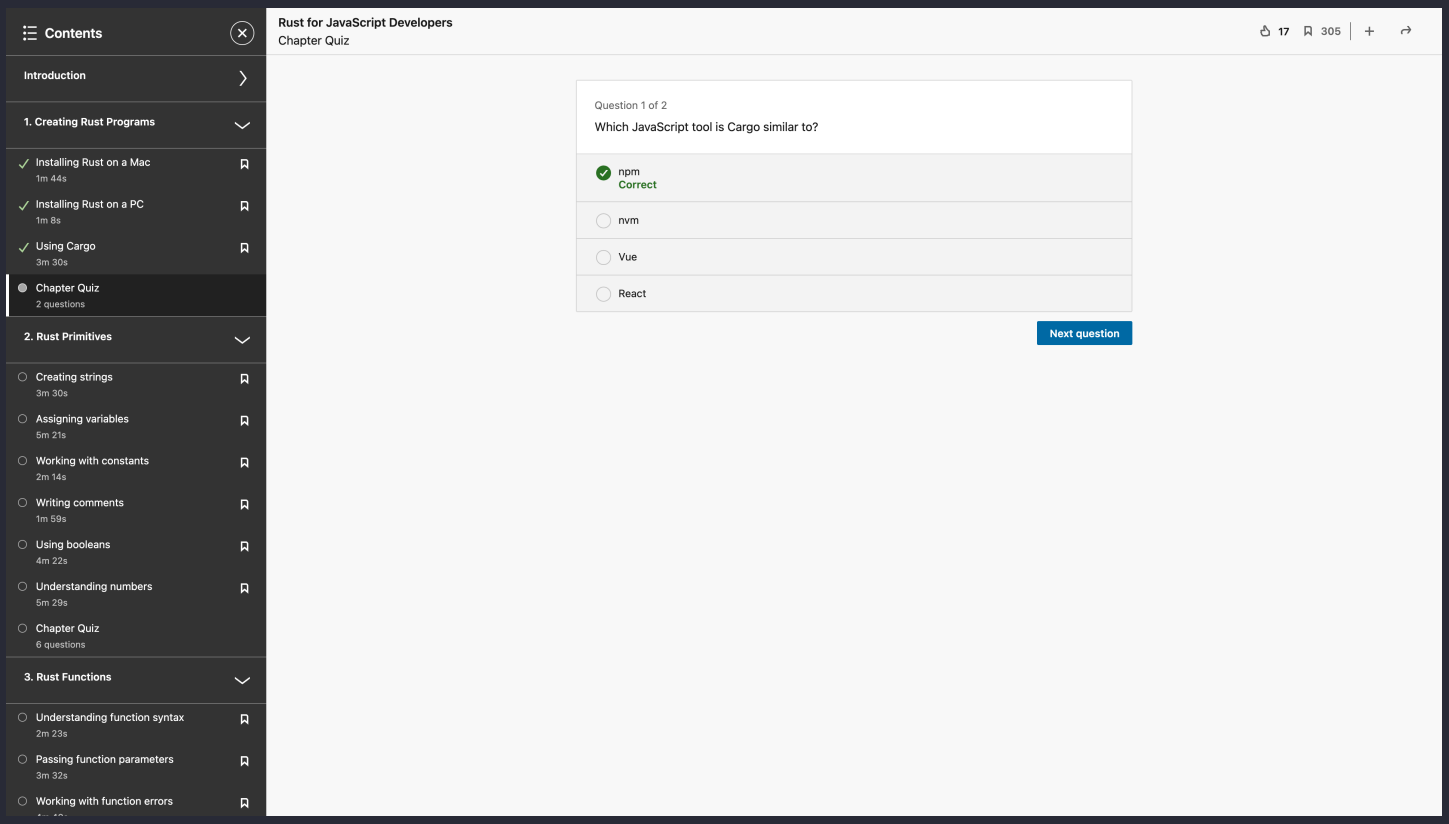
Installing Rust on a PC

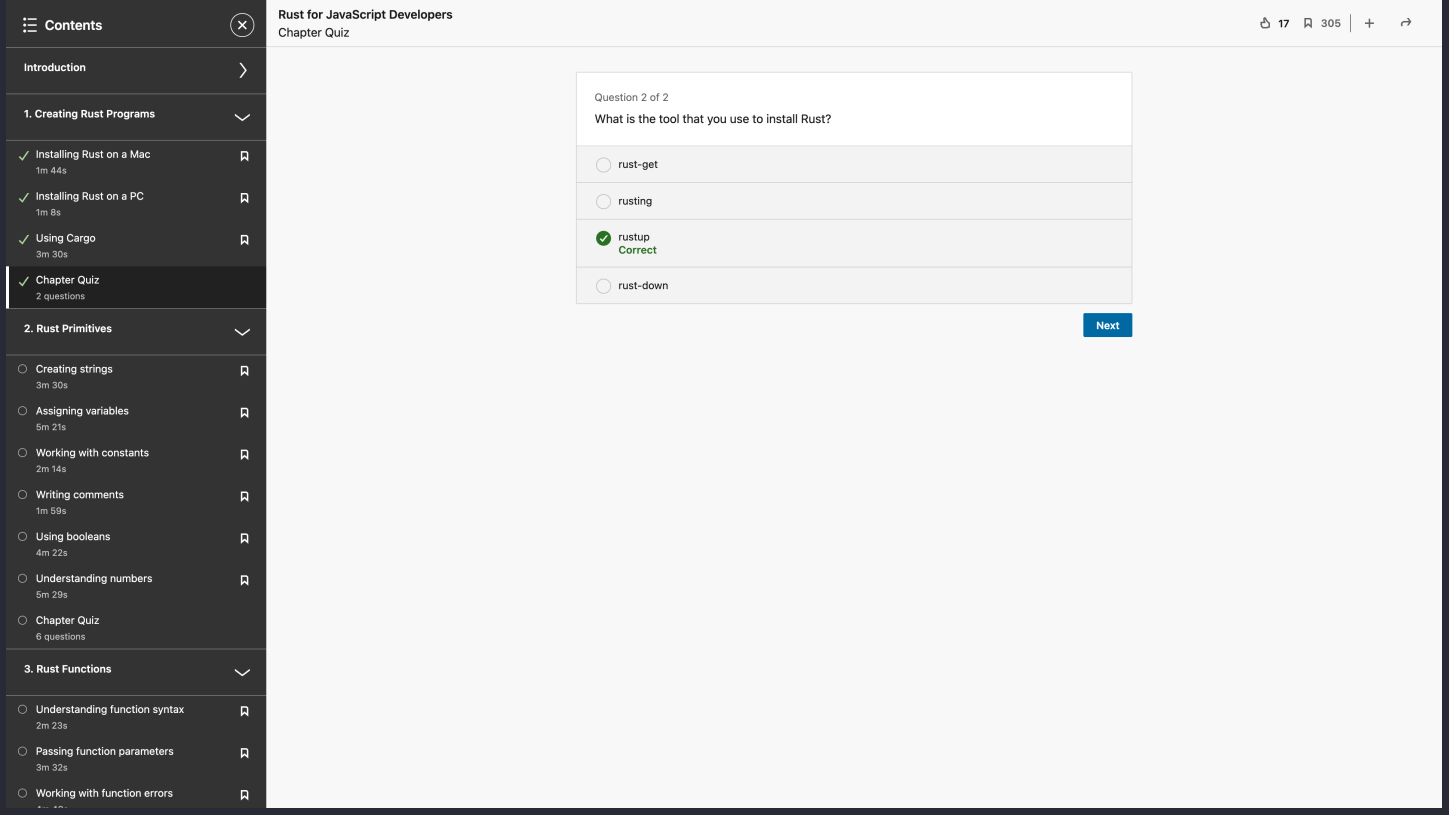
Using Cargo

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





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

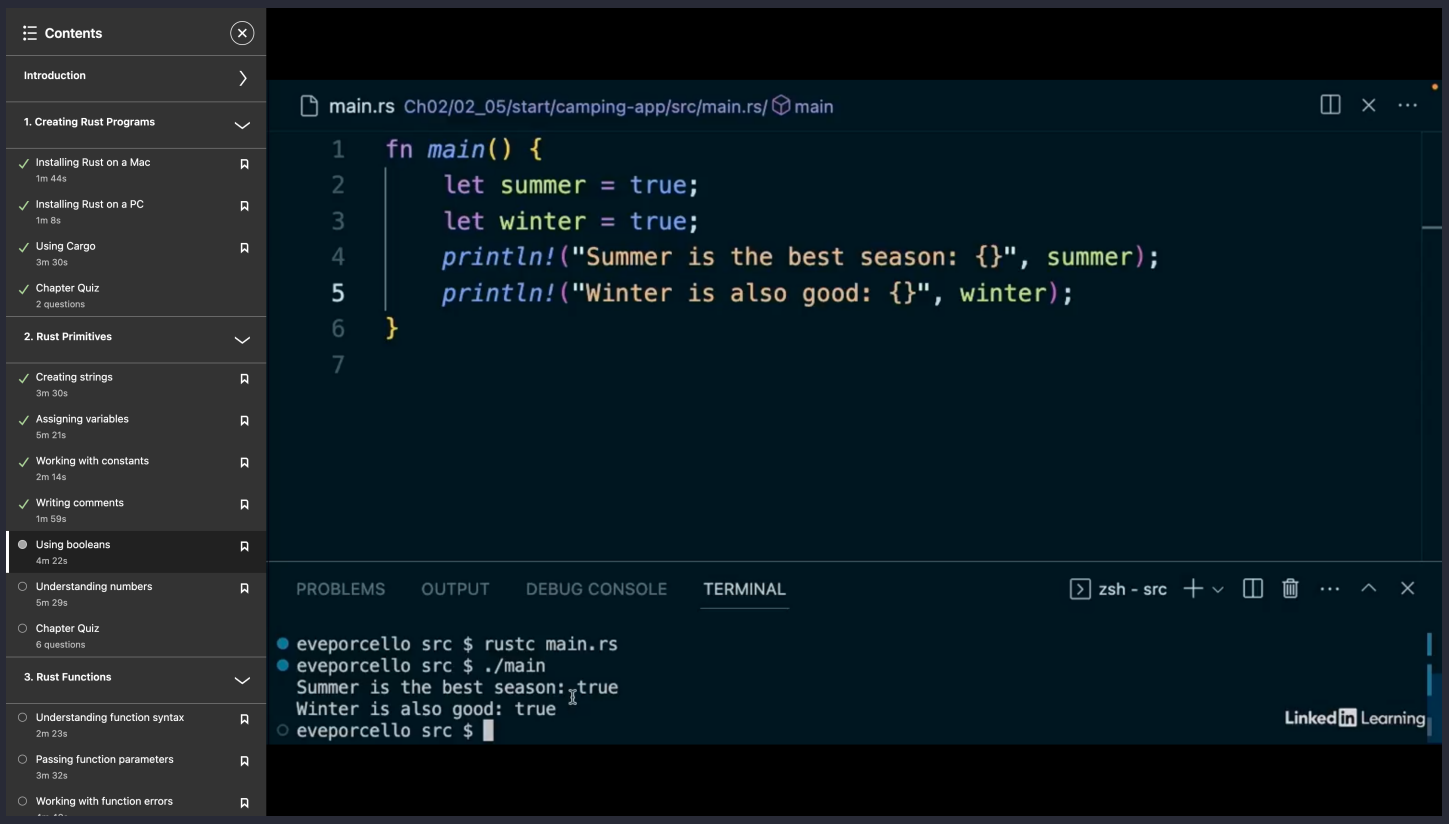
We can use `cargo doc` to generate documentation for our application based on our comments.

```
% 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` directory of our Rust application – e.g., `Exercise Files/Ch02/02_04/finished/camping-app/target/doc/camping_app/index.html`.

Using booleans

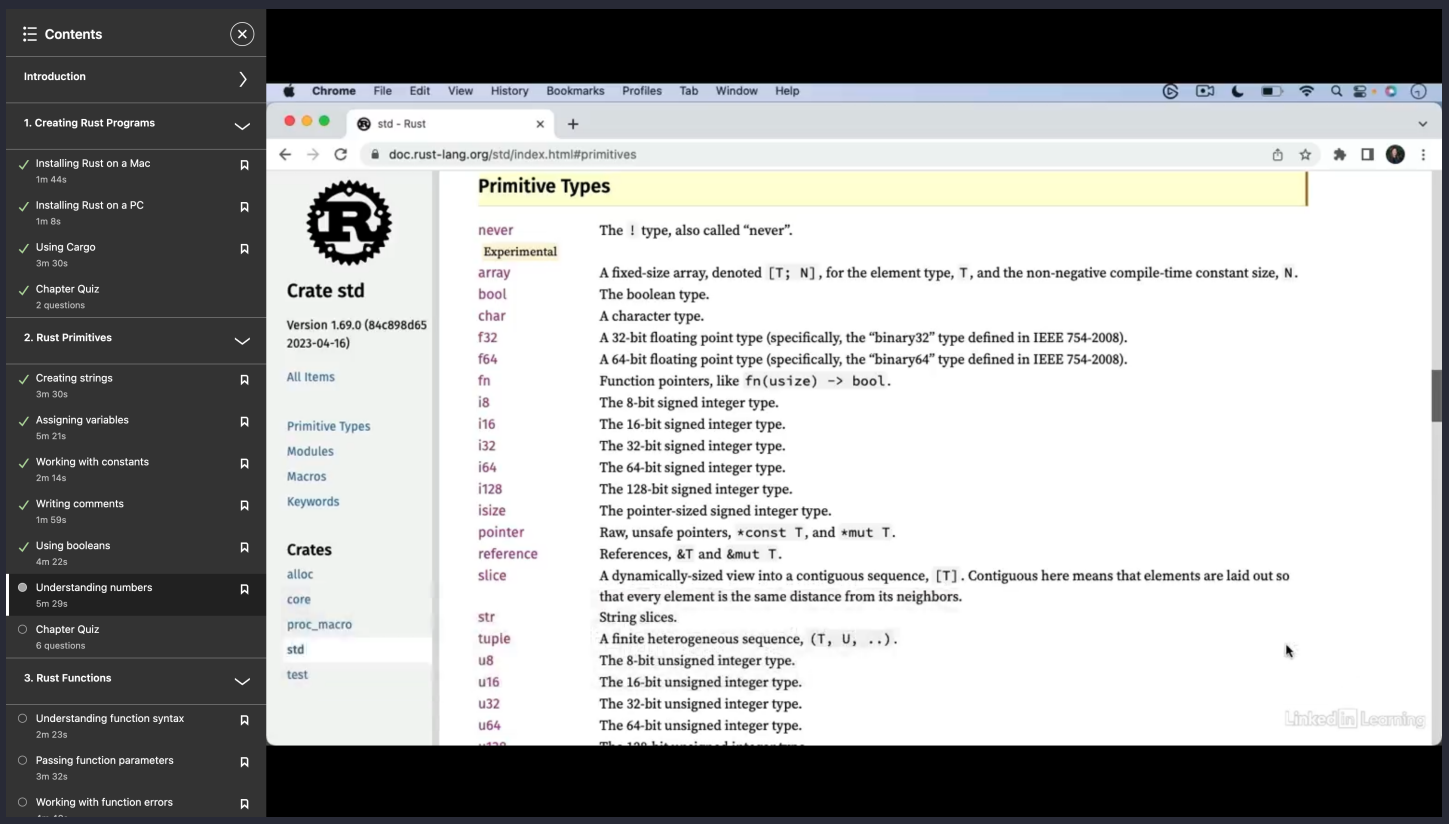
Notice that we didn't add a type annotation here. Rust will infer the variable as a boolean if it is assigned `true` or `false`.




```
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




```
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

Question 1 of 6

What is the type annotation for a boolean?

- b
- true/false
- bool
  Correct
- boolean

Next question

Contents

Rust for JavaScript Developers
Chapter Quiz

17 305

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

Question 2 of 6

How do you create documentation comments in Rust?

- ???
- /?
- ///
  Correct
- /*

Next question

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

Rust for JavaScript Developers
Chapter Quiz

17 305

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

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

Question 5 of 6

Which character do you use to denote a string?

- Backticks
- Double quotes
  Correct
- Exclamation marks
- Single quotes

Next question

Contents

Rust for JavaScript Developers
Chapter Quiz

17 305

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

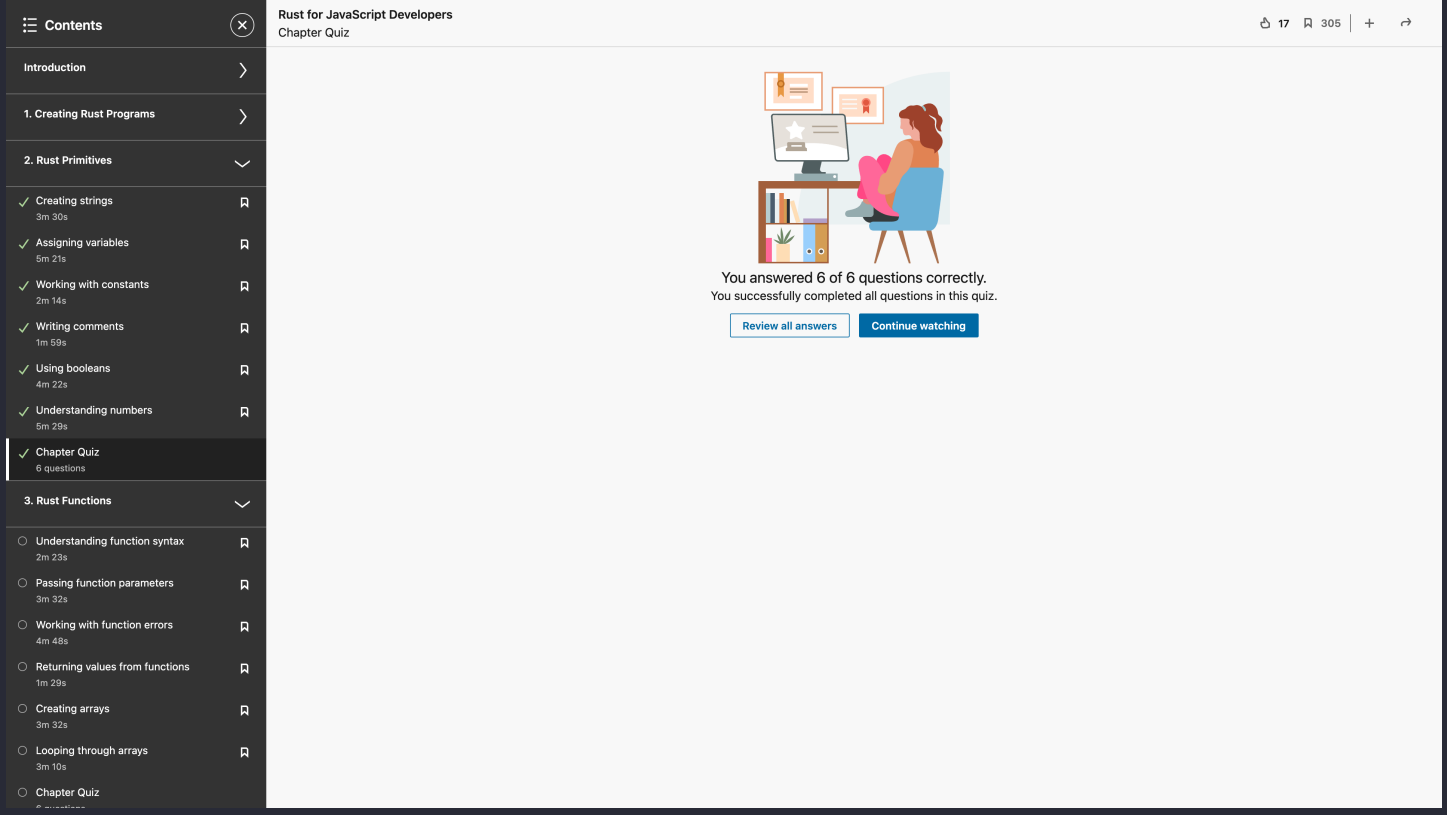
- Understanding function syntax 2m 23s
- Passing function parameters 3m 32s
- Working with function errors 4m 48s
- Returning values from functions 1m 29s
- Creating arrays 3m 32s
- Looping through arrays 3m 10s
- Chapter Quiz

Question 6 of 6

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

- unsupported
- underlined
- unbelievable
- unsigned
  Correct

Next



Rust Functions

Understanding function syntax

```rust
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

```rust
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

```rust
# 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)
    }
}
```

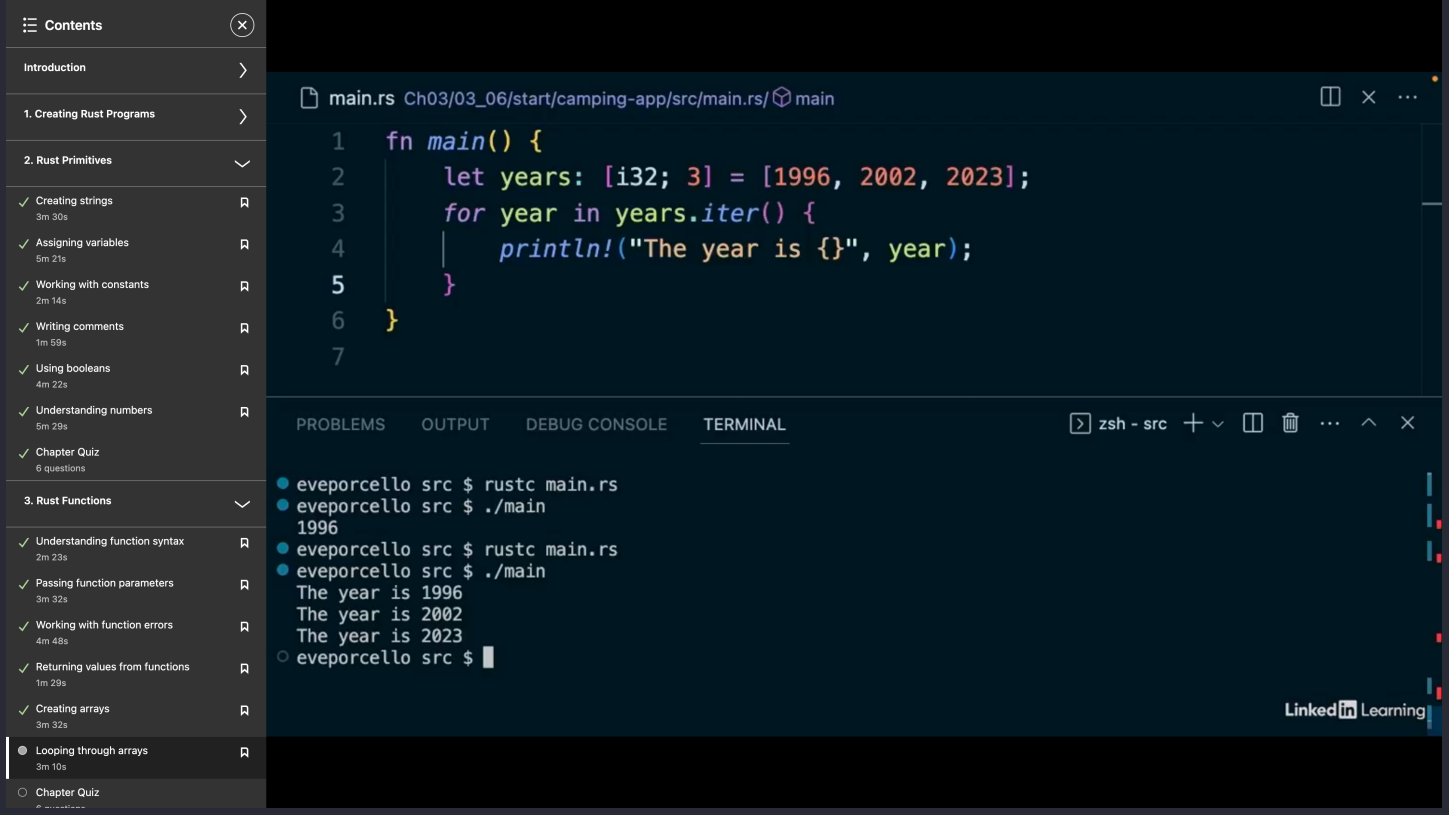
```
fn main() {
    let result = calculate_distance(0, 10);

        # 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.len());
}
```

Looping through arrays



BONUS: Create a countdown using a while loop

```rust
fn main() {
    let mut new_year_countdown = 10;

    while new_year_countdown > 0 {
        println!("{new_year_countdown}");
        new_year_countdown -= 1;
    }
}
```

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 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

17 305

Question 2 of 6

Arrays can hold:

- the same type **Correct**
- only numbers
- different types
- only strings

Next question

Question 3 of 6

What is a function that returns an error in Rust?

- `Error()`
- `Exception()`
- ✓ `Err()` Correct
- `E()`

Next question

Contents

Rust for JavaScript Developers
Chapter Quiz

17 305

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

Question 4 of 6

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

- cry for help
- function
- return
- ✓ macro Correct

Next question

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

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

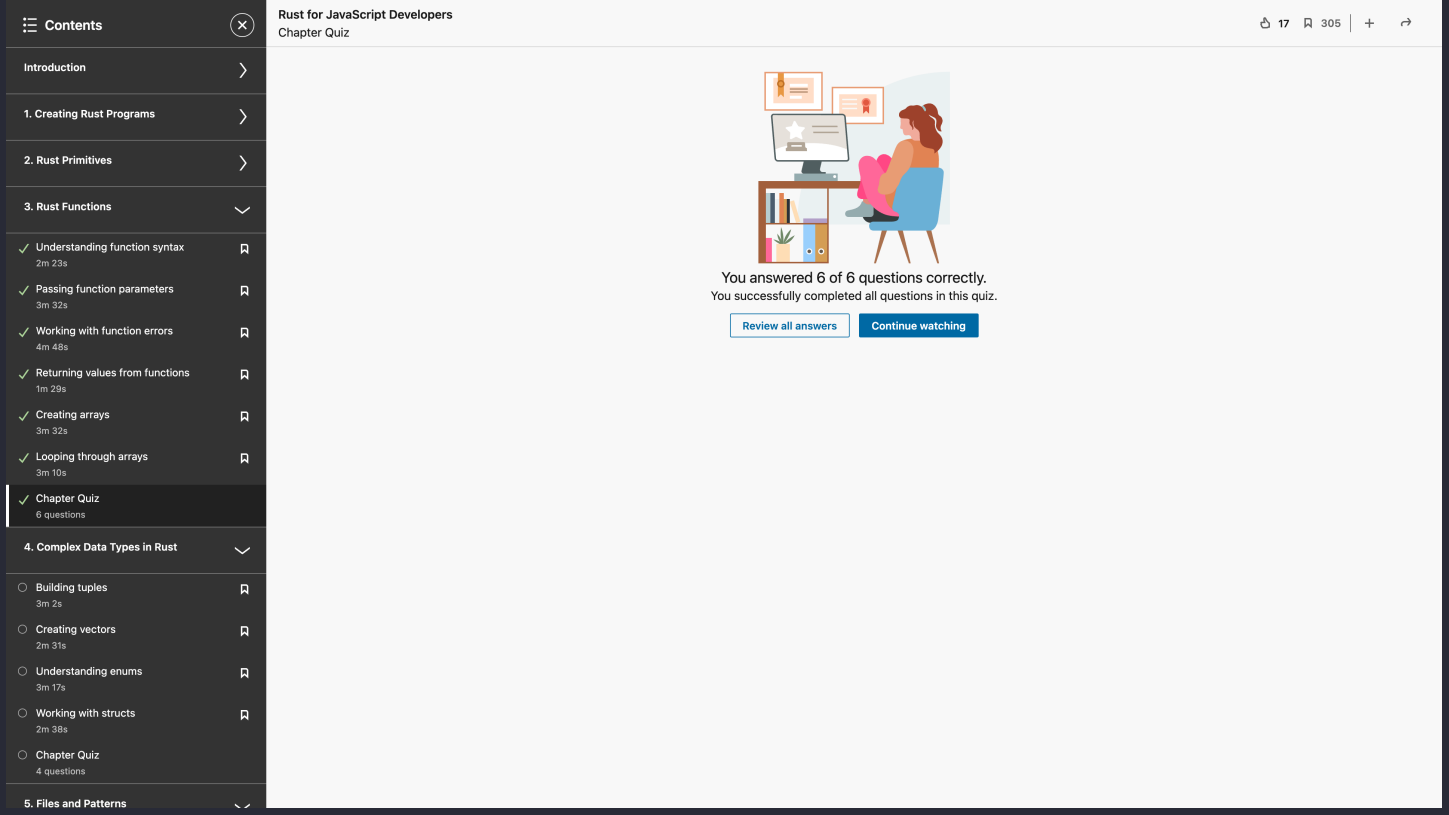
17 305

Question 6 of 6

How do you declare a function in Rust?

- fun keyword
- function keyword
- function! keyword
- fn keyword
  Correct

Next



Complex Data Types in Rust

Building tuples


```
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.

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);
}
```

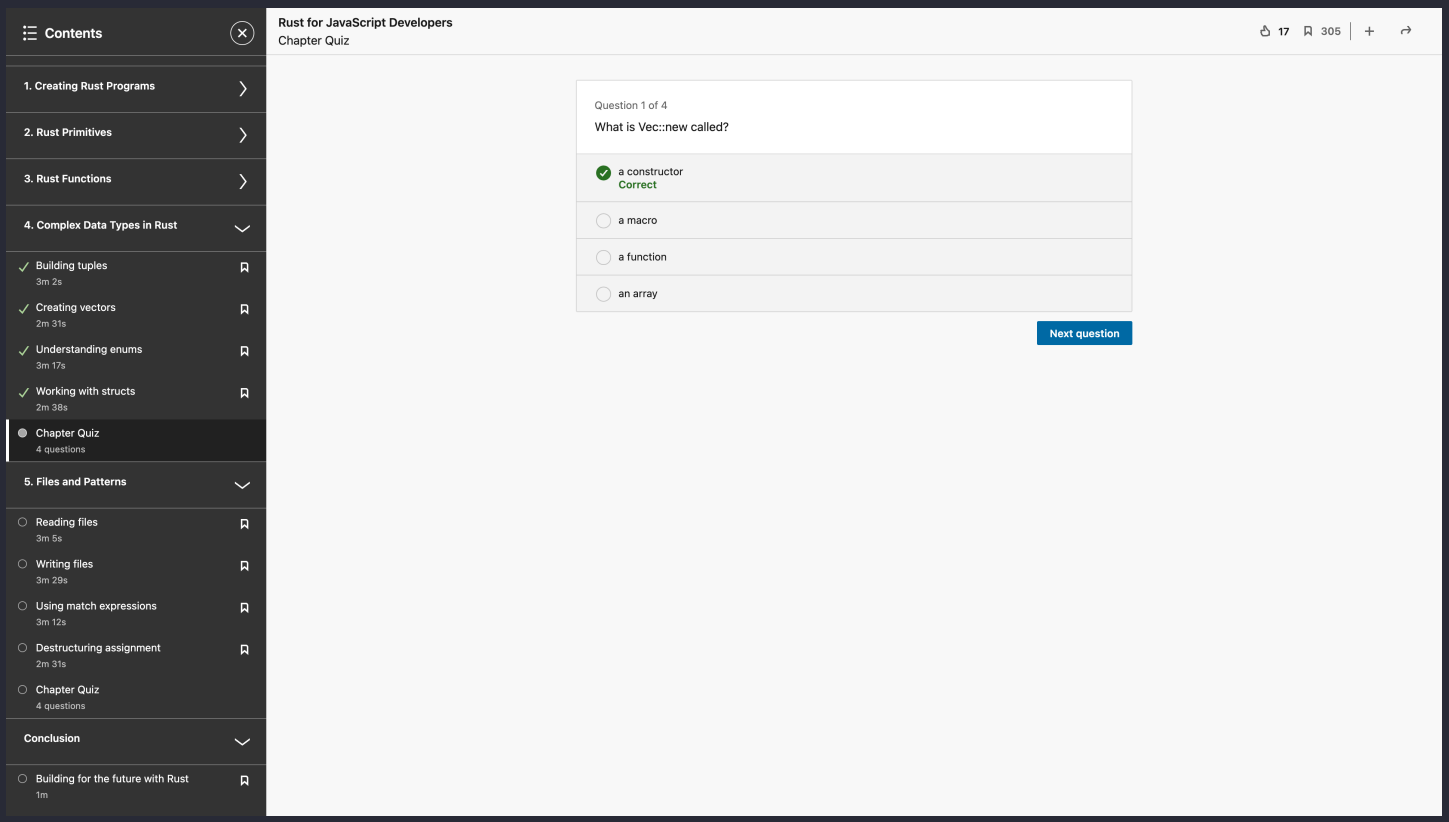
Understanding enums

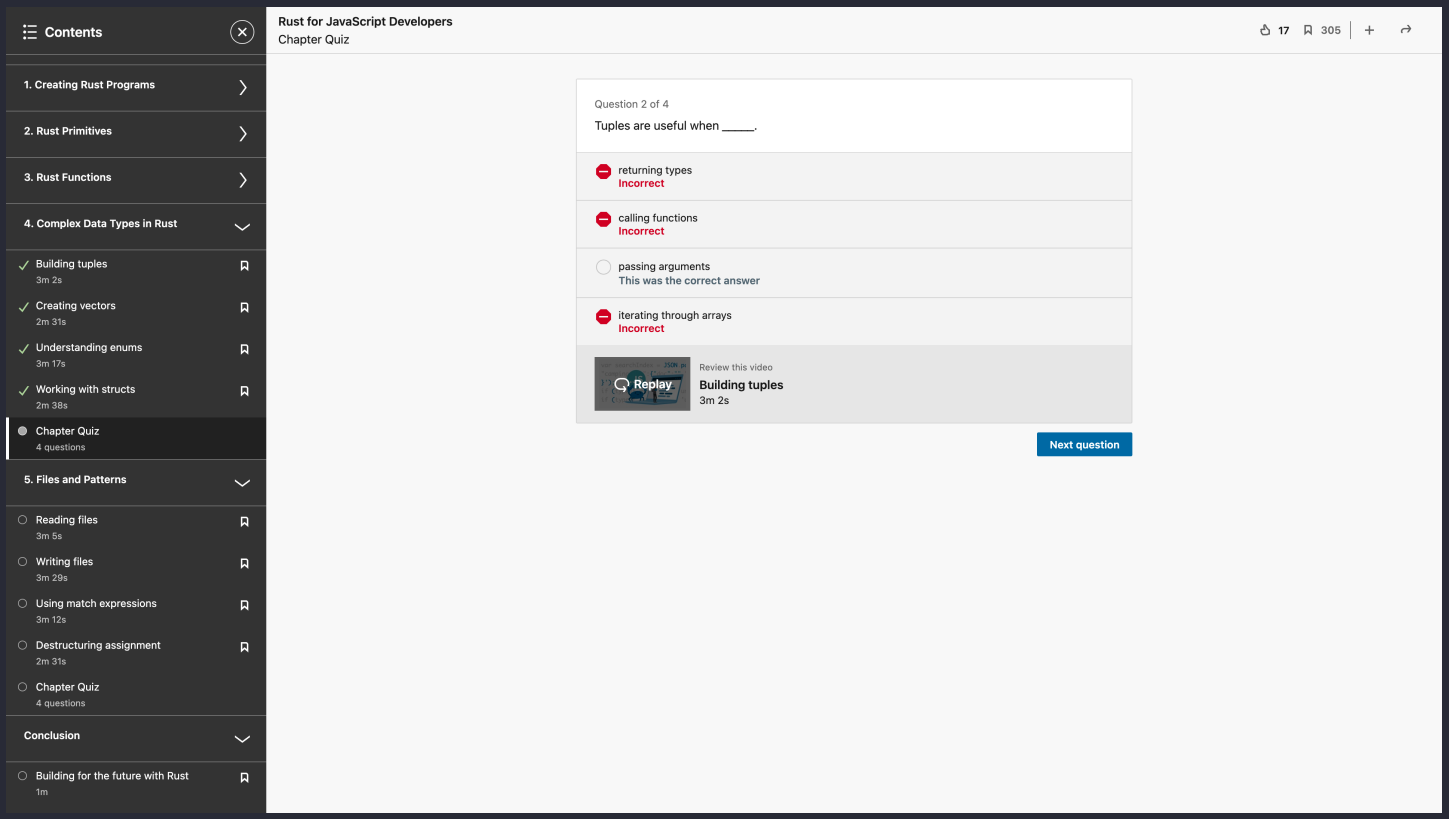
```
#[derive(Debug)]
enum Steepness {
    Easy,
    Moderate,
    Difficult,
}

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





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

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
17 305

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

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

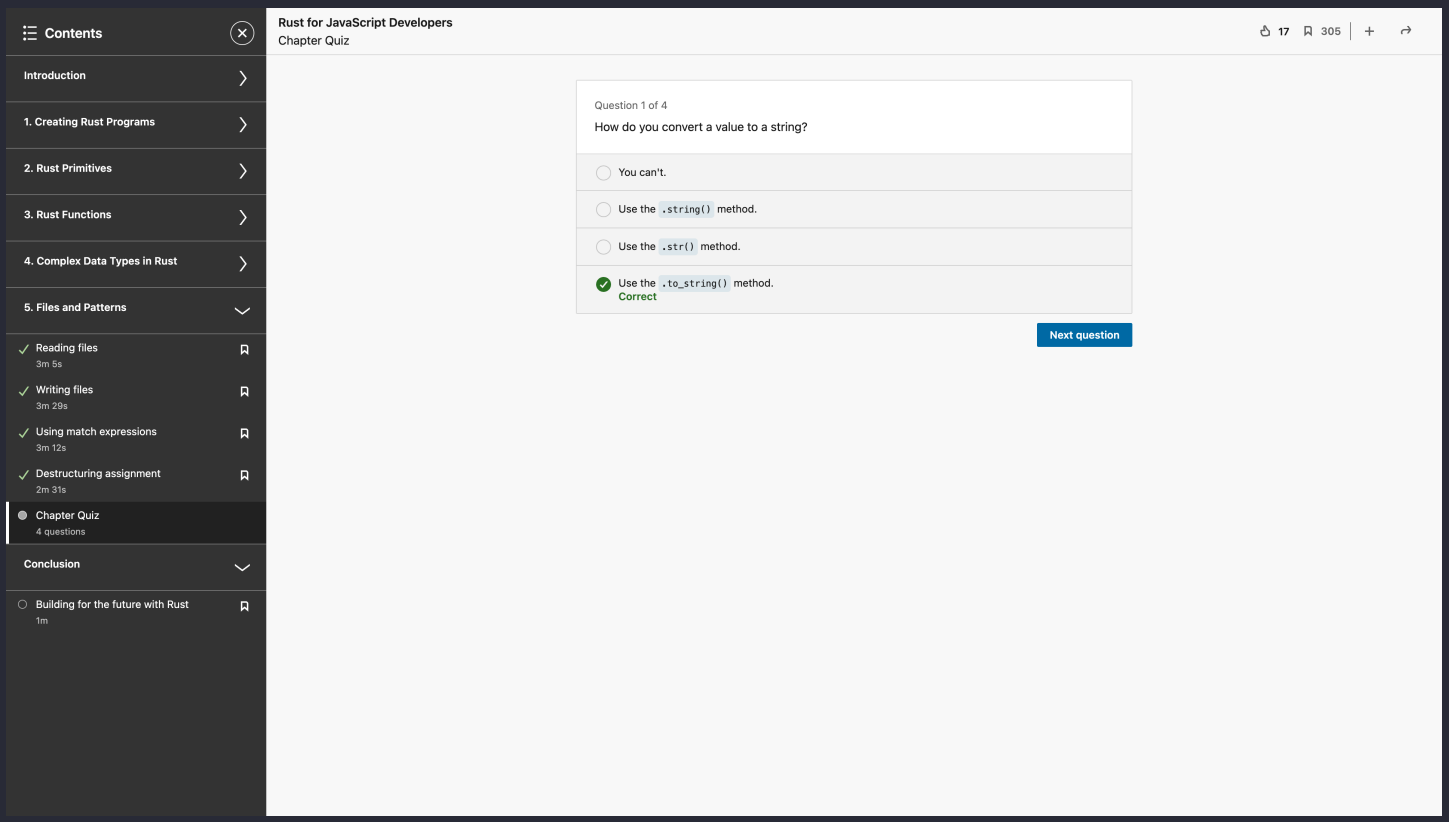
```
fn main() {
    let destination = "Snow Lake";

    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



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

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

Rust for JavaScript Developers
Chapter Quiz

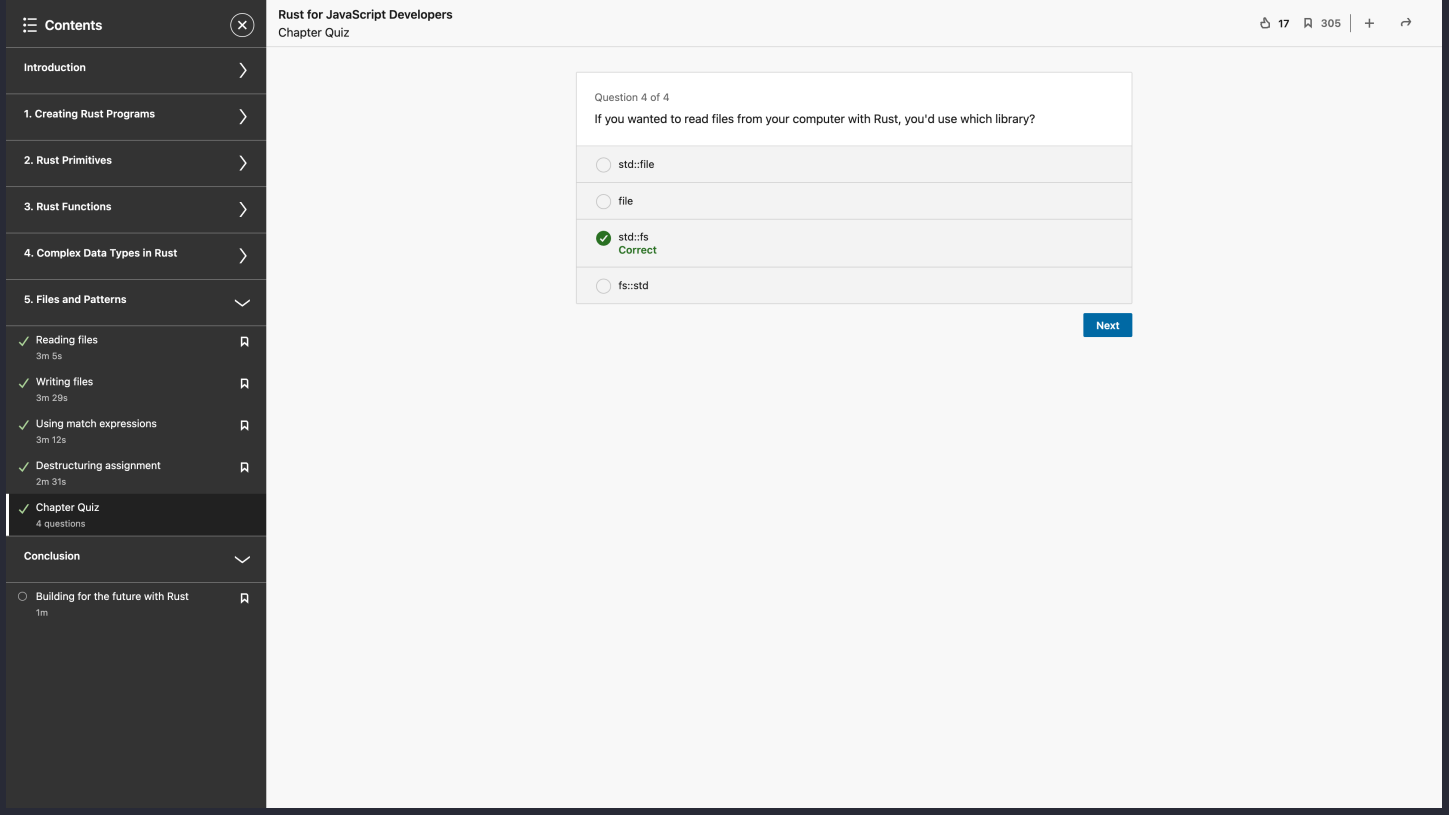
17 305

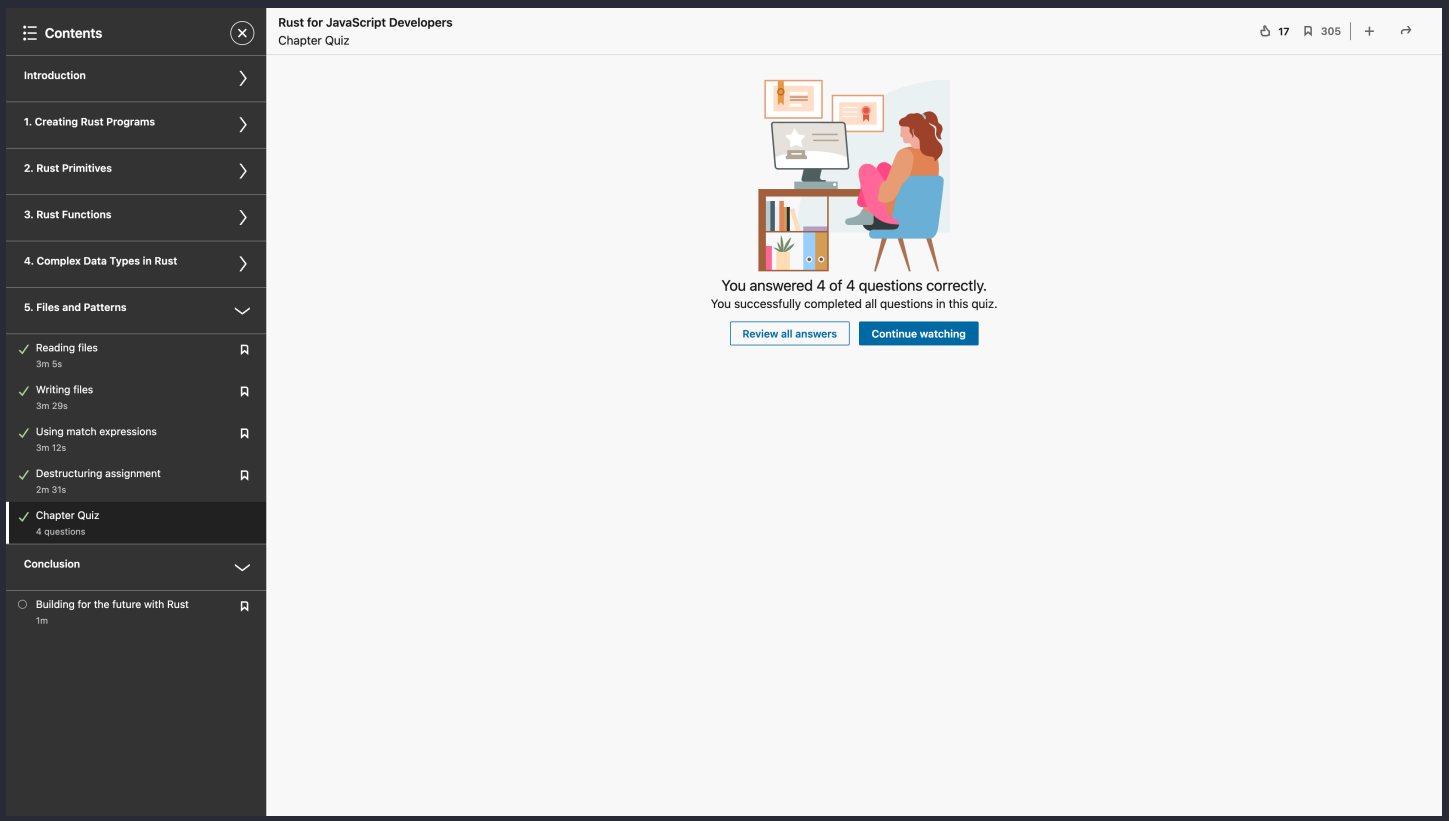
Question 3 of 4

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





Conclusion

Building for the future with Rust

Next Steps

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

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

```
1 fn main() {
2     println!("thanks for taking the class!");
3 }
```

Execution Close

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!
```