Introduction to Strings

This lesson will discuss strings, the two different types, and how to create them in Rust.

We'll cover the following

What are Strings?

Strings are a sequence of Unicode characters. In Rust, a String is not null-terminated unlike strings in other programming languages. They can contain null characters.

> **Note**: Have a look at the unicode characters

```
"Rust Programming Language"
```

Types of Strings

Strings are of two types: &str and String

String Literal (`&str`)

A String literal has the following properties:

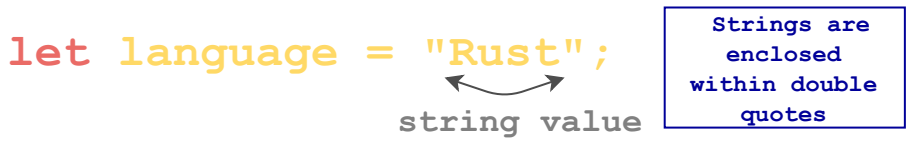
- Primitive type

- Immutable
- Fixed-length string stored somewhere in memory
- Value of string is known at compile time

> **Note**: A String literal is also known as a String slice.

Create a String Literal

The following illustration shows how to create a primitive string:




```
fn main() {
  //define a primitive String variable
  let language:&str = "Rust";
  //print the String literal
  println!("String literal: {}", language);
  //print the length of the String literal
  println!("Length of the string literal: {}", language.len());
}
```

String Object (`String`)

A String object has the following properties:

- A string is encoded as a UTF-8 sequence
- Heap-allocated data structure
- The size of this string can be modified
- Not null-terminated
- Encode string values that are given at the run time

Create a String Object

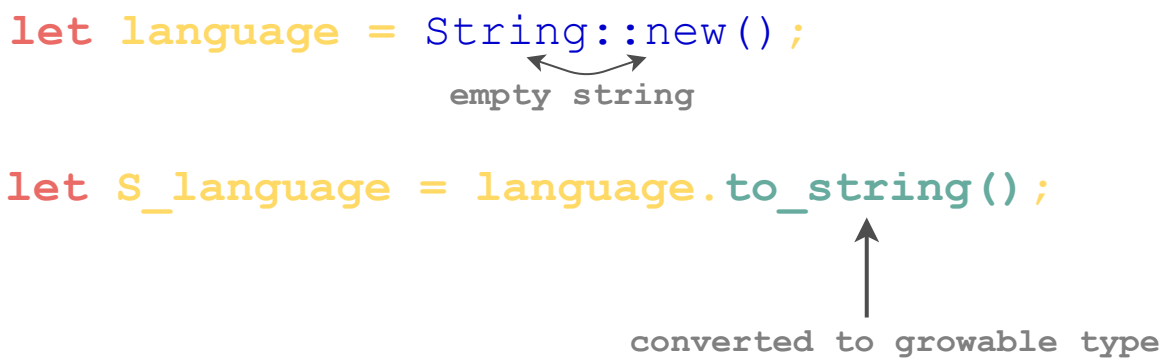
There are many different ways to create and manipulate String objects. We will discuss two here.

Creating an Empty String Object

This method converts the empty String or a String literal to a String object using

the `.tostring` method.

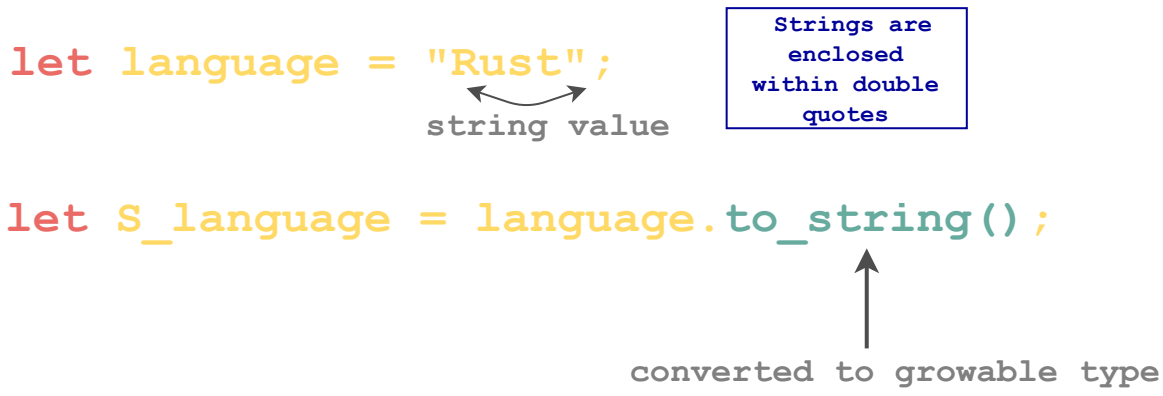
The following illustration creates an empty String and then converts into the string object using the `.to_string()` method.



Creating an Initialized String Object

- This method creates a string with some default value passed as an argument to the `from()` method.

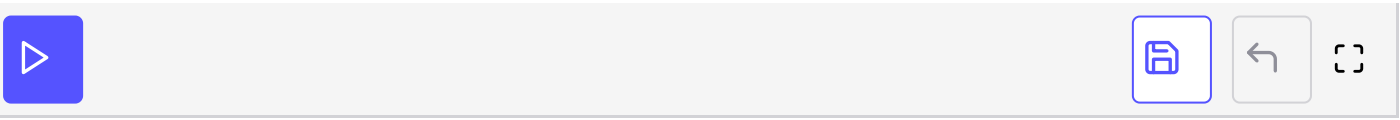
- The following illustration creates a String literal and then converts it into the String object.




```
fn main() {
  // create an empty String
  let course1 = String::new();
  // convert String literal to String object using .to_string
  let s_course1 = course1.to_string();
  // print the String object
  println!("This is an empty string {}.", s_course1);
  // print the length of an empty String  object
  println!("This is a length of my empty string {}.", s_course1.len());
```

```
  // create a String literal
  let course2 = "Rust Programming";
  // convert String literal to string object using .to_string
  let s_course2 = course2.to_string();
  // print the String object
  println!("This is a string literal : {}.", s_course2);
  // print the length of a String object
  println!("This is a length of my string literal {}.", s_course2.len());

  // define a String object using from method
  let course3 = String::from("Rust Language");
  // print the String object
  println!("This is a string object : {}.", course3);
  // print the length of an string object
  println!("This is the length of my string object {}.", course3.len());
}
```



> **Note:** `len()` is a built-in function used to find the length of a String literal and String object.

Now that we have learned about strings and their types, let's learn about the core methods of String objects in the next lesson.