Introduction to Structs

This lesson gets you acquainted with the basics of Structs.

We'll cover the following

What Are Structs?

Structs consist of related items that potentially have different data types.

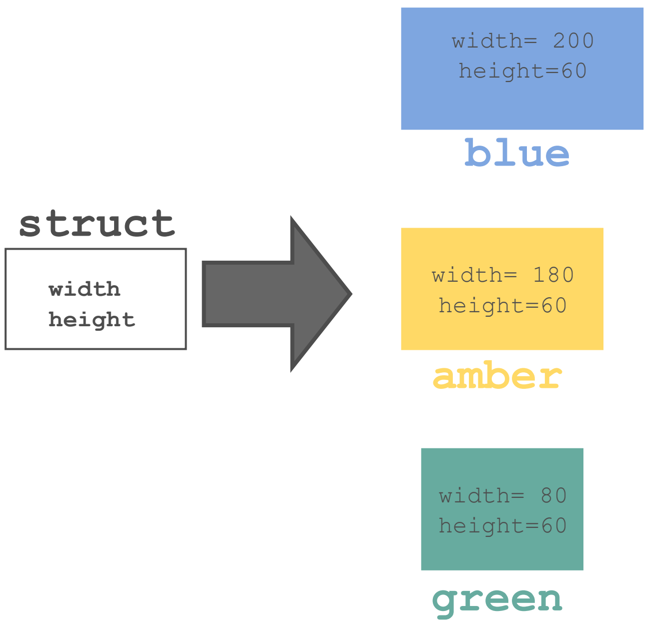
> Structs are similar to tuples in this regard. However, unlike tuples, you must define the data type of the item within the struct.

Structs help to create custom data types.

Let's consider a real life example. You know that a rectangle has two measurements, width and height. Suppose you have several rectangles which you can name. Once you have declared a rectangle items within the struct, you can initialize the values according to the type of rectangle.

Suppose the dimensions of a rectangle

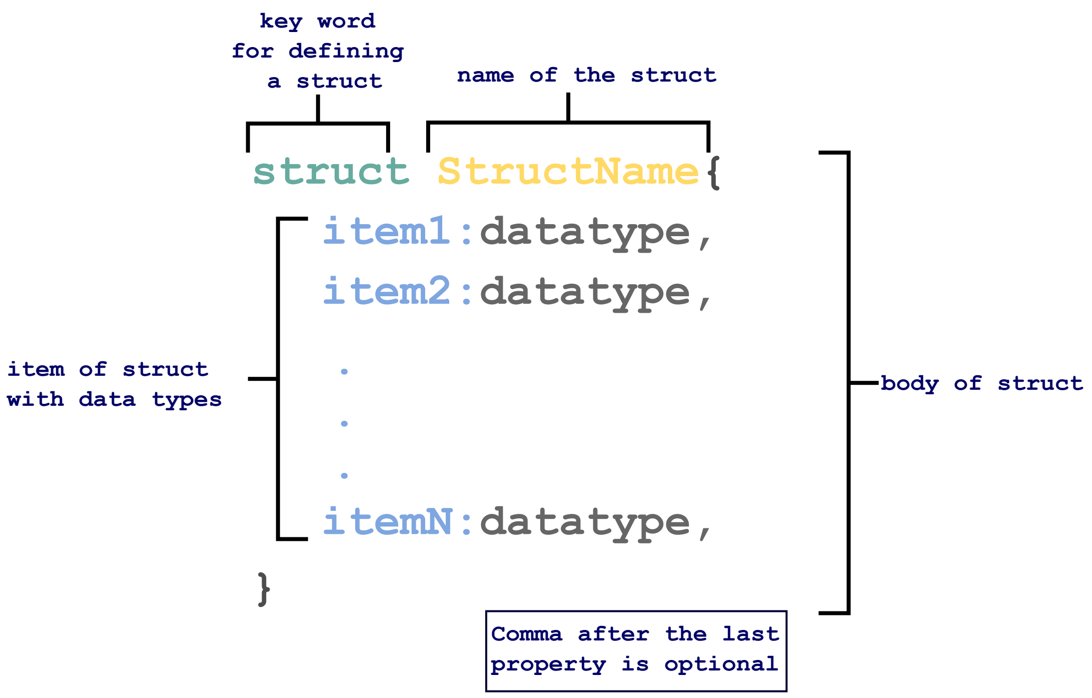
may vary according to the color of the rectangle.



Declare a Struct

Structs are declared using a `struct` keyword followed by the name of the struct and then the body of the struct enclosed within curly braces. Within the body, the items of the struct are defined as a **key: value pair** where keys are the items of the struct and value is the data type of each item.

> **Note:** The struct construct can be declared anywhere, above or below the function that initializes it.



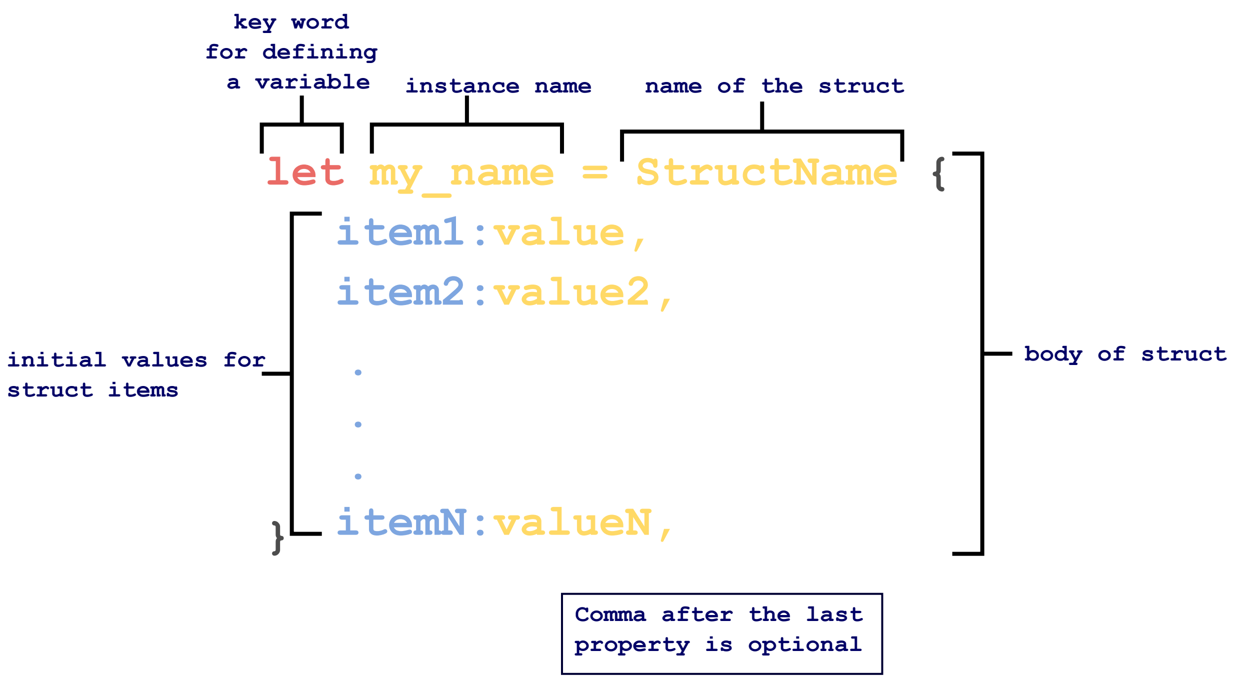
```
struct StructName{
    item1:datatype,
    item2:datatype,
     .
     .
     .
    itemN:datatype,
}
```

Defining a struct

> **Naming Convention:**
>
> The name of the struct should be in **PascalCase**, meaning, the first letter of each word in a compound word is capitalized.
>
> If this case is not followed, a warning, ⚠, is generated by the compiler.

Initialize a Struct

Declaring a struct only defines a blueprint for a custom data type, but no actual object/artifact in memory is created. That is done when a variable of that type is instantiated.


key word for defining a variable
instance name
name of the struct
initial values for struct items
body of struct
Comma after the last property is optional

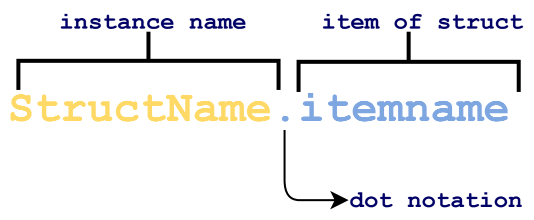
```
let my_name = StructName {
    item1:value,
    item2:value2,
    .
    .
    .
    itemN:valueN,
}
```

Initialize a struct

> **Note:** The order in which you assign values to items does not matter.

Access Values from a Struct

To access any value from the struct write the struct name followed by the `.` operator and then the name of the item to be accessed.

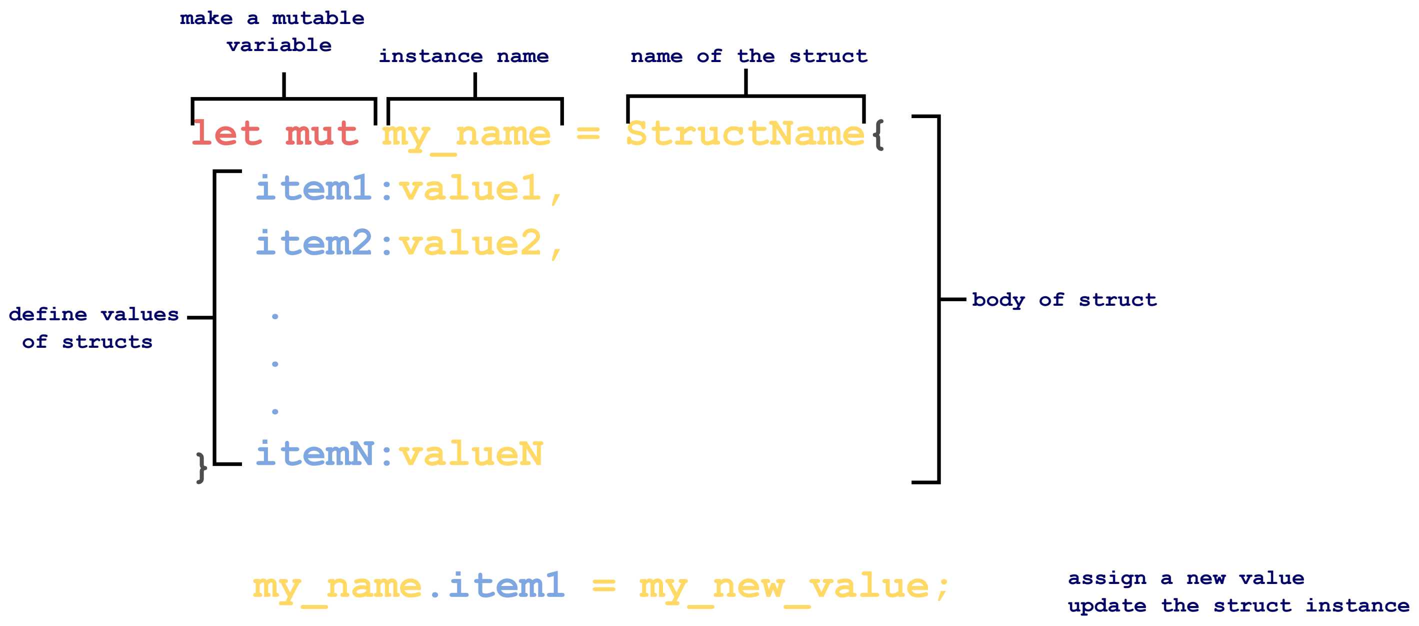


Access a value

> **Note:** A struct instance is immutable by default. Therefore it cannot be updated unless made mutable. However, the values can be accessed.

Update a Struct Instance

A struct instance can be made mutable by adding a `mut` keyword after the `let` followed by the instantiation of the struct. Now that the struct instance is mutable, any item can be accessed using the dot operator and the value of the item can be updated.



Update a struct

Example

The following example creates a `struct` named `Course` and defines three items of it: course name, course level, and course code.

```
//declare a struct
```

```
struct Course {
   code:i32,
   name:String,

   level:String,
}

fn main() {
   //initialize
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130,
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122,
   };
   //access
   println!("Name:{}, Level:{}, code: {}", course1.name, course1.level, course1.code);
   println!("Name:{}, Level:{}, code: {}", course2.name, course2.level, course2.code);
   //update
   course1.name = "Java".to_string();
   course1.code = 134;
   println!("Name:{}, Level:{} ,code: {}", course1.name, course1.level, course1.code);
}
```

Explanation

- `main` **function**

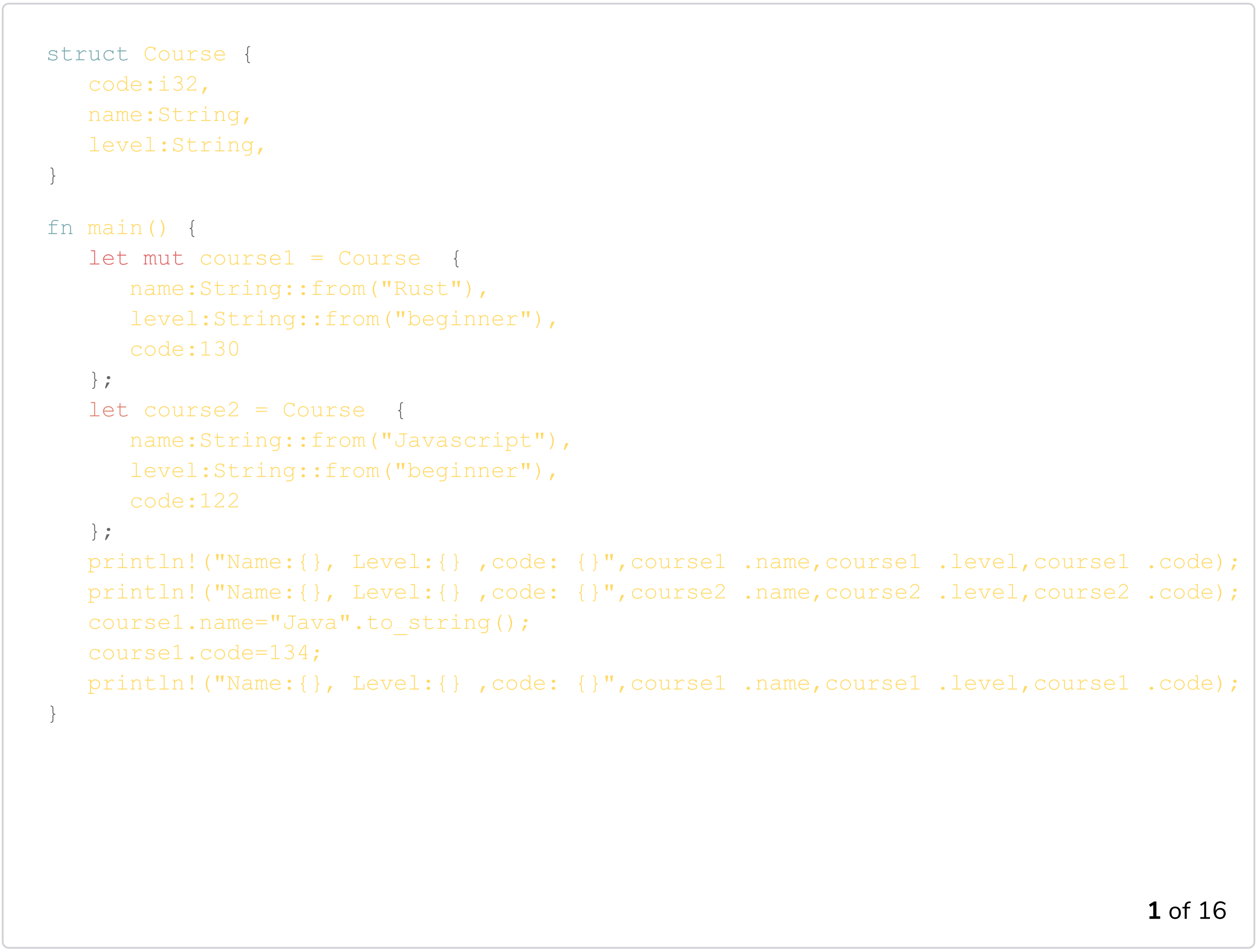
  From **line 8 to line 27**, `main` function is defined.

  - **Line 10 to line 14**, creates a mutable `mut` instance `course1` of the struct `Course`.
  - **Line 15 to line 19**, creates an immutable instances `course2` of the struct `Course`.
  - **Line 21** and **line 22**, prints the values of the instances `course1` and `course2` using dot operator.
  - Since instance 1 `course1` is mutable, the value of `name` and `code` can be updated for instance 1 on **line 24 and 25**.
  - the updated instance is displayed on **line 26**.

- `struct`

  On **line 2**, a `struct` `Course` is declared. Within the struct body, three items namely `code`, `name`, `level` are declared of type `i32`, `String` and `String` respectively.

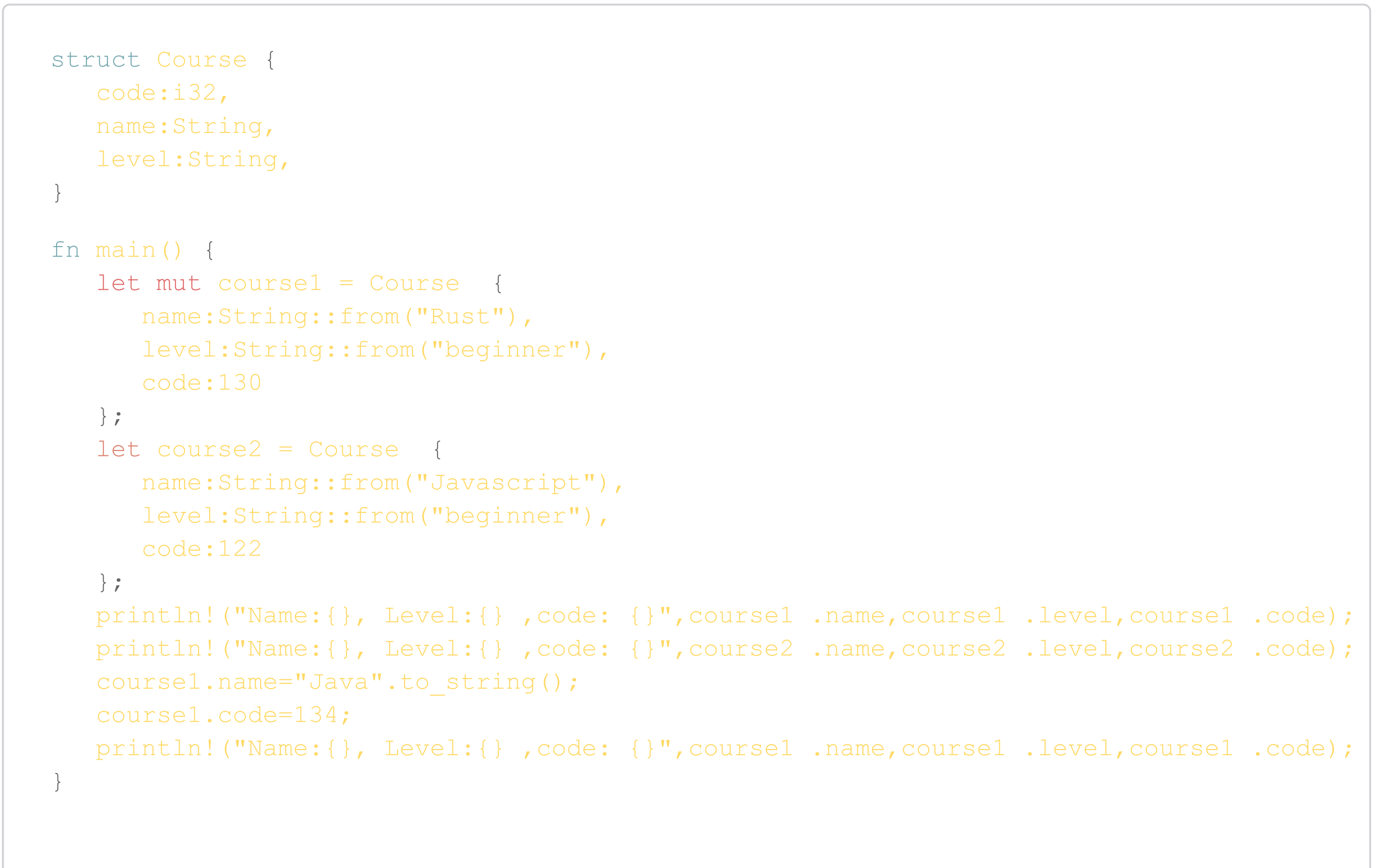
The illustration below explains the code:



```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

1 of 16



```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
Name:Rust, Level:beginner ,code: 130
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
Name:Rust, Level:beginner ,code: 130
Name:Javascript, Level:beginner ,code: 122
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
Name:Rust, Level:beginner ,code: 130
Name:Javascript, Level:beginner ,code: 122
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
Name:Rust, Level:beginner ,code: 130
Name:Javascript, Level:beginner ,code: 122
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
}
```

```
Name:Rust, Level:beginner ,code: 130
Name:Javascript, Level:beginner ,code: 122
Name:Java, Level:beginner ,code: 134
```

```
struct Course {
   code:i32,
   name:String,
   level:String,
}

fn main() {
   let mut course1 = Course  {
      name:String::from("Rust"),
      level:String::from("beginner"),
      code:130
   };
   let course2 = Course  {
      name:String::from("Javascript"),
      level:String::from("beginner"),
      code:122
   };
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
   println!("Name:{}, Level:{} ,code: {}",course2 .name,course2 .level,course2 .code);
   course1.name="Java".to_string();
   course1.code=134;
   println!("Name:{}, Level:{} ,code: {}",course1 .name,course1 .level,course1 .code);
} end of program
```

```
Name:Rust, Level:beginner ,code: 130
Name:Javascript, Level:beginner ,code: 122
Name:Java, Level:beginner ,code: 134
```

Quiz

Test your understanding of the basics of structs.

Quick Quiz on Basics of Structs!

1 The values of struct can be accessed through a:

2 The name of the struct should be in which case to avoid compiler warning?

Retake Quiz

Now that you have learned the basics of structs in Rust, let’s move on to the next lesson “Functions and Structs”.