Struct arguments in Swift are the values that are passed to a struct when it is initialized. Structs can have multiple arguments, and each argument must have a type and a name.

To define a struct with arguments, you use the `struct` keyword, followed by the struct name and a pair of parentheses. Inside the parentheses, you list the struct's arguments, separated by commas. Each argument is defined using a type and a name.

For example, the following code defines a struct called `Person` with two arguments: `name` and `age`:

Swift
```swift
struct Person {
    let name: String
    let age: Int
}
```

To initialize a `Person` struct, you pass the desired values for the `name` and `age` arguments to the struct's initializer. For example, the following code initializes a `Person` struct with the name "John Doe" and the age 30:

Swift
```swift
let person = Person(name: "John Doe", age: 30)
```

You can also use struct arguments to create default values for struct properties. For example, the following code defines a `Person` struct with two arguments: `name` and `age`, with default values of "" and 0, respectively:

Swift
```swift
struct Person {
    let name: String = ""
    let age: Int = 0
}
```

This means that you can initialize a `Person` struct without passing any arguments, and the struct will use the default values for the `name` and `age` properties.

Struct arguments are a powerful feature of Swift that allows you to create custom data types that are tailored to your specific needs. By using struct arguments, you can create structs that are easy to initialize and use.

Here are some additional tips for using struct arguments in Swift:

- You can use struct arguments to define optional properties. An optional property is a property that can have a value or be nil. To define an optional property, you add a question mark (`?`) after the property's type.
- You can use struct arguments to define variadic properties. A variadic property is a property that can have an arbitrary number of values. To define a variadic property, you add three dots (`...`) after the property's type.
- You can use struct arguments to define nested structs. A nested struct is a struct that is defined inside another struct. To define a nested struct, you simply define the nested struct inside the curly braces of the outer struct.

Overall, struct arguments are a valuable tool for any Swift developer. They allow you to create custom data types that are easy to initialize and use.