

An associated function for a trait was defined to be a method (i.e., to take a `self` parameter), but an implementation of the trait declared the same function to be static.

Erroneous code example:

```
trait Foo {  
    fn foo(&self);  
}  
  
struct Bar;  
  
impl Foo for Bar {  
    // error, method `foo` has a `&self` declaration in the trait, but not in  
    // the impl  
    fn foo() {}  
}
```

When a type implements a trait's associated function, it has to use the same signature. So in this case, since `Foo::foo` takes `self` as argument and does not return anything, its implementation on `Bar` should be the same:

```
trait Foo {  
    fn foo(&self);  
}  
  
struct Bar;  
  
impl Foo for Bar {  
    fn foo(&self) {} // ok!  
}
```