```
var kettle = Appliance()
kettle.manufacturer = "Megappliance, Inc"
kettle.model = "TeaMaster 5000"

// later, drop out of scope
```

# ARC

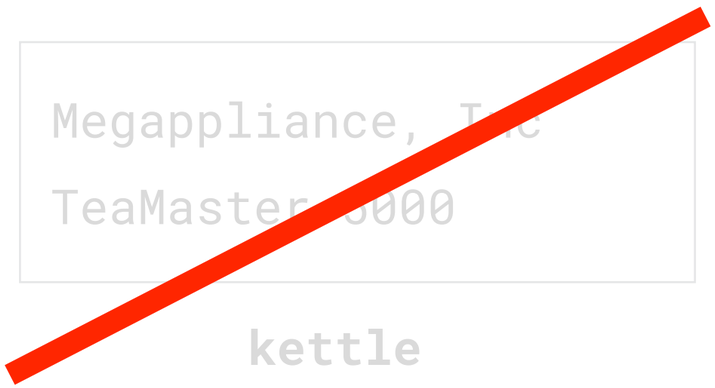
**Automatic Reference Counting**

```
var kettle = Appliance()
kettle.manufacturer = "Megappliance, Inc"
kettle.model = "TeaMaster 5000"

// later, drop out of scope
```

# ARC

**Automatic Reference Counting**



# Revisiting Structs

```
struct color {
    int red;
    int green;
    int blue;
    // no functions
};
```

# C-style Structs

**"A container for variables"**

```
struct SomeStruct {
    // properties,
    // methods,
    // initializers,
    // protocols,
    // subscripts...
}
```

# Swift Structs

**Have similar capability to Classes**

```
struct SomeStruct {
    // properties,
    // methods,
    // initializers,
    // protocols,
    // subscripts...
}
```

| | |
|---|---|
| **Int** | **Array** |
| **String** | **Dictionary** |
| **Double** | **Set** |
| **Bool** | (etc.) |

# Swift Structs

**Have similar capability to Classes**

## Structs
(and Enums)

**Value** types

Assign it to a new variable or constant?
The value is **copied**.

Pass it into a function?
The value is **copied**.

## Classes

**Reference** types

Assign it to a new variable or constant?
**Not** copied - a **reference** is passed.

Pass it into a function?
**Not** copied - a **reference** is passed.

# Stored Properties

```
class MyClass {

    // properties
    let name: String
    var width: Int
    var height: Int

    // ...
}
```

# Stored Properties

```
struct MyStruct {

    // properties
    let name: String
    var width: Int
    var height: Int

    // ...
}
```

# Stored Properties

```
enum MyEnum {
    case one, two, three
    // properties
    let name: String
    var width: Int
    var height: Int

    // ...
}
```

# Stored Properties

```
class MyClass {

    // properties
    let name: String
    var width: Int
    var height: Int

    // ...
}
```