Reducing Bugs with Immutable Data

Nathan Taylor
SOFTWARE ENGINEER

@taylonr taylonr.com

What Is Immutable Data?

```
def addToCart (cart, item) do
    cart.items.concat(item)
    cart.total += item.price
    cart
end
```

```
def addToCart (cart, item) do
    cart.items.concat(item)
    cart.total += item.price
    cart
end
```

```
def addToCart (cart, item) do
    cart.items.concat(item)
    cart.total += item.price
    cart
end
```

```
def addToCart (cart, item) do
    cart.items.concat(item)
    cart.total += item.price
    cart
end
```

```
def addToCart (cart, item) do
    cart.items.concat(item)
    cart.total += item.price
    cart
end
```

Functional Programming

A programming paradigm that treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data.

Immutable Data

An object whose state cannot be modified after it is created.

How can my program do anything if the data can't change?

```
def addToCart (cart, item) do
    cart.items.concat(item)
    cart.total += item.price
    cart
end
```

```
def addToCart (cart, item) do
    cart
end
```

How can programs work without changes?

```
def addToCart (cart, item) do
    %{
        items = cart.items.concat(item),
        total = cart.total + item.price
    }
end
```

```
def addToCart (cart, item) do
    %{
        items = cart.items.concat(item),
        total = cart.total + item.price
    }
end
```

```
def addToCart (cart, item) do
    %{
        items = cart.items.concat(item),
        total = cart.total + item.price
    }
end
```

```
def addToCart (cart, item) do
    %{
        items = cart.items.concat(item),
        total = cart.total + item.price
    }
end
```

How do you call functions with immutable data?

How Does Immutability Reduce Bugs?

```
def addToCart (cart, item) do
    %{
        items = cart.items.concat(item),
        total = cart.total + item.price
    }
end
```

```
book = %{title = "Don't Waste Your Life", author = "John
Piper", quantity = 1, price = 10.00}


cart = addToCart(%{items = []}, book)

cartWithMultipleCopies = updateQuantity(cart, book, 2)

cartWithDiscount = applyDiscount(cartWithMultipleCopies,
"20Off")
```

```
book = %{title = "Don't Waste Your Life", author = "John
Piper", quantity = 1, price = 10.00}


cart = addToCart(%{items = []}, book)

cartWithMultipleCopies = updateQuantity(cart, book, 2)

cartWithDiscount = applyDiscount(cartWithMultipleCopies,
"20Off")
```

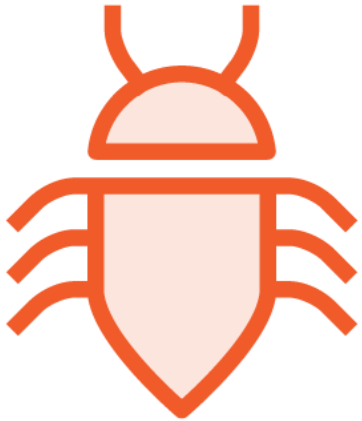
```
%{
    items = [%{title = "Don't Waste Your Life", quantity = 1 ...}],
    total = 10.00
}
```

```
book = %{title = "Don't Waste Your Life", author = "John
Piper", quantity = 1, price = 10.00}


cart = addToCart(%{items = []}, book)

cartWithMultipleCopies = updateQuantity(cart, book, 2)

cartWithDiscount = applyDiscount(cartWithMultipleCopies,
"20Off")
```

```
%{
    items = [%{title = "Don't Waste Your Life", quantity = 2 ...}],
    total = 20.00
}
```

```
book = %{title = "Don't Waste Your Life", author = "John
Piper", quantity = 1, price = 10.00}


cart = addToCart(%{items = []}, book)

cartWithMultipleCopies = updateQuantity(cart, book, 2)

cartWithDiscount = applyDiscount(cartWithMultipleCopies,
"20Off")
```

```
%{
    items = [%{title = "Don't Waste Your Life", quantity = 2 ...}],
    total = 16.00
}
```



How does this reduce bugs?

```
var letters = new List<string>{"C", "a", "b", "A"};
var otherLetters = letters;

letters.Sort();

Console.WriteLine(String.Join("\n", otherLetters);
```

```
a
A
b
C
```

```
var letters = new List<string>{"C", "a", "b", "A"};
var otherLetters = letters;
<30 lines of code … >

letters.Sort();

<25 lines of code … >

Console.WriteLine(String.Join("\n", otherLetters);
```

Threads can cause difficult bugs

Immutability makes code easier to reason about

Performance of Immutability

```
book = %{title = "Don't Waste Your Life", author = "John
Piper", quantity = 1, price = 10.00}


cart = addToCart(%{items = []}, book)

cartWithMultipleCopies = updateQuantity(cart, book, 2)

cartWithDiscount = applyDiscount(cartWithMultipleCopies,
"20Off")
```

Cart

```
%{
    items = [%{title = "Don't Waste Your Life", quantity = 1}],
    total = 10.00
}
```

updatedCart

```
updatedCart = %{
    items = [%{title = "Don't Waste Your Life", quantity = 1 ...},
             %{title = "You Are What You Love", quantity = 1 ...}],
    total = 23.00
}
```

```
updatedCart = 0xA43E278F
updatedCart.items = 0xBC7F4879 = [0xCF8A32D4, 0xD585F20A]
updatedCart.total = 0xEE82F10F
```

Variable Hash

```
updatedCart = 0xA43E278F
updatedCart.items = 0xBC7F4879 = [0xCF8A32D4, 0xD585F20A]
updatedCart.total = 0xEE82F10F
```

Variable Hash

```
updatedCart = 0xA43E278F
updatedCart.items = 0xBC7F4879 = [0xCF8A32D4, 0xD585F20A]
updatedCart.total = 0xEE82F10F
```

Variable Hash

```
updatedCart = 0xA43E278F
updatedCart.items = 0xBC7F4879 = [0xCF8A32D4, 0xD585F20A]
updatedCart.total = 0xEE82F10F
```

Variable Hash

```
updatedCart = 0xA43E278F

updatedCart.items = 0xBC7F4879 = [0xCF8A32D4, 0xD585F20A]

updatedCart.total = 0xEE82F10F
```

Variable Hash

Cart with Free Items

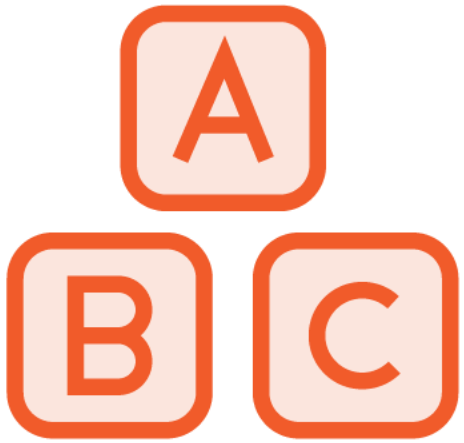
```
cartWithFreeItem = %{
    items = [%{title = "Don't Waste Your Life", quantity = 1, price =
10.00},
      %{title = "You Are What You Love", quantity = 1, price = 13.00},
      %{title = "Bookmark", quantity = 1, price = 0}],
    total = 23.00
}
```

```
cartWithFreeItem = 0xB7CC942F

cartWithFreeItem.items = 0xC002F3D2 = [0xCF8A32D4, 0xD585F20A,
0xE3A91F2D]

cartWithFreeItem.total = 0xEE82F10F
```

Free Item Hash

```
updatedCart.items (0xBC7F4879) != cartWithFreeItem.items (0xC002F3D2)
```

Determine Equality from Hashes



Simplified illustration

Immutable data is performant