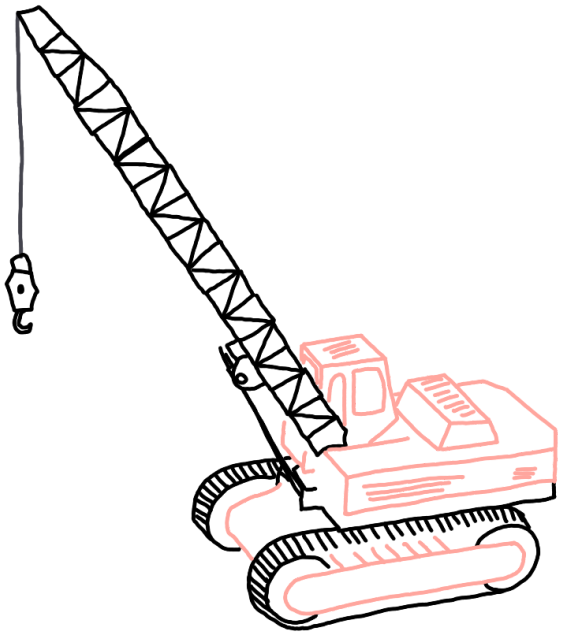
Functors and lifting



Functors are like cranes lifting a function into a context

```
class Functor f where
    fmap :: (a -> b) -> f a -> f b
```

It's like function application in the context of f:

```
  ($)    ::              (a -> b) ->   a ->   b
 (<$>)   :: Functor f => (a -> b) -> f a -> f b
```

Let's work through an example

```
let replace = const "a" <- function to apply
let value = [ Just [ "this", "that" ] ] <- variable to change

replace value <- apply to []
"a"

fmap replace value <- lift once - apply to Just
[ "a" ]

(fmap . fmap) replace value <- lift twice - apply to the internal list
[ Just "a" ]

(fmap . fmap . fmap) replace value <- lift thrice - apply to the elements of the internal list
[ Just [ "a", "a" ] ]

(fmap . fmap . fmap . fmap) replace value <- lift one last time - apply to each letter of each string
[ Just [ [ "a", "a", "a", "a" ], [ "a", "a", "a", "a" ] ] ]
```