Cheatography

Haskell Typeclasses Cheat Sheet

by cawinkelmann via cheatography.com/91540/cs/20520/

Functor Definitions

```
class Functor f where
   fmap :: (a -> b) -> f a -> f b
    (<$) :: a -> f b -> f a
    (<$) = fmap . const
-- Example Instances
instance Functor ((->) r) where
   fmap = (.)
instance Functor [a] where
   fmap = map
instance Functor Maybe where
   fmap _ Nothing = Nothing
   fmap f (Just a) = Just (f a)
instance Functor IO where
   fmap f x = x >>= (pure . f)
```

Functor Laws:

Functors must preserve identity morphisms

```
fmap id = id
```

Functors preserve composition of morphisms

```
fmap (f . g) = fmap f . fmap g
```

Applicative Functor

```
class Functor f => Applicative f where
       {-# MINIMAL pure, ((< *>) | liftA2) #-}
       pure :: a -> f a
        (<* >) :: f (a -> b) -> f a -> f b
        (<* >) = liftA2 id
        liftA2 :: (a -> b -> c) -> f a -> f b -> f c
        liftA2 f x = (<* >) (fmap f x)
        (*>) :: f a -> f b -> f b
       a1 a2 = (id <$ a1) < > a2
        (<*) :: f a -> f b -> f a
        (<*) = liftA2 const
-- Example instances
instance Applic ative Maybe where
       pure = Just
       Just f <*> m = fmap f m
        Nothing <*> _m = Nothing
        liftA2 f (Just x) (Just y) = Just (f x y)
        liftA2 _ _ _ = Nothing
       Just _m1 *> m2 = m2
        Nothing *> _m2 = Nothing
```


By **cawinkelmann**

cheatography.com/cawinkelmann/

Not published yet.

Last updated 14th September, 2019.

Applicative Functor (cont)

```
> instance Applicative IO where
    {-# INLINE pure #-}
    {-# INLINE (*>) #-}
    {-# INLINE liftA2 #-}
    pure = returnIO
    (*>) = thenIO
    (<*>) = ap
    liftA2 = liftM2
```

By **cawinkelmann**

cheatography.com/cawinkelmann/

Not published yet.
Last updated 14th September, 2019.