TCs and Variance

TCs and Variance

How generic variance affects type classes

- recap of variance
- how TCs work with variance

Optional video

- does not (greatly) impact your general Cats skills
- counts to the final certificate

TCs and Variance

Contravariant type classes

- can use the superclass TC instance if nothing available strictly for that type

```
trait SoundMaker[-T]
implicit object AnimalSoundMaker extends SoundMaker[Animal]
def makeSound[T](implicit soundMaker: SoundMaker[T]): Unit = ???

makeSound[Animal] // ok
makeSound[Cat] // also ok, uses the Animal instance
```

Covariant type classes

- will always use the more specific instance

```
trait AnimalShow[+T]
implicit object GeneralAnimalShow extends AnimalShow[Animal]
implicit object CatShow extends AnimalShow[Cat]
def organizeShow[T](implicit event: AnimalShow[T]) = ???

// organizeShow[Animal] // doesn't compile - both instances are applicable
organizeShow[Cat] // ok, uses the more specific one
```

Cats TCs and Variance

Can't have both

- ability to use the superclass TC instance
- preference for more specific TC instance

Cats has invariant TCs

```
import cats.Eq
import cats.syntax.eq._
import cats.instances.int._
import cats.instances.option._

// Some(2) === None // doesn't compile - no specific Eq[Some] exists, can't choose Eq[Option[Int]]
Option(2) === Option.empty[Int] // compiles - Eq[Option[Int]] chosen
```

Takeaway: use the general type & "smart" constructors

Cats rock