Generics and associated types

What is a generic

```rust
trait Eat<F: Food> {
    fn eat(food: F);
}
```

What is an associated type

```
trait Barf {
    type Output;

    fn barf() -> Self::Output;
}
```

What is the difference

When to use generics?

When it makes sense to implement it for many types:

```rust
impl ServeDinner<CatFood> for Cat {...}
impl ServeDinner<DogFood> for Cat {...}
```

When to use associated types?

When it makes sense to implement it for one type only:

```rust
impl Barf for Cat {
    type Output = CatFood;

    fn barf() -> Self::Output {
        CatFood {}
    }
}
```

Are generics strictly more powerful than associated types then?

Build-a-bear

```rust
struct Factory {}

trait FoodFactory<A: Animal> {
    type FoodOutput;

    fn produce(animal: A) -> Self::FoodOutput;
}

impl FoodFactory<Cat> for Factory {
    type FoodOutput = CatFood;

    fn produce(animal: Cat) -> Self::FoodOutput {
        println!("Putting {:?} into a blender", animal);
        CatFood {}
    }
}
```

Smart implementations

```
~~impl Distribute<CatFood> for Factory~~
```

```
impl Distribute<<Factory as FoodFactory<Cat>>::FoodOutput> for Factory
```