U5AEH3L05 : Makes sense -- I'm still not as comfortable with folds as i'd like to be!
U23SA861Y : fold is sort of the ultimate traversal primative
U23SA861Y : you can use it to implement map for example. Getting a feel for fold helps alot.
U5AEH3L05 : it hadn't crossed my mind to use fold with a tuple before, but <@U48AEBJQ3> had a nice implementation with it
U4F64AKQV : Might be a good addition to List.Extra. Consider submitting a PR.
U23SA861Y : can get even better I think
U5AEH3L05 : I'll try and chip away at it for fun. should I prefer foldl vs foldr for any reason?
U23SA861Y : it depends on what you want/need
U23SA861Y : foldl starts processing right away which foldr needs to traverse to the bottom before it processes back up
U23SA861Y : you might even be able to do it with map
U23SA861Y : ```scanl: (a -> a -> b) -> List a -> List b
scanl fn l =
    List.tail l
        |> Maybe.withDefault []
        |> (\x -> List.map2 (,) x l)
        |> List.map (uncurry fn)
```

U5AEH3L05 : I like that -- split it into two lists and then operate on those
U48AEBJQ3 : How about```
    List.Extra.zip
        xs
        (List.tail xs |&gt; Maybe.withDefault [])
        |&gt; List.map (uncurry f)
```

U23SA861Y : what you doing with List.extra in there
U23SA861Y : you don't need that
U48AEBJQ3 : Because `List.Extra` already implements zip?
U4F64AKQV : <@U48AEBJQ3> That last implementation is probably as nice as it will get.
U23SA861Y : zip is just map2
U23SA861Y : I wouldn't pull an import just for (,)
U4F64AKQV : In all likelihood you'll be using other stuff from List.Extra anyway
U23SA861Y : ```scanl: (a -> a -> b) -> List a -> List b
scanl fn l =
    List.tail l
        |> Maybe.withDefault []
        |> List.map2 (flip fn) l
```

U5DDM498S : what tools would you recommend to mock RESTful APIs?
```