How Stateless Can You Go?

Few Moving Parts = Robust

Stateless Code = Fewer Moving Parts

Easier to Test & Debug

Possibly Easier Multithreading

Functional Languages

Haskell
Clojure
Erlang
F#
...

But Applies To Any Language!

Stateful

```
sumOfElements(array){
  var sum = 0;
  for (var i=0; i < array.length; i++) {
       sum += array[i];
  }
  return sum;
}
```

Less state

```
sumOfElements(array){
  var sum = 0;
  _(array).each(function(element){
       sum += element;
  });
  return sum;
}
```

Stateless

```
sumOfElements(array){
  return _(array).reduce(function(sum, element) {
       return sum + element;
  }, 0);
}
```

Your Mission

Solve A Problem Using Minimal State

Tactics

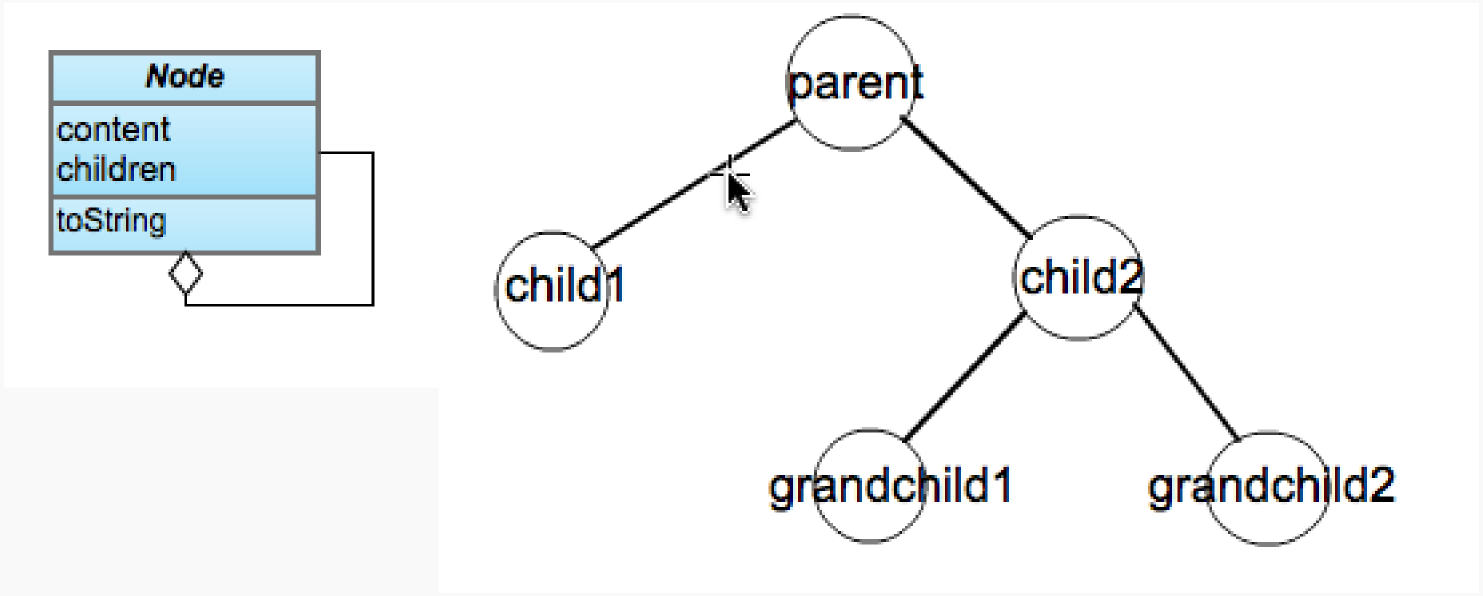
Start with “normal” code
Then refactor away state
Eliminate mutable variables
Rely on functions & constants

Practice Only This One Thing!

Use Any Language, Tool, Method

Partner Up!
Then Try This:

Implement a simple api. A tree consists of Nodes. Each node has a collection of child nodes. Each Node also has a “content” property which could be a string. In the following illustrations, each nodes “content” is a string representing its place in the tree, ie. “parent” for the root node, “child1-2” for its children etc.



Now, implement the method ‘toString’ which returns out a properly indented string representation of the tree, like this:

```
parent
    child1
    child2
        grandchild1
        grandchild2
```