Approach in Rust

We have a Stream. This ideally has to a Rust Stream or an Iterator
Stream Resources:

So we have a trait called `Streamable`. This has to be implemented by something in order to work with Approach.

```
pub trait Streamable{
  fn render() -> string
  fn stream() -> void
  fn RenderHead(&output_stream) -> void
  fn RenderCorpus(&output_stream) -> void
  fn RenderTail(&output_stream) -> void
}
```

It looks something like this.
Essentially, we would want to stream stuff into the Stream.
Now this stream can be an `async` stream in rust or it can be just an `Iterable` that yield stuff.

However, we also have a stream passed into a Node as well which has a bunch of `Streamables`, so like this

```
struct Stream{ Box<dyn Streamable>; Streamable nodes; }
struct Node{ Box<dyn Stream>; String content; }
```

Now the render function has something like this for *each* `Streamable`

```
fn render(){
        RenderHead();
        RenderCorpus();
        RenderTail();
}
```

Hence finally, we can put out something like this

```
fn stream(output_stream){
    self.StreamHead( &output_stream );
    self.StreamCorpus( &output_stream );
```

```
    self.StreamTail( &output_stream ); // Probably cloned, idk
}

fn render(){
  return self.stream();
}

fn RenderHead(){
  return self.StreamHead();
}
fn RenderCorpus(){
  return self.StreamCorpus();
}
fn RenderTail(){
  return self.StreamTail();
}
```

Here, the stream, render, `RenderHead` and the `RenderTail` belong to a `Streamable` and the output_stream refers the `Iterable` we passed in as the output.