this are the necessary imports for this project

⟨*import statements*⟩≡
```
var lexer = require("./lexer");
```

there are three main parts for the analysis part of the program

⟨*lexical analysis*⟩≡
```
/*
  @param1 = the source text of the jade file
  @return = returns a sequence of tokens/symbols, more information about tokens
              is available at --REF--
*/
function lexicalAnalysis(sourceText){
  return lexer(sourceText);
}
```

this is the part that will conver the syntactic analysis to a syntax tree

⟨*syntactic analysis*⟩≡
```
/*
  @param1 tokens_list = this is a list of tokens that got parsed from the jade code file.
              more information about tokens is available at -REF--
  @return = This function will return the abstract syntax tree that is produced from the t
*/
function syntacticAnalysis(tokens_list){
  return [];
}
```

this is the contextual analysis part

⟨*contextual analysis*⟩≡
```
/*
  @param1 syntaxTree = this is the syntax tree
  @return = a syntax tree with contextual information, in this case it is adding the infor
  needed for the variables. More information is available at --REF--.
*/
function contextualAnalysis(syntaxTree){
  return syntaxTree;
}
```

this is the parse function

⟨*parse*⟩≡
```
/*
  @param1 sourceText = the source text which would be the jadeimp code as a string
  @return = this will be the abstract syntax tree with contextual analysis
*/
function parse(sourceText){
  return contextualAnalysis(syntacticAnalysis(lexicalAnalysis(sourceText)));
}
```

this is the api portion of the code, meaning the part of the code that is visible from other files. For the porposes of the analysis portion, we only need to make the analysis portion of the program public because other programs do not need access to the other functions. By hiding the other functions we allow this code to be more abstract and changable if needed.

⟨*api*⟩≡
```
// this is the api of the program, the only function that should be visible from outside
// this file is parse.
module.exports.parse = parse;
```

this part will simply produce the output of the program, in this order

⟨*analyzer.nw*⟩≡
⟨*lexical analysis*⟩
⟨*syntactic analysis*⟩
⟨*contextual analysis*⟩
⟨*parse*⟩
⟨*api*⟩