A COMPILATION OF ERRORS

There are many kinds of errors: compile-time errors, logical errors, run-time errors and generated errors. I'll focus on compile-time errors in this section and go through the others in the next sections.

Compile-time errors are often syntactic mistakes: check your function names, the tokens in the language (brackets, parentheses, periods, comas), the arity of your functions, etc. Here's a list of some of the common compile-time error messages and potential resolutions in case you encounter them:

module.beam: Module name 'madule' does not match file name 'module'

The module name you've entered in the `-module` attribute doesn't match the filename.

./module.erl:2: Warning: function some_function/0 is unused

You have not exported a function, or the place where it's used has the wrong name or arity. It's also possible that you've written a function that is no longer needed. Check your code!

./module.erl:2: function some_function/1 undefined

The function does not exist. You've written the wrong name or arity either in the `-export` attribute or when declaring the function. This error is also output when the given function could not be compiled, usually because of a syntax error like forgetting to end a function with a period.

./module.erl:5: syntax error before: 'SomeCharacterOrWord'

This happens for a variety of reason, namely unclosed parentheses, tuples or wrong expression termination (like closing the last branch of a `case` with a comma). Other reasons might include the use of a reserved atom in your code or unicode characters getting weirdly converted between different encodings (I've seen it happen!)

./module.erl:5: syntax error before:

All right, that one is certainly not as descriptive! This usually comes up when your line termination is not correct. This is a specific case of the previous error, so just keep an eye out.

./module.erl:5: Warning: this expression will fail with a 'badarith' exception

Erlang is all about dynamic typing, but remember that the types are strong. In this case, the compiler is smart enough to find that one of your arithmetic expressions will fail (say, `llama + 5`). It won't find type errors much more complex than that, though.

./module.erl:5: Warning: variable 'Var' is unused

You declared a variable and never use it afterwards. This might be a bug with your code, so double-check what you have written. Otherwise, you might want to switch the variable name to `_` or just prefix it with an underscore (something like *`_Var`*) if you feel the name helps make the code readable.

./module.erl:5: Warning: a term is constructed, but never used

In one of your functions, you're doing something such as building a list, declaring a tuple or an anonymous function without ever binding it to a variable or returning it. This warning tells you you're doing something useless or that you have made some mistake.

./module.erl:5: head mismatch

It's possible your function has more than one head, and each of them has a different arity. Don't forget that different arity means different functions, and you can't interleave function declarations that way. This error is also raised when you insert a function definition between the head clauses of another function.

./module.erl:5: Warning: this clause cannot match because a previous clause at line 4 always matches

A function defined in the module has a specific clause defined after a catch-all one. As such, the compiler can warn you that you'll never even need to go to the other branch.

./module.erl:9: variable 'A' unsafe in 'case' (line 5)

You're using a variable declared within one of the branches of a `case ... of` outside of it. This is considered unsafe. If you want to use such variables, you'd be better of doing `MyVar = case ... of`...

This should cover most errors you get at compile-time at this point. There aren't too many and most of the time the hardest part is finding which error caused a huge cascade of errors listed against other functions. It is better to resolve compiler errors in the order they were reported to avoid being misled by errors which may not actually be errors at all. Other kinds of errors sometimes appear and if you've got one I haven't included, send me an email and I'll add it along with an explanation as soon as possible.

Source : http://learnyousomeerlang.com/errors-and-exceptions