Question 1

Which of the following statements about Erlang programming is false?

- Erlang uses single assignment, so that an instance of a variable cannot be re-assigned once it has a value. It is, however, possible to pattern match against bound variables.
- **Evaluation in Erlang is demand-driven: an argument to a function is only evaluated if its value is needed by subsequent computation.**
- Functions in Erlang are "first-class citizens": they can be included in data structures, be passed to functions as parameters, returned as results, and compared using equality and ordering.

Correct

You scored: 2

Simon Thompson LEAD EDUCATOR

No, this is not true, and, for instance the built-in conjunction operation `and` will evaluate both arguments, even if the first evaluates to `false`. There is a "lazy" version of conjunction, `andalso` that will not evaluate its second argument in such a case, and simply return `false`.

Question 2

Which of these statements about type checking in Erlang is correct?

- [x] **Most type checking for Erlang is performed at runtime, even in some cases when it is possible to check a condition at compile time.**
- [ ] Type checking for Erlang could take place entirely at compile time, so that no type errors would be generated from running code.
- [ ] No type checking takes place at compile time in Erlang.

Correct

You scored: 3

Simon Thompson LEAD EDUCATOR

Yes, a function which returns `true and 3` will fail every time it is called, and this could be predicted at compile time.

PREVIOUS QUESTION

NEXT QUESTION

Question 3

What is the result of evaluating the following expression?

```
lists:foldr(fun(X,Y) when X>Y -> Y; (X,Y) -> X end, 0, [2,1,-4,2,4]).
```

- [x] -4
- [ ] 0
- [ ] 4

Correct

You scored: 3

Simon Thompson LEAD EDUCATOR

Yes, that's right: this gives the minimum value in the list, or zero, whichever is smaller.

PREVIOUS QUESTION

NEXT QUESTION

Question 4

What is the result of evaluating the following expression?

```
lists:foldr(fun(X,Y) -> [X|Y] end, [2,1,-4,2,4], [2,1,-4,2,4]).
```

- [ ] `[4,2,-4,1,2,2,1,-4,2,4]`
- [ ] `[2,1,-4,2,2,1,-4,2,4]`
- [x] **`[2,1,-4,2,4,2,1,-4,2,4]`**

Correct

You scored: 3

Simon Thompson LEAD EDUCATOR

Yes; as you can see, this is another way of defining the ++ operator on lists.

PREVIOUS QUESTION

NEXT QUESTION

Question 5

Which of the following definitions is a correct implementation of a function to return the Nth element (first argument) of a list (second argument).

```
nth(0,[X|_]) -> X;
nth(N,[_|Xs]) -> nth(N-1,Xs);
nth(N,Xs) -> 0.
```

```
nth(0,[X|_]) -> X;
nth(N,[_|Xs]) -> nth(N-1,Xs).
```

```
nth(N,[_|Xs]) -> nth(N-1,Xs);
nth(0,[X|_]) -> X.
```

Correct

You scored: 3

Simon Thompson LEAD EDUCATOR

Yes. Note that this will give an error when indexing "off the end of the list"; that's in line with Erlang's "let it fail" philosophy.

PREVIOUS QUESTION

NEXT QUESTION

Support

Question 6

Which higher-order function would you use to implement `lists:zipwith` assuming that you could call `lists:zip` in your definition?

- The filter function, `lists:filter`.
- The function `lists:splitwith`.
- **The function `lists:map`.**

Correct

You scored: 3

Simon Thompson LEAD EDUCATOR

That's right! Once you have zipped the two lists, then producing the result is a matter of mapping the function along that list of pairs.

PREVIOUS QUESTION

NEXT QUESTION

Question 7

Which of these is *not* a feature of Erlang?

- [x] **Explicit memory allocation.**
- [ ] Compilation to virtual machine code.
- [ ] Garbage collection.
- [ ] Multi-platform.

Correct

You scored: 3

Simon Thompson LEAD EDUCATOR

Yes, Erlang memory is allocated automatically, and it provides no mechanisms for the direct allocation of memory.

PREVIOUS QUESTION

NEXT QUESTION

Question 8

Which of the following is an Erlang atom?

- case
- 'i'm an atom'
- **'case'**
- "case"

Correct

You scored: 3

Simon Thompson LEAD EDUCATOR

Yes, this *is* an atom, but not equal to case without the quotes, which is usually the case.

PREVIOUS QUESTION

NEXT QUESTION

Question 9

Which of the following is not a feature of Erlang?

- Passing a function as an argument.
- Building a list of functions.
- Returning a function as the result of a function.
- **Passing one argument to a two argument function.**

Correct

You scored: 3

Simon Thompson LEAD EDUCATOR

Yes, this is not directly possible, but can be achieved by building a "Curried" form of the function. For example, here is a Curried form of multiplication, `fun(N) -> fun(X) -> X*N end end.` which will return the function that doubles its argument if it is applied to 2.

PREVIOUS QUESTION

NEXT QUESTION

Question 10

Which of the following statements is true?

- Erlang functions cannot be compared for equality.
- The operators == and =:= give the same results when applied to numbers.
- Evaluating `lists:map == lists:filter` will return `false`.
- **Atoms can be compared for equality and ordering.**

Correct

You scored: 3

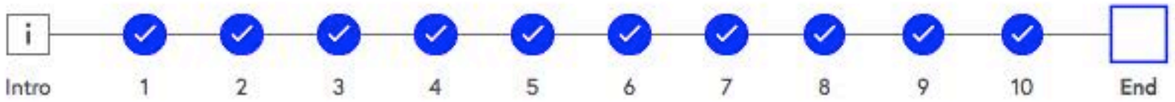
Simon Thompson LEAD EDUCATOR

Yes, and that is *all* that can be done with atoms.

PREVIOUS QUESTION

SUMMARY

How well do you know Erlang?



Summary

29

You've scored 29 out of 30

Moving on from the test, in the final step of the course Simon will round off with a quick overview of what we've learned - and an invite for you to continue exploring Erlang in our follow-up course on concurrent programming.

PREVIOUS QUESTION

SUMMARY AND CONCLUSION VIDEO