JARLANG

Compiling Erlang for the Browser

(Because we can)

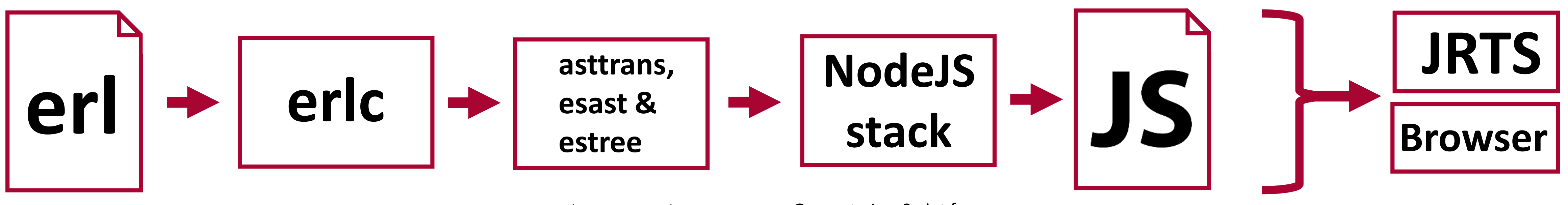
ABOUT

Jarlang is a transpiler written in `Erlang` and `JavaScript` with the ultimate goal of being able to translate a subset of `Erlang` (primarily focusing on the standard library) to valid `JavaScript`.

Jarlang does this by hooking into the `Erlang` compiler which allows us to work in `Erlang's` intermediate language representation which can then be manipulated as an `Abstract Syntax Tree`.

We then simply translate each of the `Abstract Syntax Tree` nodes one by one into their `JavaScript` equivalent and rely on existing `NodeJS` tools such as `escodegen` to generate valid output.

THE JARLANG PIPELINE



Write any normal Erlang code.
Browser APIs exposed through custom module.

Compile with erlc as per usual, allowing us to utilise erlc's optimisations.

Parse erlc output and return AST.
Perform Erlang AST to JavaScript AST translation

Generate JavaScript from AST, and optimise it with 'escodegen', 'BabelJS' and 'Closure Compiler'

Write output to a file ready to be used on the web

Runtime emulates concurrency, message passing, global atom table & more to allow transpiled output to work as intended.

EXAMPLE INPUT & OUTPUT

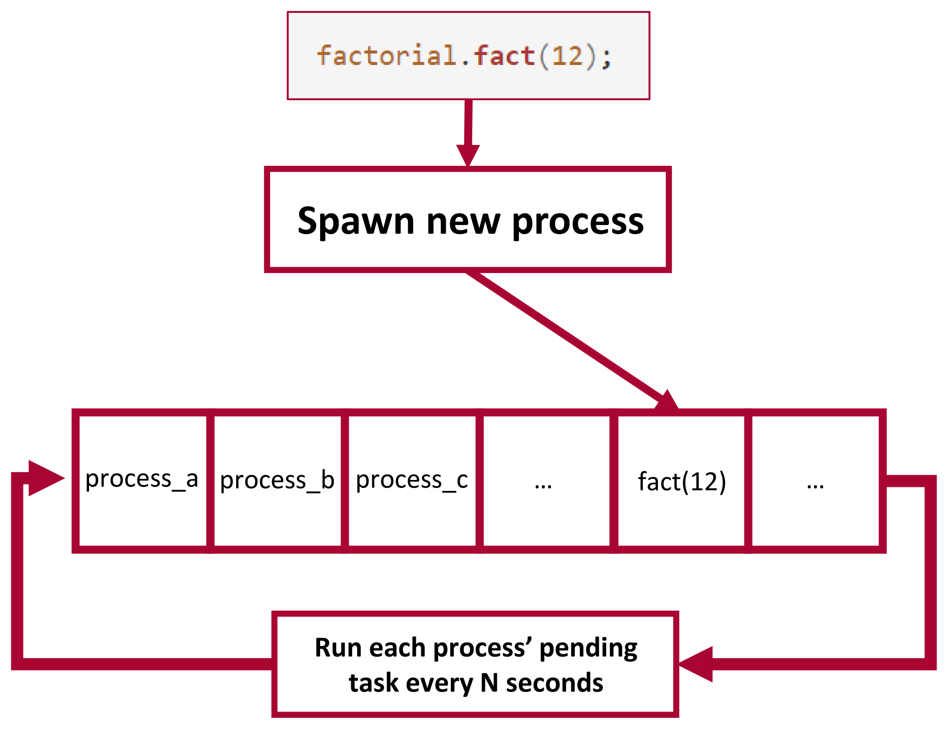
```
fact(N) ->
    fact(N, 1).

fact(0, Res) ->
    Res;
fact(N, Res) ->
    fact(N - 1, Res * N).
```

Jarlang

```
'fact/1': function (_cor0) {
    return functions['fact/2'].bind(this)(_cor0, new Int(1));
},
'fact/2': function (_cor1, _cor0) {
    if (function () {
            let Res = _cor0;
            if (_cor1.match(new Int(0))) { return new Atom('true'); }
        }.bind(this)()) {
        Res = _cor0;
        return Res;
    } else if (function () {
            let N = _cor1, Res = _cor0;
            if (true) { return new Atom('true') }
        }.bind(this)()) {
        N = _cor1;
        Res = _cor0;
        let _cor3 = erlang['-'].bind(this)(N, new Int(1));
        let _cor2 = erlang['*'].bind(this)(Res, N);
        return functions['fact/2'].bind(this)(_cor3, _cor2);
    }
},
```

EMULATING CONCURRENCY



TECH STACK

ERLANG JS node es BABEL Gulp

University of Kent | Computing

By Chris Bailey, Nick Laine & Andrew Johnson. Supervised by Scott Owens.