ROCK SOLID Web Apps
With HASKELL And JAVASCRIPT!
!!!!!!

About Me

- 9+ years as a software developer
- Consulting in Web, Mobile, and Embedded Systems domains
- Currently Bootstrapping a Mobile App Startup
- In Santiago as a part of Startup Chile

I ♥ Programming Languages

- 6+ Years as a Javascript Developer
- 10+ Years as Haskell Developer (on and off)
- Have also worked with other “modern” languages such as Ruby, Python, Erlang, & Lua
- Still in search for the elusive “Sweet Spot” of Web Development

“The Javascript Problem”

- Javascript is *the de facto* platform for web development
- However it wasn't designed for large complex apps
- Javascript is elegant and frustrating at the same time!

This is What JS Feels Like To Me

Javascript Is Huge

- Universal
- Huge Developer Community
- Large Number of Libraries and Frameworks
- Build and Infrastructure tools
  - NPM
  - Grunt
  - Bower
- IDE Support

Javascript Is Elegant

- Higher Order Functions
- Anonymous Functions
- Closures
- Recursion
- Dynamic Objects
- Object and Array Literals

JS Lacks Crucial Functionality

- No Typing
- No Module System or “import” Statements
- No Data Structures
- Broken 'this' keyword and Inheritance Model
- Single Threaded
- Callback Hell

JS Has Lots Of Warts

- args = Array.prototype.slice.call(arguments);
- handler(function() {this.foo()}.bind(this))
- for (v in o) {if (o.hasOwnProperty(v)) {
- (function(newVar) { … })(oldVar);
- addHandler(Klass, 'method', args);
- if(!isNaN(parseFloat(n)) && isFinite(n)) {
- setTimeout(function() { … }, 0);

Transpilers To The Rescue

- Coffeescript
- Typescript
- Purescript
- Dart
- GWT
- Elm
- Roy
- And perhaps a million more

https://github.com/jashkenas/coffeescript/wiki/List-of-languages-that-compile-to-JS

Haskell Solutions

- QuasiQuotation
  - Julius
  - JMacro
- Subset of Haskell
  - Fay
  - Haste
- Full Haskell
  - GHCJS

Haskell JS Interop Used To Be Like This

Now Haskell And JS Are Awesome Together!

Why Haskell + JS

- Elegant
- Concise
- Modular
- Strongly Typed
- Overcome Most (All?) Javascript Warts
- Can Drop Down to JS Whenever Needed
- Focus on Features instead of Workarounds
- Works Best With Haskell on the Server Side

Quasiquoted Haskell Solutions

Julius

- Type Safe JS Includes
- Type Safe Variable Interpolation
- Type Safe URL Interpolation
- “Stick your Haskell into JS – with Type Safety!”

```
$(function(){
    $("section.#{sectionClass}").hide();
    document.location = "@{SomeRouteR}";
    ^{addBling}
});
```

Jmacro

- Programmatic Generation Of JS Code
- QuasiQuoted Syntax mostly compatible with JS
- Syntax is Statically Checked At Compile Time
- Type Safe Variable Interpolation
- Hygienic Names
- A Bunch of Additional Syntactic Features
- “Haskell Macros for Javascript”

Jmacro Syntactic Additions

- Block Level Scoping
- Concise Syntax for Lambdas and Function Application

```
var x = “abcd”
var foo = \x -> bar x + x
```

- Destructuring Bind (a.k.a. Pattern Matching)

```
\[a,b] -> a;
\{| x:u, y:v |} → [u,v]
```

Subset of Haskell Solutions

Fay

- Proper Subset of Haskell
- Converts Haskell Source to JS Source
- Tiny Runtime
- Generates Simple Readable Code
- Supports Cabal (Haskell Package Installer)
- Trivial To Use FFI
- Provides Automatic Marshalling of JS values to Haskell and Vice Versa (The Dispatcher)
- Provides a Fay Monad for Side Effects
- Small Codebase

Sample Fay Program

```
import Prelude

main :: Fay ()
main = print $ fib 10
  where
    fib :: Int -> Int
    fib 0 = 0
    fib 1 = 1
    fib n = fib (n - 1) + fib (n - 2)
```

Optimising Fay “Hello World”

Stage	Size
Vanilla Compilation	62k
Google Closure	3.8k
-O flag	3.7k
--no-dispatcher & --no-builtins	807 bytes
Nested scopes removed and inlining	798 bytes

Fay Generated Output

```
var i=new function(){function f(a){return new
b(function(){var c;if(0===d(a))c=0;else
if(1===d(a))c=1;else{var j=f(g(a,1)),e=f(g(a,2));c=new
b(function(){return d(j)+d(e)})}return c})}function
d(a,c){for(;a instanceof b;)a=a.f(c);return a}function
b(a){this.a=!1;this.value=a}function k(){this.value=void
0}function h(a,
c){this.d=a;this.e=c}function g(a,c){return new
b(function(){return d(a)-d(c)})}
b.prototype.f=function(a){return a?this.value():this.a?
this.value:(this.value=this.value(),this.a=!
0,this.value)};this.b=new b(function(){var a=f(10),c=new
b(function(){for(var c=d(a)+"",e=null,b=c.length-
1;0<=b;b--)e=new h(c[b],e);return e
});return new b(function(){for(var a=c,b="",a=d(a);a
instanceof h;)b+=a.d,a=d(a.e);console.log(b);return new
k})});this.c=d};i.c(i.b);
```

Fay In The Real World

- I worked on a Large Production App About a Year Ago
  - Over 10k lines of Fay
  - 200-300 lines of JS
- Very Pleasant To Use
- If It Compiles, It Usually Works
- Being Able to Share Datatypes Across Server and Client Side is a Big Win!
- You Start Missing Typeclasses After a While
- No Source Maps Makes Debugging Hard
- Fay Forgets Types During Compilation!

Real World Fay Code

```
section :: String -> Dom ()
section name = do
    Namevar ← var name
    h1 $ “Section ” ++ show name
    input $ do
      value name
      attribute "placeholder" "Name of the
section'"
      bind "change" $ \newname → do
        alert $ “Name changed to ” ++ newname
        set namevar newname
        return False

alert :: String → Fay ()
alert = ffi “alert(%1)”
```

Haste

- Like Fay But Supports Almost All Haskell
- GHC Compatible Whenever It Makes Sense
- Parses and Converts Haskell STG To JS
- Small Runtime (Not as Small as Fay)
- Generates Sort Of Readable Code
- Easy To Use FFI
- Provides Automatic Marshalling of JS values to Haskell and Vice Versa
- Compatible With Cabal
- Seamless RPC Calls with Haste.App

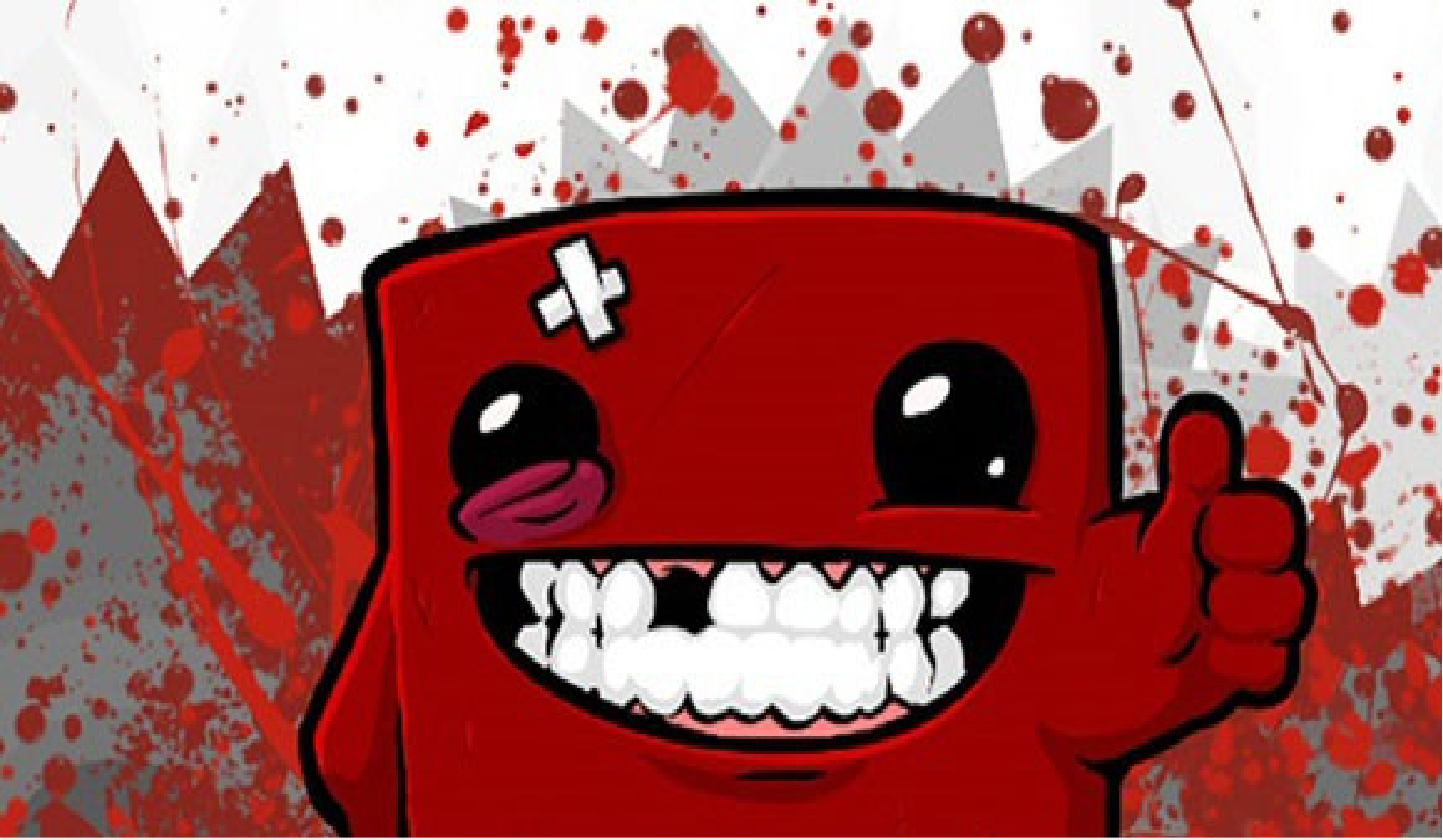
Haste In The Real World

- Haven't Yet Had a Chance to Use Haste In Production But Used It In Several Toy Projects
- Has All The Advantages And Very Few Problems Of Fay
- Generally Less Buggy Than Fay
- Implements All The Haskell That Is Relevant To Javascript Development
- Package Management is Much More natural
- Currently My Favorite Solution

Full Haskell Solutions

GHCJS

- A JS Backend For Haskell
- Supports Everything In Haskell
- Threads, Memory Management, Transactional Memory, IO, Everything!
- FFI to JS Instead of C
- Slow and Complex
- Currently Hard To Get Started With
- Rapidly Improving And May Become A Viable Alternative



Haste Is My Solution of Choice

Thank You!
Questions?