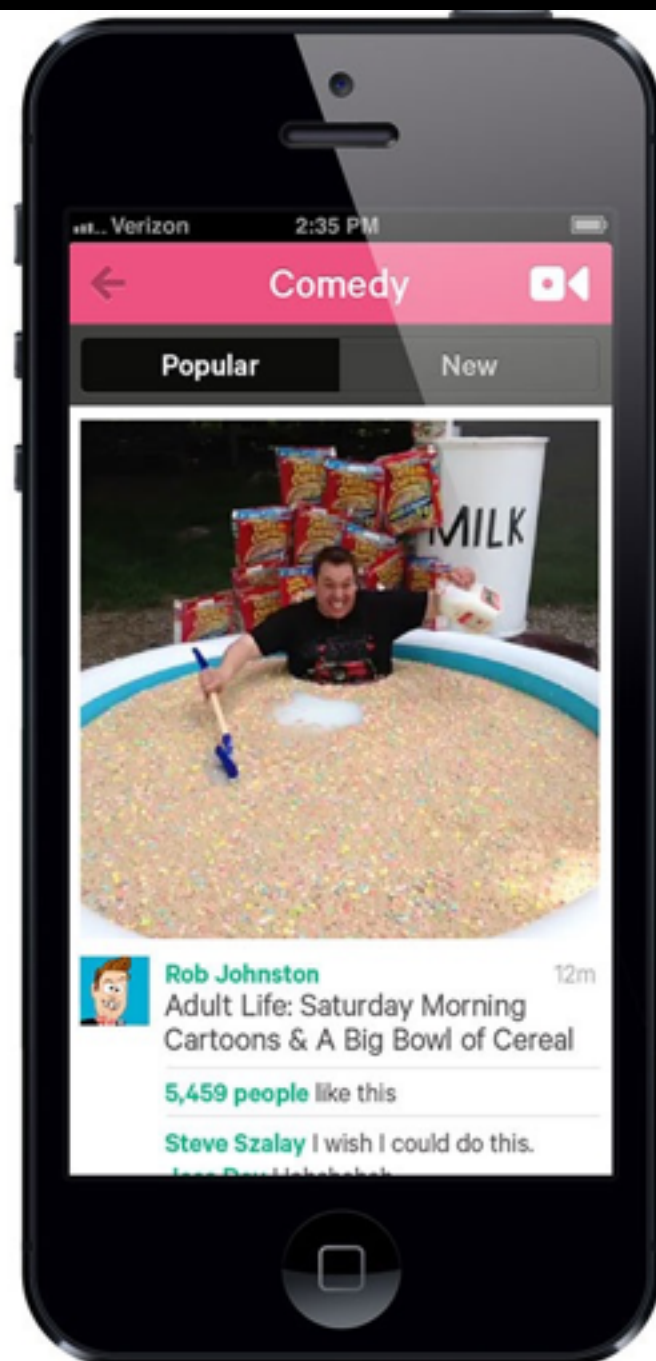
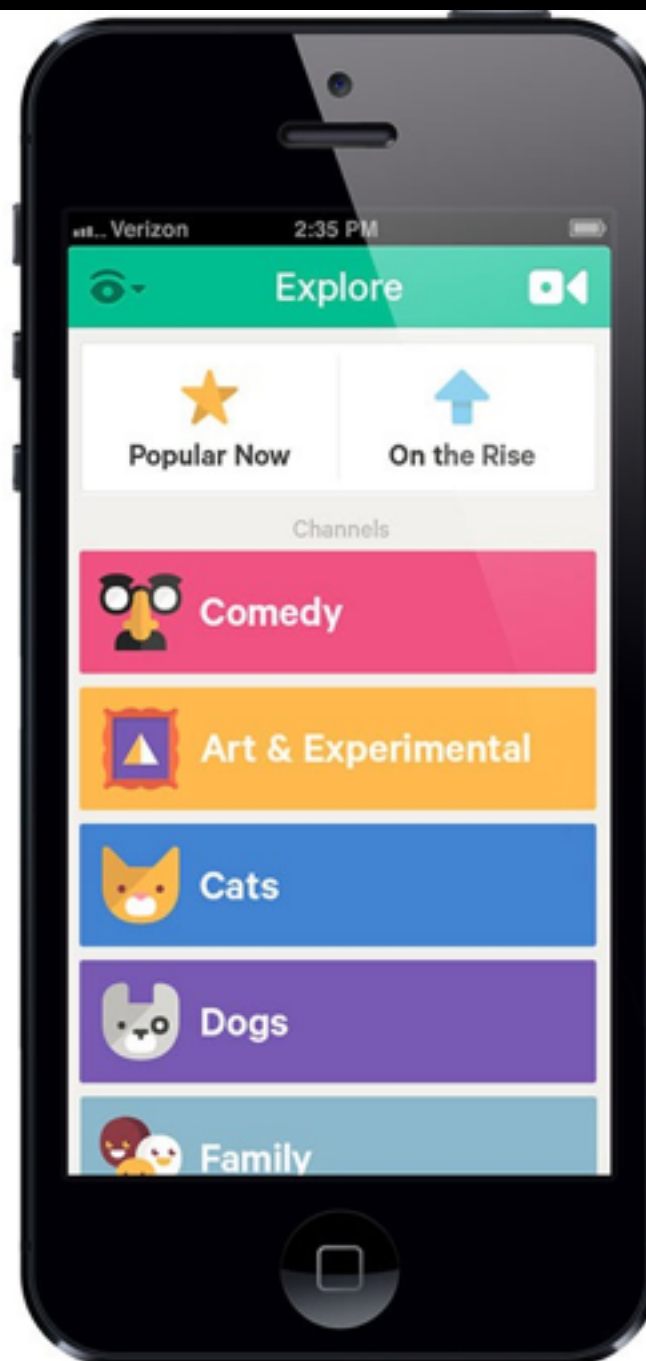


throw *new Error*

Ralph Holzmann · @rlph · jQuery Chicago 2014

Vine



Web Engineer

Vine

- Previously Twitter Ads, Bitovi JavaScript consulting
- Remote engineer from Fond du Lac, Wisconsin



Web Engineer

Vine

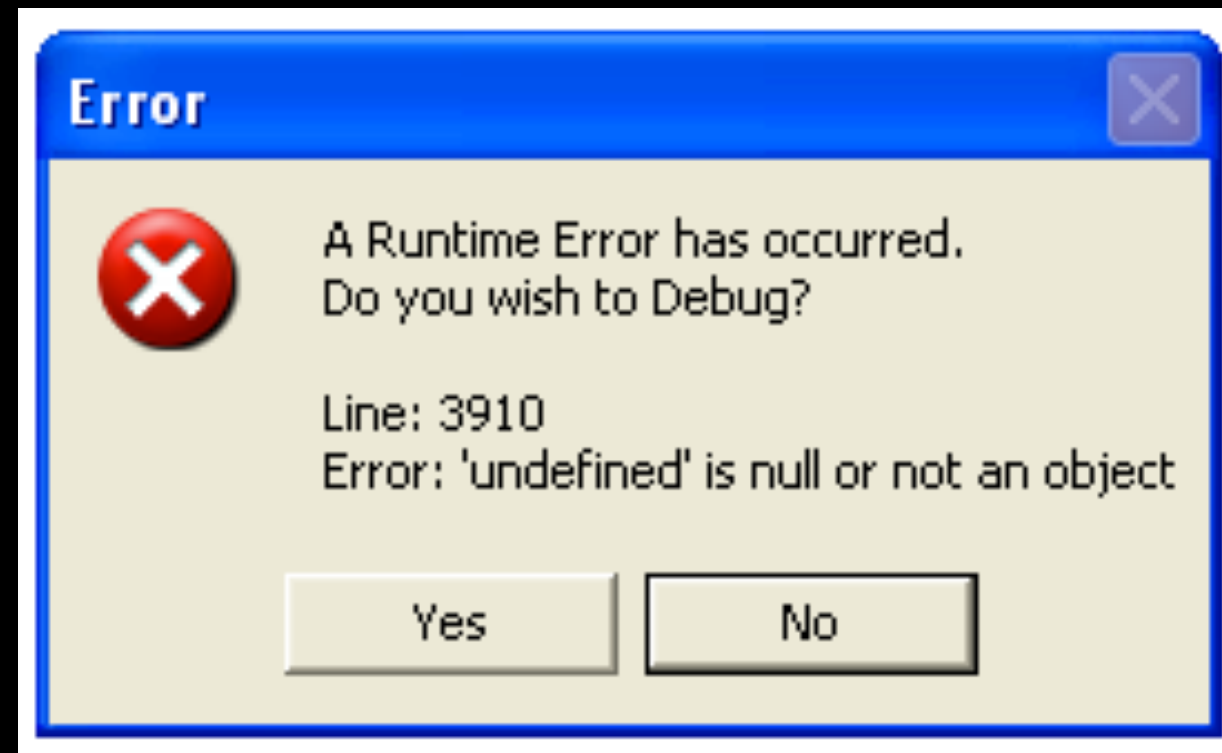
- Previously Twitter Ads, Bitovi JavaScript consulting
- Remote engineer from Fond du Lac, Wisconsin



Overview

- Classes of Errors
- Creating Errors
- Handling Errors
- When to throw

Classes of Errors



Syntax
Runtime
Logic

Synchronous

Syntax

Runtime

Logic

```
try {  
    syncOp()  
} catch (error) {  
    console.error(error)  
}
```

Asynchronous

Syntax

Runtime

Logic

```
asyncOp().catch(function(error) {  
    console.error(error);  
});  
  
$.ajax({  
    url: "foo/bar",  
    error: function(error) {  
        console.error(error)  
    }  
});
```

Syntax
Runtime
Logic

Runtime Logic

Runtime

Creating Errors

Always instantiate a new Error object

Never throw strings

```
throw "User not found."
```

Always throw errors

```
var error = new Error("User not found.")  
throw error;
```

Always instantiate a new Error object

Never throw strings

```
throw "User not found."
```

Always throw errors

```
var error = Error("User not found.")  
throw error;
```

Always instantiate a new Error object

Never throw strings

throw "User not found."

Always throw errors

throw Error("User not found.")

Always instantiate a new Error object

Never throw strings

throw "User not found."

Always throw errors

throw new Error("User not found.")

Error Constructor

```
new Error([message])
```


Error Constructor

```
new Error([message[, fileName[, lineNumber]])
```

* Non-standardized

Error Constructor

```
new Error([message[, fileName[, lineNumber]])
```

* Non-standardized

Error Subclasses

EvalError

SyntaxError

RangeError

TypeError

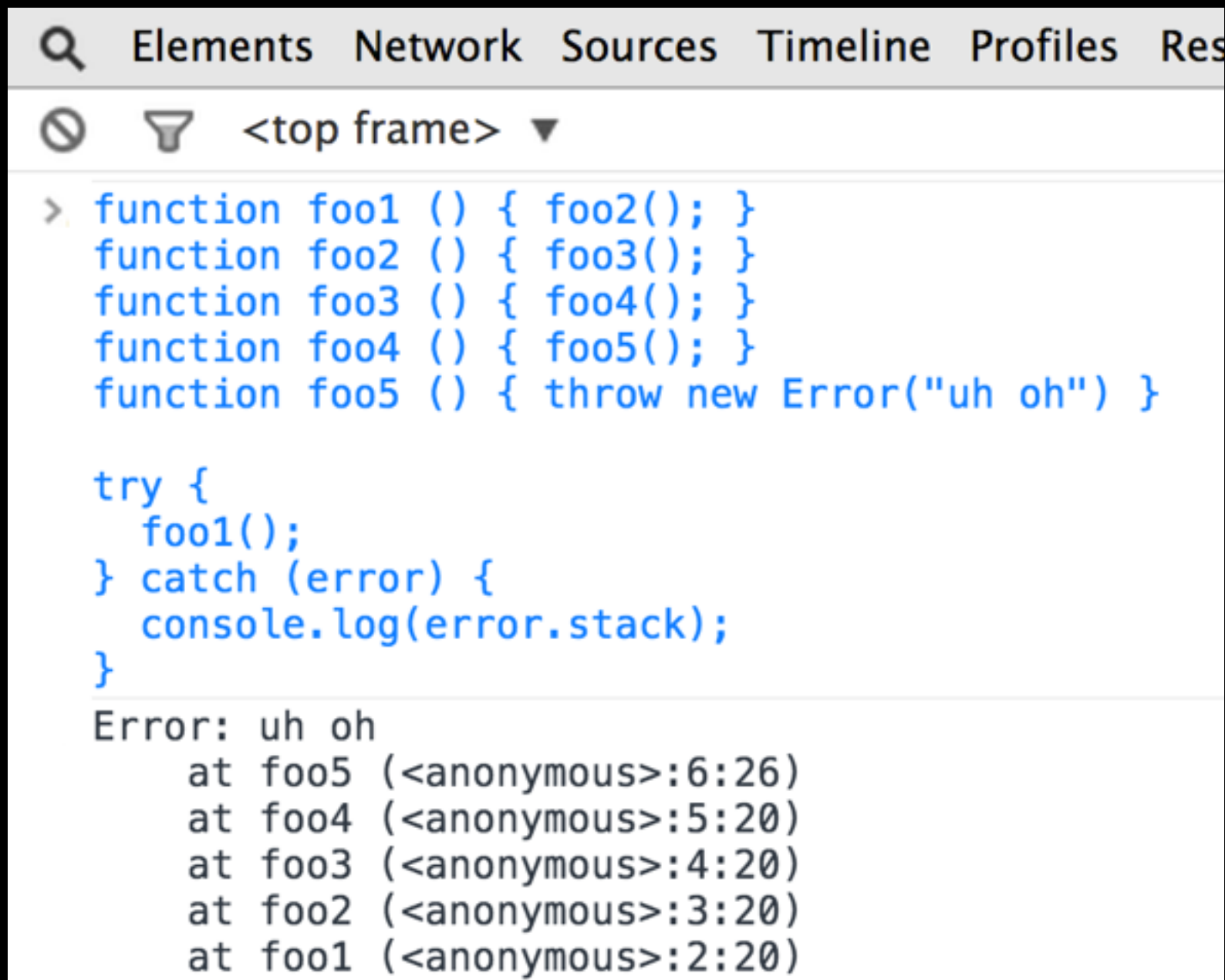
ReferenceError

URIError

InternalError

Error object

error.message and error.stack



The screenshot shows a web browser's developer console with the 'Sources' tab selected. The console displays a series of function definitions (foo1 through foo5) and a try-catch block. foo5 throws an Error with the message 'uh oh'. The error stack is visible below the code, showing the call sequence from foo1 to foo5. The stack trace is as follows:

```
> function foo1 () { foo2(); }  
function foo2 () { foo3(); }  
function foo3 () { foo4(); }  
function foo4 () { foo5(); }  
function foo5 () { throw new Error("uh oh") }  
  
try {  
  foo1();  
} catch (error) {  
  console.log(error.stack);  
}  
  
Error: uh oh  
    at foo5 (<anonymous>:6:26)  
    at foo4 (<anonymous>:5:20)  
    at foo3 (<anonymous>:4:20)  
    at foo2 (<anonymous>:3:20)  
    at foo1 (<anonymous>:2:20)
```

Handling Errors

Standard Sync Errors

```
try {  
    var user = User.find({  
        id: 1  
    });  
} catch (error) {  
    handleError();  
}
```


Standard Sync Errors

```
try {  
    var user = User.find({  
        id: 1  
    });  
} catch (error) {  
    if (error instanceof TypeError) {  
        handleTypeError();  
    } else if (error instanceof RangeError) {  
        handleRangeError();  
    }  
}
```

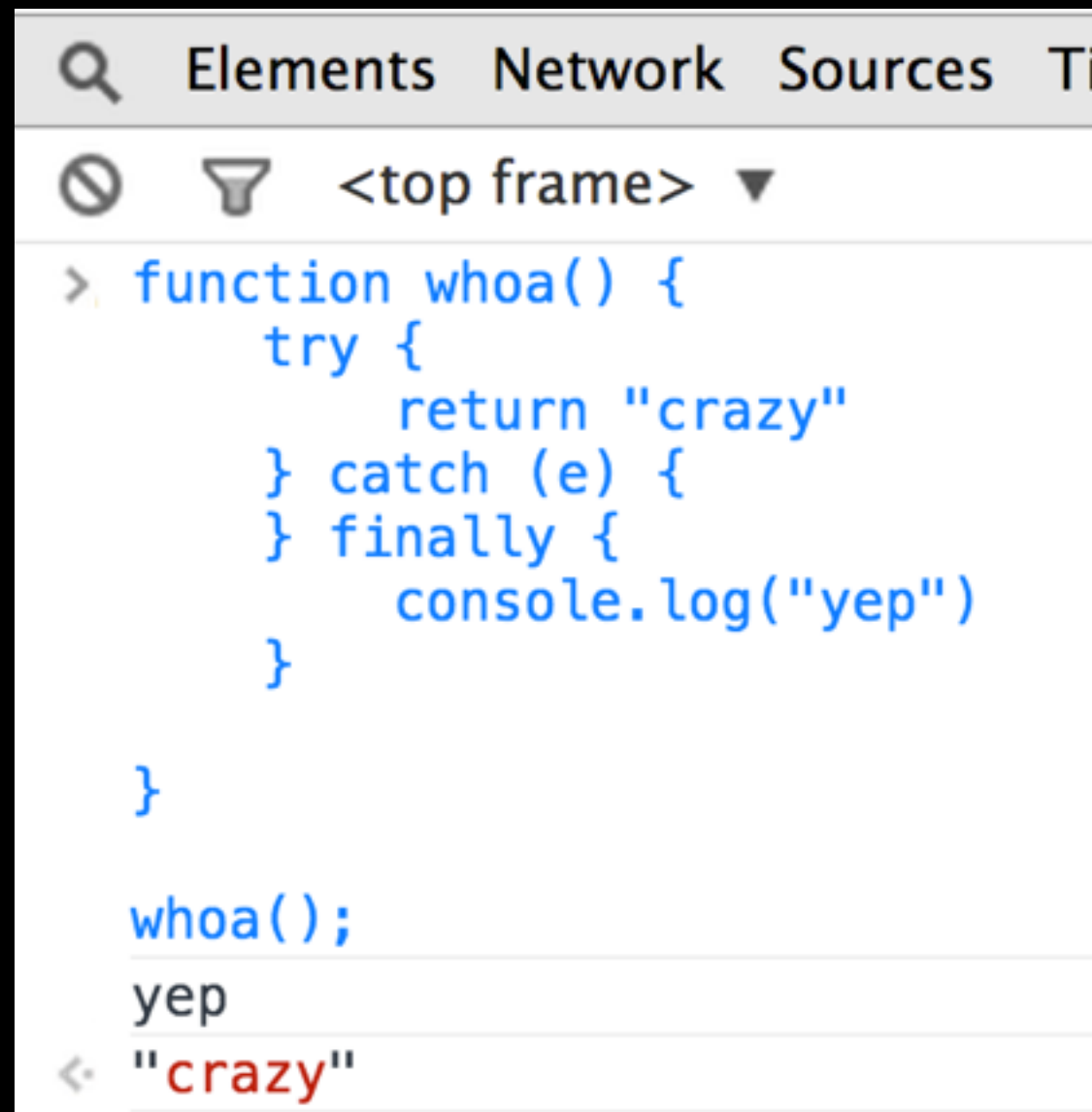
Standard Sync Errors

```
try {  
    var user = User.find({  
        id: 1  
    });  
} catch (error) {  
    if (error instanceof TypeError) {  
        handleTypeError();  
    } else if (error instanceof RangeError) {  
        handleRangeError();  
    }  
}  
} finally {  
    whateverKeepGoing();  
}
```

Aside: finally

```
function whoa() {  
  try {  
    return "crazy"  
  } catch (e) {  
  
  } finally {  
    console.log("yep")  
  }  
}  
  
whoa();
```

Aside: finally



```
Elements Network Sources Timeline  
<top frame>  
> function whoa() {  
    try {  
        return "crazy"  
    } catch (e) {  
    } finally {  
        console.log("yep")  
    }  
}  
  
whoa();  
yep  
"crazy"
```

Custom Errors

```
function ExpressError(message, status) {  
  this.name = "ExpressError";  
  this.message = message || "Default Message";  
  this.status = status || 500;  
}  
ExpressError.prototype = new Error();  
ExpressError.prototype.constructor = ExpressError;
```

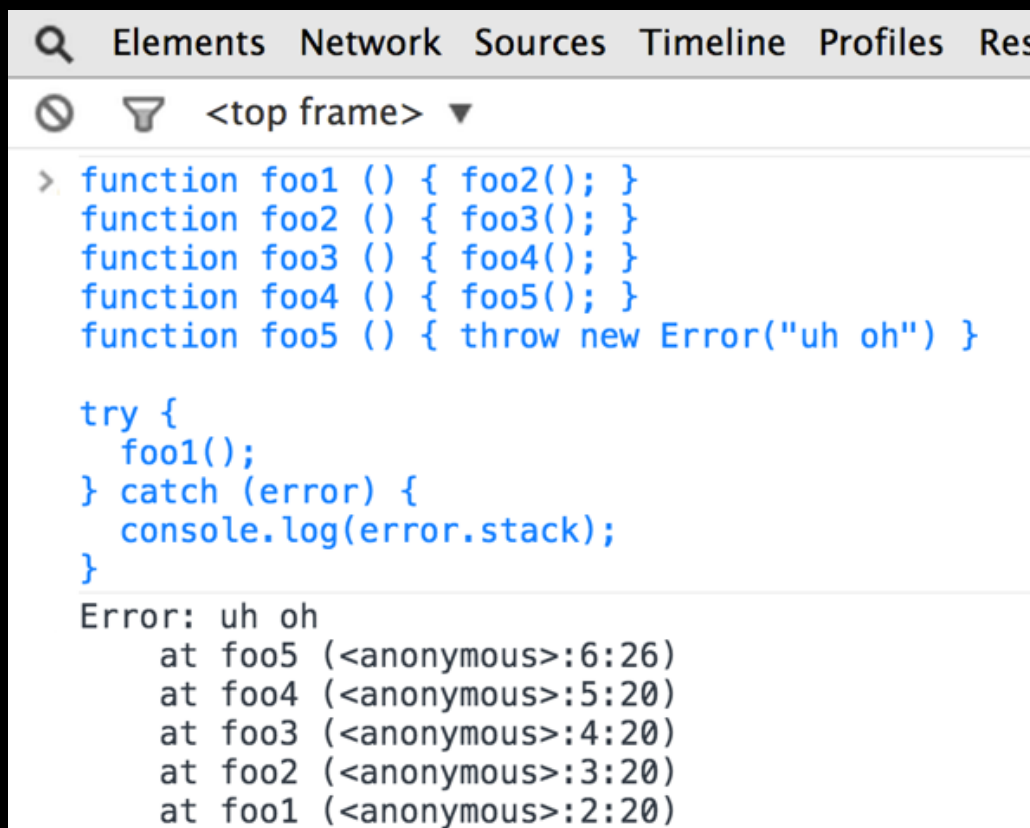

Custom Errors

```
function ExpressError(message, status) {  
  this.name = "ExpressError";  
  this.message = message || "Default Message";  
  this.status = status || 500;  
}  
ExpressError.prototype = new Error();  
ExpressError.prototype.constructor = ExpressError;  
  
app.get("/users/:id", function(req, res, next) {  
  User.findOne(req.params.id).catch(function() {  
    next(new ExpressError("User not found", 404));  
  });  
});
```

Async Errors

Tough track because you lose your call stack

Sync

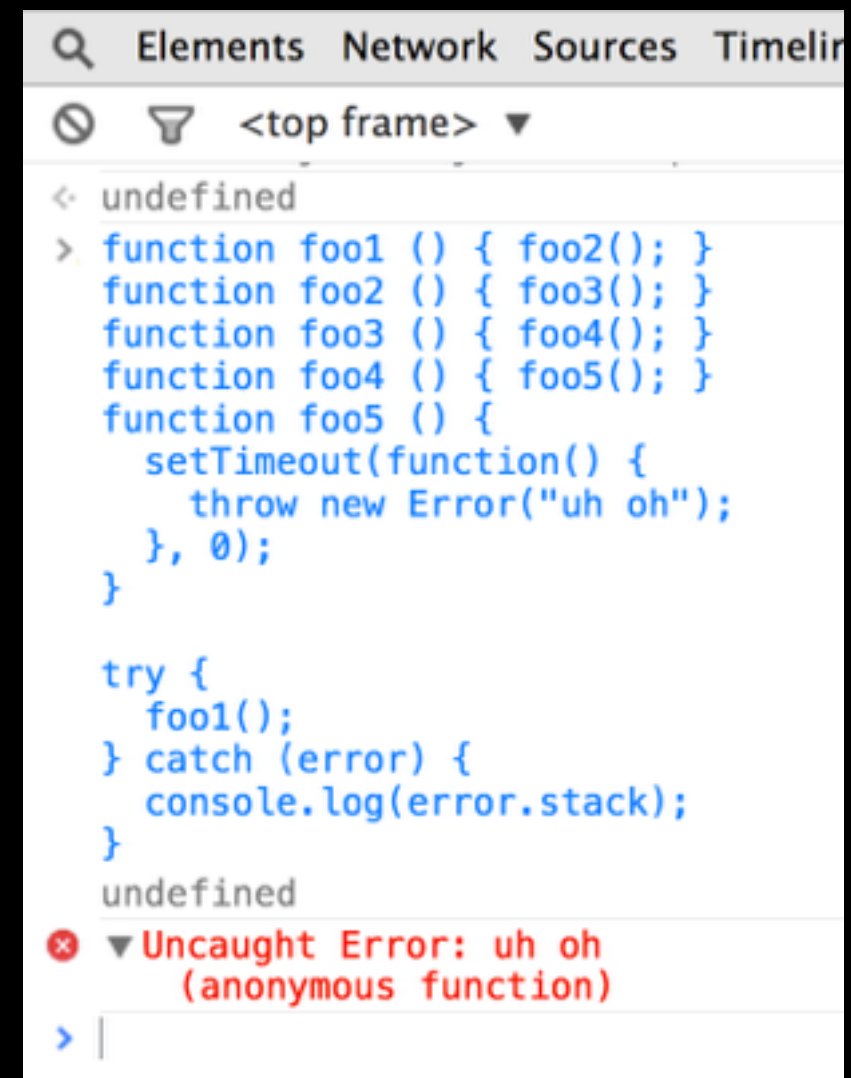


```
function foo1 () { foo2(); }
function foo2 () { foo3(); }
function foo3 () { foo4(); }
function foo4 () { foo5(); }
function foo5 () { throw new Error("uh oh") }

try {
  foo1();
} catch (error) {
  console.log(error.stack);
}
```

Error: uh oh
at foo5 (<anonymous>:6:26)
at foo4 (<anonymous>:5:20)
at foo3 (<anonymous>:4:20)
at foo2 (<anonymous>:3:20)
at foo1 (<anonymous>:2:20)

Async



```
< undefined
> function foo1 () { foo2(); }
function foo2 () { foo3(); }
function foo3 () { foo4(); }
function foo4 () { foo5(); }
function foo5 () {
  setTimeout(function() {
    throw new Error("uh oh");
  }, 0);
}

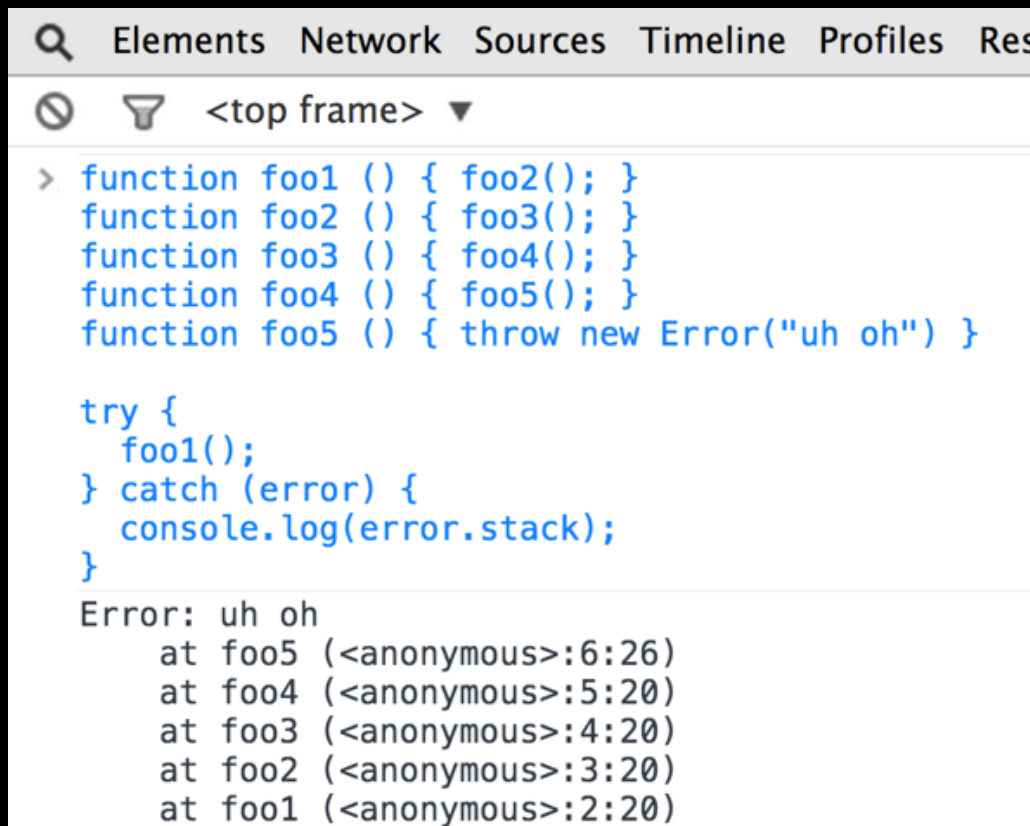
try {
  foo1();
} catch (error) {
  console.log(error.stack);
}

undefined
Uncaught Error: uh oh
(anonymous function)
```

Async Errors

Tough to catch

Sync

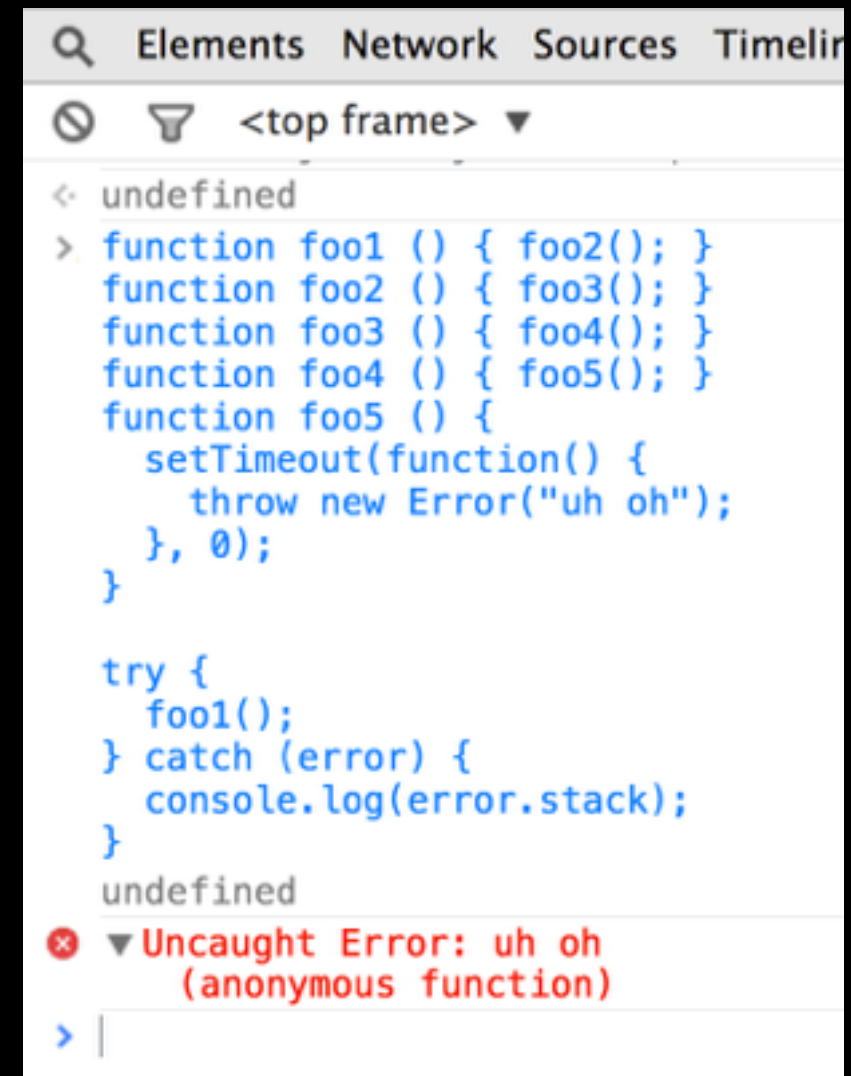


```
function foo1 () { foo2(); }
function foo2 () { foo3(); }
function foo3 () { foo4(); }
function foo4 () { foo5(); }
function foo5 () { throw new Error("uh oh") }

try {
  foo1();
} catch (error) {
  console.log(error.stack);
}
```

Error: uh oh
at foo5 (<anonymous>:6:26)
at foo4 (<anonymous>:5:20)
at foo3 (<anonymous>:4:20)
at foo2 (<anonymous>:3:20)
at foo1 (<anonymous>:2:20)

Async



```
< undefined
> function foo1 () { foo2(); }
function foo2 () { foo3(); }
function foo3 () { foo4(); }
function foo4 () { foo5(); }
function foo5 () {
  setTimeout(function() {
    throw new Error("uh oh");
  }, 0);
}

try {
  foo1();
} catch (error) {
  console.log(error.stack);
}

undefined
✖ Uncaught Error: uh oh
  (anonymous function)
```

Promises/A+



then

```
new Promise(function(resolve, reject) {  
  return User.findOne(1);  
});
```

Promises/A+



then

```
new Promise(function(resolve, reject) {  
  return User.findOne(1);  
}).then(function (user) {  
  
}, function (error) {  
  
});
```

Promises/A+



then

```
new Promise(function(resolve, reject) {  
  return User.findOne(1);  
}).then(function (user) {  
  
}, function (error) {  
  
});
```

Promises/A+



then

```
new Promise(function(resolve, reject) {  
  return User.findOne(1);  
}).then(function (user) {  
  throw new Error("Suspended user");  
}, function (error) {  
  // Haha nice try  
});
```

Uncaught exception

Promises/A+



then

```
new Promise(function(resolve, reject) {  
  return User.findOne(1);  
}).then(function (user) {  
  throw new Error("Suspended user");  
}, function (error) {  
  // Haha nice try  
});
```

~~Uncaught~~ Swallowed exception

Promises/A+



then

```
new Promise(function(resolve, reject) {  
  return User.findOne(1);  
}).then(function (user) {  
  throw new Error("Suspended user");  
}, function (error) {  
  // Handle user not found error  
}).then(null, function(error) {  
  // Handle suspended user error  
});
```

Promises/A+



then

```
new Promise(function(resolve, reject) {  
  return User.findOne(1);  
}).then(function (user) {  
  throw new Error("Suspended user");  
}, function (error) {  
  // Handle user not found error  
}).catch(function(error) {  
  // Handle suspended user error  
});
```

When to throw Errors

Be liberal

Throw whenever something unexpected happens

Unexpected Arguments

```
function ajax(options) {  
    if (typeof options !== "object") {  
        throw new Error("ajax requires options object")  
    }  
}
```

Unexpected Code Paths

```
switch (type) {  
    case "apples":  
        handleApples();  
        break;  
    case "oranges":  
        handleOranges();  
        break;  
    default:  
        throw new Error("Unexpected fruit.");  
}
```

Wrong Number of Args

```
function swap(a, b) {  
  if (arguments.length !== 2) {  
    throw new Error("swap function expects exactly 2 args.")  
  }  
}
```

Runtime assertions

```
function assert(message, test) {  
    if (!test) {  
        throw new Error(message);  
    }  
}
```


Runtime assertions

```
function ajax(options) {  
  if (typeof options !== "object") {  
    throw new Error("ajax requires options object")  
  }  
}
```

Runtime assertions

```
function ajax(options) {  
  assert("ajax requires options object", typeof options !== "object")  
}
```

Runtime assertions

```
function swap(a, b) {  
  if (arguments.length !== 2) {  
    throw new Error("swap function expects exactly 2 args.")  
  }  
}
```

Runtime assertions

```
function swap(a, b) {  
  assert("swap function expects exactly 2 args.", arguments.length !== 2)  
}
```

Additional Reading

- Mozilla Developer Network Error
- V8 Stack Trace API
 - `Error.stackTraceLimit`
 - `Error.captureStackTrace`

That's it.

Reach out!

- ralph@holzmann.io
- @rlph on Twitter
- vine.co/ralph
- ralphholzmann on freenode IRC
- Thanks!