Promises in JavaScript

George Adams, IV

What are Monads?

Monad = Design Pattern

Monad Laws

A Function

There must be a function to take a simple value and return a promise.

```
var myPromise = Q(7)
```

Binding

There must be a way to work with the value of the Promise.

```
function doubleIt (value) {
    return Q(value * 2)
}

myPromise
    .then(doubleIt)
```

Chainable

Promises must be chainable.

```
myPromise
    .then(doubleIt)
    .then(function (value) {
        console.log(value)
    })
```

JavaScript is Not Typed

Promises must be a little forgiving.

```
function doubleIt (value) {
    return Q(value * 2)
}

function doubleItPrime (value) {
    return value * 2
}
```

Left Identity

```
Q(7).then(doubleIt) == doubleIt(7)
```

Right Identity

```
myPromise.then(Q) == myPromise
```

Associativity

```
myPromise.then(foo).then(bar) ==
myPromise.then(function (value) { foo(value).then(bar) })
```

Usage

Why is this Helpful?

- Deferment
- Immutable data
- Repeatability

Deferment

`foo` executes asynchronously when `http()` returns.

```
http()
    .then(foo)

bar()
```

Immutable Data

`myPromise` can't be changed once it's data has been set.

Repeatability

```
myPromise
    .then(foo)

myPromise
    .then(bar)
```

The Real World

…is full of failure.

Two Paths

```
function happy (value) {
    return Q(value * 2)
}

function sad (reason) {
    // default to 7
    return Q(7)
}

http()
    .then(happy, sad)
    .then(function (value) {
        console.log(value)
    })
```

Path Traversal

```
http()
    .then(function (value) { throw "I can't handle a " + value + "!" })
    .then(
        function (value) { /* does not execute */ }),
        function (reason) {
            console.log(reason)
            return 10
        }
    )
    .then(
        function (value) { return value * 2 },
        function (reason) { /* does not execute */ }
    )
    .then(function (value) { console.log(value) })
```

Further Reading

- Either
- Q
- jQuery Deferred