

Who Am I Anyway?

- Currently developing e-commerce applications as a software engineer at Niche Retail, LLC.
- Ten years of professional experience, primarily on the client-side.
- Focusing on Java, PHP, MySQL, JavaScript and some Ruby on Rails.
- Authoring articles for JavaScriptAnt.com.
- Blogging sporadically at blog.reindel.com.

The Fringe Benefits of Libraries

- Innovators bring to light new techniques and design patterns through libraries, which helps to advance other developers.
- These techniques are “institutionalized” when developers use a library.
- Libraries help bring designers into the technology fold, which might otherwise be difficult to do.

The Namespace

```
// The namespace is called a package in Java.
```

```
import java.util.Map;
```

```
/*
```

```
    A namespace is:
```

- + An organizational structure.
- + One way to manage scope for identical names.
- + One way to maintain access control.

```
*/
```

```
YAHOO.util.Easing.easeOut; // From the YUI namespace.
```

The Namespace

```
// Same language; different dialect.
```

```
var BJR = new Object();  
var BJR = {};  
var BJR = new function(){};  
var BJR = Function();  
var BJR = function(){}();
```

```
// Not sure? Keep it simple...
```

The Namespace

```
// The object literal is easy to use.
```

```
var BJR = {  
    myProperty : true,  
    myMethod : {  
        var msg = "Hello World!";  
        alert( msg );  
    }  
};  
if ( BJR.myProperty ) {  
    BJR.myMethod(); // Alerts "Hello World!".  
}
```

The Namespace

```
// Why bother? Purpose. Scope.  
  
var myProperty = true; // I suck. I'm global.  
var BJR = {  
    myProperty : false, // I don't care.  
    myMethod : function() {  
        return this.myProperty; // Neither do I.  
    }  
};  
  
alert( myProperty ); // Alerts true.  
alert( BJR.myProperty ); // Alerts false.
```

The Namespace

```
// Enhanced scope in the namespace.

var BJR = function() {
    var myPrivateProperty = true;
    return {
        myPublicMethod : function() {
            return myPrivateProperty; // Closure.
        }
    };
}(); // I'm self-invoked.

alert( BJR.myPublicMethod() ); // Alerts true.
```

The Namespace

```
var BJR = function( el ) {
    this.el = el;
    this.myMethod = function() {
        return this; // I can be chained.
    };
    if ( this instanceof BJR ) {
        return this.BJR;
    } else {
        return new BJR();
    }
};

BJR( "myEl" ).myMethod(); // Namespace selector.
```

Constructor Type-Checking

```
function getObjectType( obj ) {  
    return obj.constructor.name;  
} // No more nebulous [ object Object ].
```

```
window.onload = function() {  
    alert( getObjectType( "Hello World!" ) );  
    function Cat() {}  
    alert( getObjectType( new Cat() ) );  
} // Array, Function, String, Number, Boolean, [ object ]
```

Variable Arguments

```
// Arguments as an array... but not really.
```

```
function myFunction() {  
    var args = >>>  
        Array.prototype.slice.call( arguments );  
    for ( i = 0; i < args.length; i++ ) {  
        alert( args[ i ] );  
    }  
}  
window.onload = function() {  
    myFunction( "Hello World!", >>>  
        "Howdy World!", "Hey World!" );  
}
```

The Callback Function

```
function myFunction( args, callback ) {
    for ( i = 0; i < args.length; i++ ) {
        alert( args[ i ] );
    }
    return new callback;
}
window.onload = function() {
    myFunction( { "Hello World!", "World?" }, >>>
        function() { // I'm an anonymous constructor.
            alert( "Goodbye World!" );
        }
    );
}
```