Integrating client side JavaScript with server-side business procedures is quite straightforward.

This is accomplished by defining the libraries, functions and scripting language for the functions in an exports.conf file which can reside in the app and/or the library directory. Example export.conf file:

This is an example exports.conf file # # Each line describes a function that will be available directly from JavaScript code in the client # # library, function, type (source library, name of the function, language vfp or javascript) # (php and python will be added later) # # or # # library, function (implied type of Lianja/vfp) # # or # # function (file expected to exist with a .dbo extension) # mylib,myfunc1,vfp myjslib,myfunc2,javascript myfunc3,vfp myfunc4

Now in your App logic running in the Lianja JavaScript Web/Mobile App you can call the server-side functions directly:

// Note that arguments are automatically converted into the correct format for the // remote function call and the function call is proxied // using a synchronous Lianja.evaluate() call var result = myfunc1("hello world", 2015, true); var result2 = myfunc2(); var result3 = myfunc3();
// When calling remote functions with objects as parameters the object is automatically // converted to JSON and sent base64 encoded.
// // This allows you to send and receive complete JSON encoded objects between the // client and the server. In the server-side myfunc4 function all you need to do is base64_decode()
// the parameter then json_decode() it and then you have a server side object that has the same
// members as the client side object.
//
// The return value from the myfunc4 function could be a simple data type or a JSON encoded object that you return
// using json_encode() var result4 = myfunc4({"name":"barry", "company":"Lianja" });