Profound.js Converter Benefits

The following is a list of the benefits offered by the Profound.js Converter.

Ability to hire and use modern developers to work on your applications

Translating RPG into JavaScript allows organizations to tap into an ecosystem of millions of developers.

JavaScript is a simple, performant, portable, and powerful languages that's everywhere these days. While JavaScript (along with Node.js) is still growing significantly, the technology has already surpassed all other languages in terms of usage, popularity, number of open source projects, and developers. Compared to traditional IBM i languages, such as RPG and CL, JavaScript offers more examples, education, tools, and open source resources.

And, regardless of any existing server-side technologies, JavaScript is already the defacto high level language for all client-side user interface development. Therefore, all developers today must know JavaScript, which is yet another reason why choosing JavaScript for both server-side and client-side development is more efficient compared to using other languages on the server-side.

Tapping into any open source node package to extend your code

Once a program is converted, it can immediately start utilizing open source packages provided by NPM (Node Package Manager), a vast repository developed and maintained by the Node.js community. There are hundreds of thousands of packages on NPM available, and billions of packages are downloaded from NPM every single week.

Much of the functionality found on NPM provides capabilities that would traditionally cost thousands of dollars in software licensing or would entail time-consuming custom development. But with Profound.js, these capabilities become available to you without additional cost or effort. Here are just a few examples of what you will find:

- PDF, Excel, Image generation and processing
- Email
- Web Services Capabilities
- IoT, AI, BlueMix, Watson API, Cloud, Cognitive
- Encryption / Security
- JSON, XML processing, Web, Mobile Tools
- Integration: drivers and API to access data on other systems

Making your applications SOA accessible

Converted programs are automatically SOA accessible and can be accessed as a REST web service that communicates using the JSON protocol.

This is the case even when a program is not well modularized with the user interface logic neatly separated from business logic, a typical requirement for virtually all other SOA efforts.

With Profound.js, program logic becomes easily accessible because the code no longer runs directly within an interactive IBM i job and sends information to and from a screen as JSON data. The data can now be accessed both programmatically via web services and by a live user interacting with the application.

Cleaning up dead code

Over time, it is typical for an application to accumulate unused code (routines or definitions in the program that are no longer utilized). However, normally, without a lengthy manual analysis, you just don't know which sections of the code are used and which ones are not. The Profound.js Converter has the capability to automatically find those sections and eliminate them, giving you less lines of code to work with and therefore simplifying future development.

Automatically modularizing code

The converter can modularize your code into separate functions with isolated scopes. This simplifies maintenance by allowing developers to work with code in smaller units.

In a typical large monolithic RPG program, all subroutines have global scope. This means that when you introduce new variables and business logic, they can inadvertently affect data and logic in other routines within a big program. This makes development and testing very difficult.

The Profound.js Converter will transform these subroutines into more independent functions, allowing developers to declare local variables and introduce logic that is scoped only to the current routine, thus avoiding any unintended consequences outside of that routine.

Code becoming more self-explanatory

Not only do converted programs use JavaScript, a language that is familiar to most developers in the world today, but they also become more self-explanatory. The statements in the new code are more verbose and there are less abbreviations. Obscure abbreviated operations like “SETGT” or similar become self-explanatory method names like “positionAfter()”.

Adding object-oriented capabilities

The converter automatically introduces more mainstream object-oriented patterns into your traditional purely procedural code. For example, objects with properties and methods are created to represent the user interface. The same goes for database access – each database table becomes and object accessed by methods.

Making it easy for RPG developers to transition to a new language

Profound.js creates code that is easy to understand for both experienced Node.js developers and RPG developers that are new to Node.js. The simple transition to Node.js is made possible by the easy to use Profound.js API that provide virtually all capabilities of RPG, including:

- Record Level Access for database tables
- Ability to working with display files concepts, such as record formats and subfiles
- Top-down transactional programming without any requirement to use JavaScript callbacks
- All RPG data types, including complex data structures
- Data manipulation API for common RPG operations
- Integration API to seamlessly call existing RPG programs, CL programs, and IBM i commands

Portability

Node.js is inherently portable, which opens up exciting new possibilities for your applications, such as offering offline mobile capabilities to your users or being able to do development without a connection to the server. A converted program will generally still require an active connection to IBM i at runtime in order to access your IBM i database tables and other IBM i resources. However, the Node.js code itself can be tested and executed on IBM i or other types of servers, including directly on your development PC. This provides flexibility in how you can work with the programs and what tools you can use for debugging and testing.

Full integration with existing IBM i resources

Converting RPG programs to Node.js opens them up to many new capabilities, including direct usage of any open source NPM component.

Fortunately, Profound.js does this in a way that doesn’t take away existing capabilities of RPG and IBM i. RPG programs are known for tight integration with the DB2 database, other ILE programs on IBM i, and IBM i resources like data areas. The new Node.js code has all of the same IBM i integration capabilities. In fact, the converted Node.js module can run just like any other ILE program on IBM i, and has access to API for tasks like:

- DB2 Record Level Access and SQL operations, including support for overrides and QTEMP in an interactive job
- Calling other RPG, CL, and COBOL programs seamlessly
- Ability for Node.js modules to act like native ILE programs on IBM i, with full support for traditional object authority settings
- Calling 5250 green-screen interactive commands from Node, and having them render as HTML5 on-the-fly
- Ability for Node.js to be called from RPG, CL, or the command line using a standard IBM i CALL operation or command
- Support for all native IBM i data types and seamless handling of parameters between native programs and Node.js
- Calling any service program procedure on IBM i from Node, including operating system API

Incremental Conversion & Agile Modernization

According to market surveys, the two main challenges faced by IBM i shops with RPG applications are outdated user interfaces and the inability to find RPG developers to work on their applications. Traditionally, companies have tried to address these concerns either by rewriting applications or migrating to a new software package. However, the effort involved in projects like this is often grossly underestimated and the projects are rarely successful. This is because a complete rewrite or a package migration cannot be done incrementally. It involves a “big bang” approach with drastic changes to the application architecture, organizational processes, and IT practices.

An incremental, agile approach, where one or a handful of programs can be modernized and deployed at a time, is a lot more manageable and always more successful.

The Profound.js Converter allows for incremental, agile modernization because all program dependencies are automatically preserved after the conversion is complete.

In an application, programs are deeply interconnected. A typical program will call a number of other programs and may also be called by a variety of programs. A manual rewrite of a program to a more open language would break these dependencies, and thus require that all dependent programs are migrated before this part of the application is functional. However, with Profound.js, applications can be converted in very small increments, allowing for an agile, iterative process.