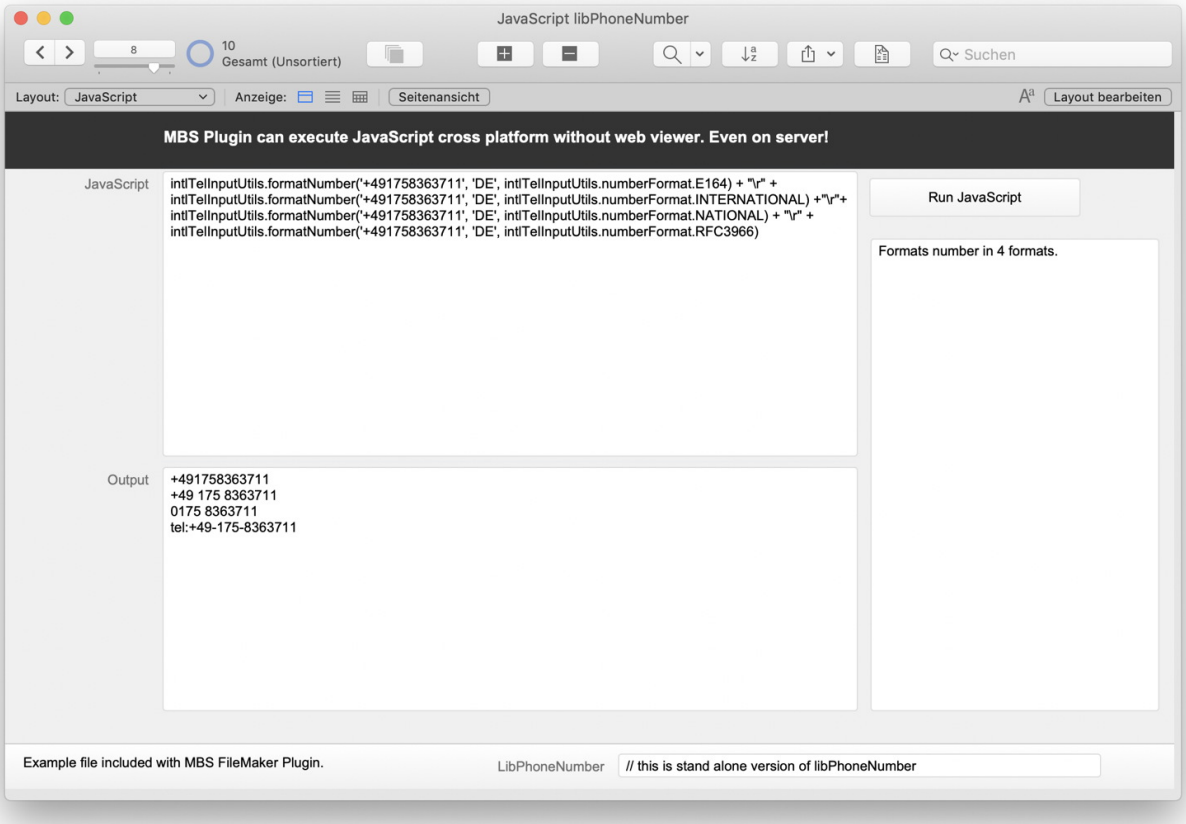
Using libPhoneNumber for phone number formatting

Recently a client asked whether we could integrate the libPhoneNumber library into the plugins. Well, it's a big C++ library with a lot of dependencies and it looks like building a static version for our plugin or a DLL version to load at runtime is a bit tricky. But we found the JavaScript port of the library on the same website. This one has a couple of dependencies, but it can be compiled into a single file version with all required libraries embedded in one JavaScript file.
As you may know we introduced a JavaScript engine into our plugins last December for both FileMaker. This JavaScript engine is capable to load the standalone version of libPhoneNumber and execute queries to work with phone numbers. Let us look at a few examples. First you load the library and initialize everything. Then you query for example the intlTelInputUtils.formatNumber function to format a number:

```
intlTelInputUtils.formatNumber('01751234567', 'DE',
intlTelInputUtils.numberFormat.INTERNATIONAL)
```

The result is "+49 175 1234567", which formats the number for international use with spaces in-between for humans. All javaScript functions used here are in the intlTelInputUtils namespace, so all function names have this as prefix. The formatNumber function itself takes three parameters with the number, the country code and the format. Possible formats are given as constants in JavaScript, so you can pass here E164, INTERNATIONAL, NATIONAL or RFC3966. The last one is the tel: URL for use in links on a website. For our test number, we would get "+491751234567" for E164, "0175 1234567" for NATIONAL and "tel:+49-175-1234567" for RFC3966.



Next we can use getNumberType to learn the types of a number:

intlTelInputUtils.getNumberType('+491758363711', 'DE')

This returns 1, which means mobile. If you try a landline number like "02632123456", you get back 0. In the table below we list the types for you.

Number	Name	Comment
0	Fixed Line	
1	Mobile	
2	Fixed or Mobile	In some regions (e.g. the USA), it is impossible to distinguish between fixed-line and mobile numbers by looking at the phone number itself.
3	Toll Free	
4	Premium Rate	
5	Shared Cost	The cost of this call is shared between the caller and the recipient, and is hence typically less than premium rate calls.

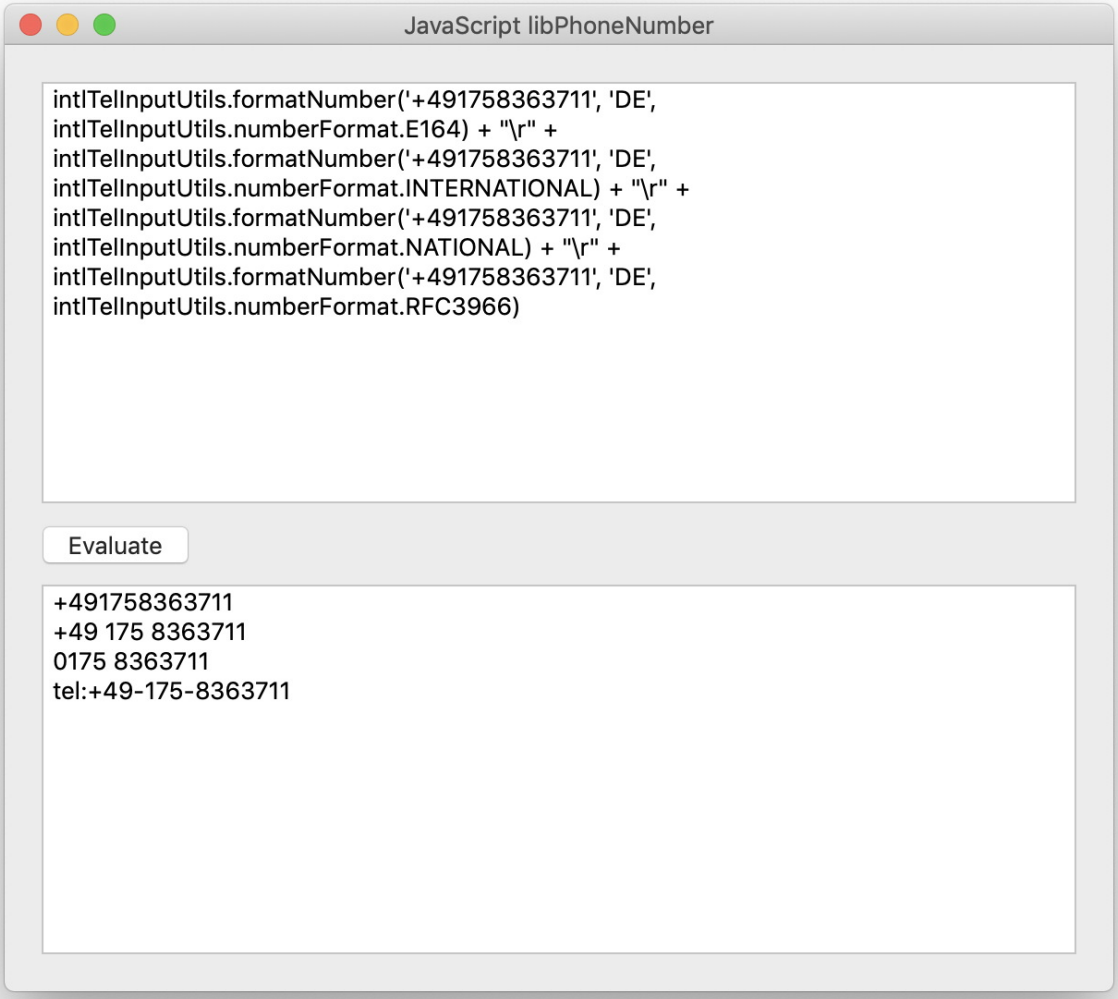
6	VOIP	Voice over IP numbers. This includes TSoIP (Telephony Service over IP).
7	Personal Number	A personal number is associated with a particular person, and may be routed to either a mobile or fixed line number.
8	Pager	
9	UAN	Used for 'Universal Access Numbers' or 'Company Numbers'. They may be further routed to specific offices, but allow one number to be used for a company.
10	VoiceMail	Used for 'Voice Mail Access Numbers'.
-1	Unknown	A phone number is of type unknown when it does not fit any of the known patterns for a specific region.

The function intlTelInputUtils.getExampleNumber can query you an example number to show in the user interface. The function takes the country code, whether to get national format and the number type you need. Let's query various types for the USA:

```
intlTelInputUtils.getExampleNumber('US', 0, 0) + " " +
intlTelInputUtils.getExampleNumber('US', 1, 0) + "\r" +
intlTelInputUtils.getExampleNumber('US', 0, 3) + " " +
intlTelInputUtils.getExampleNumber('US', 1, 3) + "\r" +
intlTelInputUtils.getExampleNumber('US', 0, 4) + " " +
intlTelInputUtils.getExampleNumber('US', 1, 4) + "\r" +
intlTelInputUtils.getExampleNumber('US', 0, 8) + " " +
intlTelInputUtils.getExampleNumber('US', 1, 8)
```

This returns:

```
+1 201-555-0123 (201) 555-0123+1 800-234-5678 (800) 234-5678+1 900-234-5678 (900) 234-5678
```



Next you may want to check intlTelInputUtils.isValidNumber to check if a number is valid and returns a boolean. For more details use intlTelInputUtils.getValidationError function to do the same, but get back the result as an integer. Value 0 means possible, 1 an invalid country code, 2 a too short number, 3 a too long number and 4 in case the given text is not a number.

As we run the JavaScript via our plugin without a browser, we can run this on a server without user interface. Works in FileMaker for client and server, even within a Runtime application.

The example for FileMaker will be included with next pre-release of our plugins. Please don't hesitate to contact us with questions or to get an early copy.