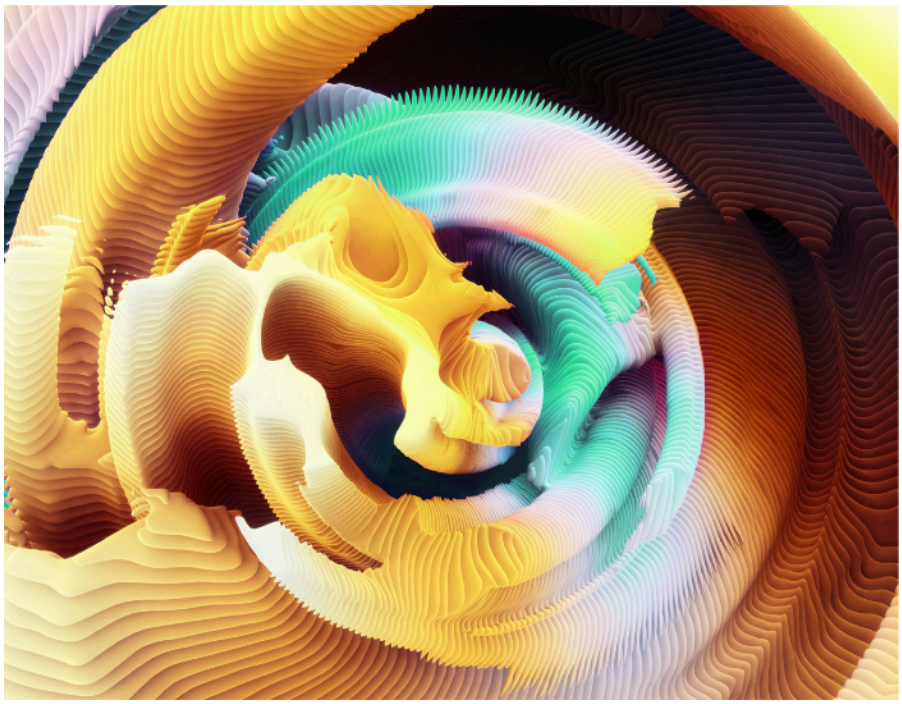
Claris

Claris FileMaker

JSON関数

事始め

- その1 -



SEP. 2020

VER. 1.1

Claris

目 次

はじめに

このドキュメントは、2020 年 9 月実施の Web セミナー「はじめての JSON 関数 その 1」（Claris 法人営業本部）の内容をベースとしています。

本 Web セミナーの録画は次の Web サイトで視聴できます。

https://content.claris.com/ja_recorded-webinars

なお、本ドキュメントの続編である「JSON 関数事始め - その 2 -」も同様に、対応する Web セミナーをベースとしています。

「JSON 関数事始め その 1」と同「その 2」、そしてそれぞれに対応する Web セミナーを併せてご利用になって、Claris® FileMaker® プラットフォームにおける JSON 形式のデータおよび JSON 関数についての理解を深めてください。

JSONとは

JSONとは

JSON (JavaScript Object Notation) は、テキストベースのデータ交換用フォーマットです。

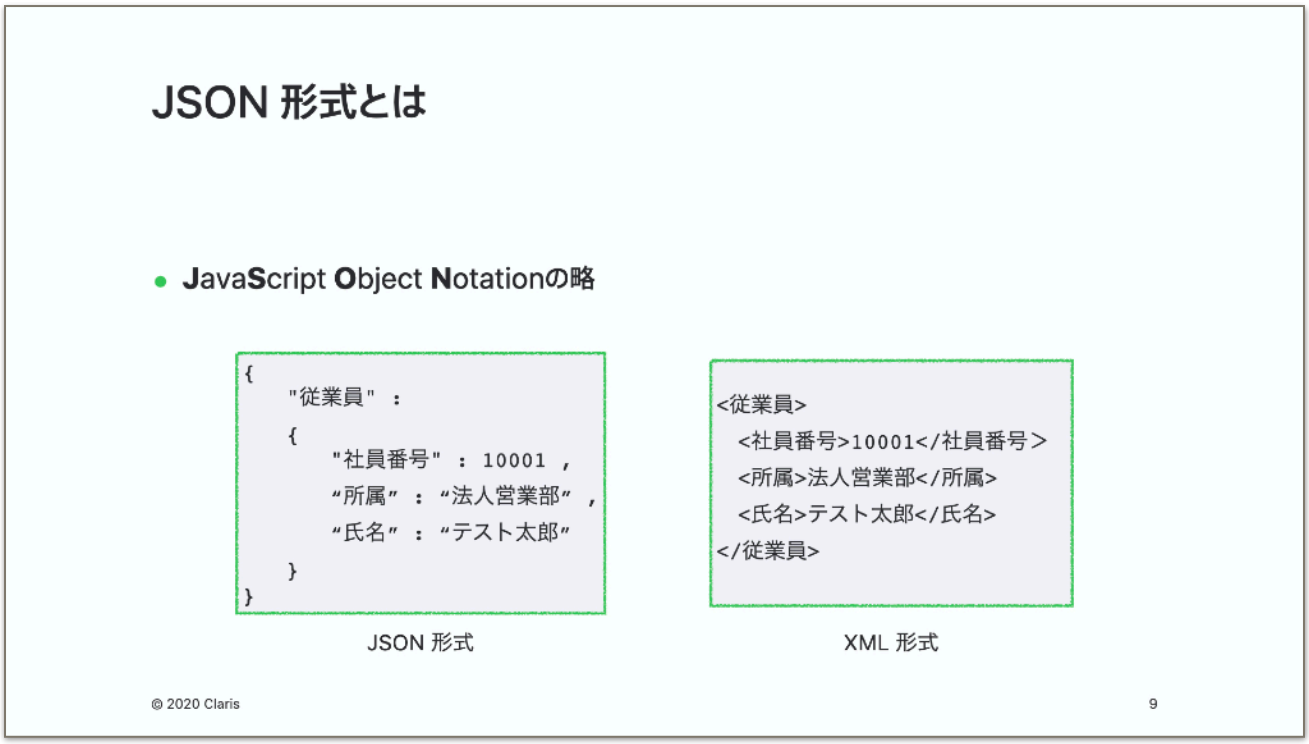
その名前から想像できるように、JavaScript におけるオブジェクト表記法の一部を使っていますが、JavaScript に限らず、様々な言語、および、FileMaker を含む様々なプラットフォームで利用されています。特に、Web API とのデータ交換においては JSON の利用がデファクトスタンダードとなっています。

JSON は、"{}"（中カッコ、波括弧）の中にキーと値をペアにしてデータを記述します。値の部分に JSON データを入れ子にして記述することもできます。シンプルな構造で様々なデータを扱うことができることから、人間が読んで理解しやすいだけでなく、コンピュータにおいても処理しやすいことが特徴です。

他のテキストベースのデータ交換用フォーマットとしては、例えば XML (Extensible Markup Language) があります。XML は、ユーザが定義したタグを用いて文章構造を記述するマークアップ言語です。XMLでは、開始タグと終了タグの間にテキスト（要素）を記述することによって特定の意味を持たせることができます。XML も JSON と同様にデータ交換の用途に利用することができますが、終了タグがあることによって、人間が読むには若干冗長で、また、データ量も JSON に比べて大きくなります。

XML は、どちらかというと、次の例のように、文章中の任意の場所にタグを追加して意味を追加する用途に向いています。ただし、XML ではスキーマ言語を使ってタグに関する厳密な構文規則を定義する必要があります。

```
<?xml version="1.0" encoding="UTF-8"?>
<製品名>FileMaker Pro</製品名> <バージョン release_data="2020/05/21">19</バージョン>には、次の<内容 sec="1">新機能</内容>と<内容 sec="2">改善点</内容>が含まれています。
```



FileMakerとJSON

FileMaker 純正の JSON 関数は FileMaker バージョン 16（2017）で登場しました。以来、REST API を利用した Web サービスとデータ交換する場合や、FileMaker スクリプトで複雑な構造のデータ（複数のパラメータ）を取り扱う場合などに活用され、その存在感を増してきています。

FileMaker における JSON 形式のデータの主な利用方法には、大きく分けて次の 3 つがあります。

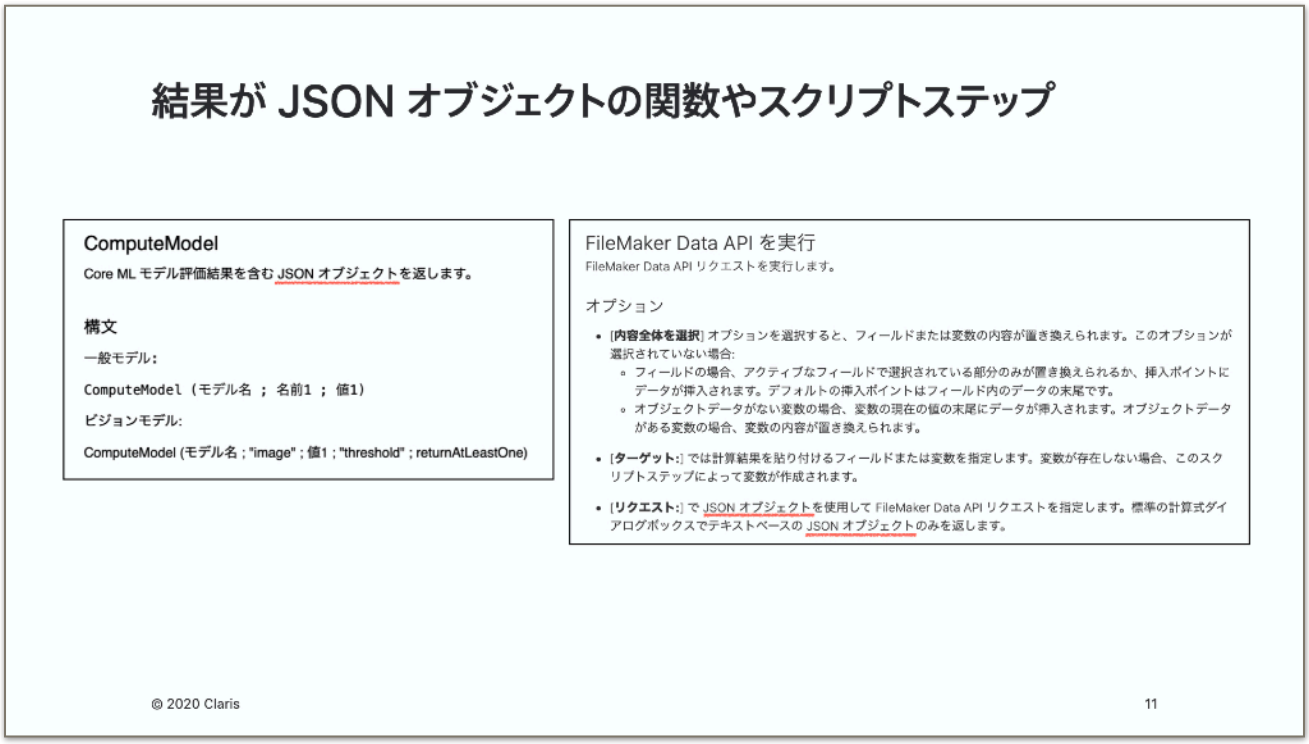
1) Web API とのデータ送受信用フォーマットとして使う
2) スクリプト引数として使う
3) 関数やスクリプトステップの戻り値として使う

1) の利用方法は、JSON 形式のデータフォーマットでデータを送受信する Web API とのデータ交換に使うというものです。昨今の Web API はデータ送受信に JSON 形式のフォーマットが使われることが多いので、JSON 形式を利用することによって、FileMaker から様々なサービスを提供する Web API を 利用することができます。本ドキュメントの「JSON 形式を利用する」セクションでも、郵便番号データサービスの Web API を使って郵便番号から住所を検索するスクリプトを紹介しています。

2) の利用方法は、FileMaker のスクリプトに複数の値を渡す場合に JSON 形式のデータを使うというものです。FileMaker のスクリプトが受け取るスクリプト引数は 1 つだけなので、改行区切りの文字列を使う方法の他、FileMaker ユーザの間でもこれまで様々な工夫がなされてきています

が、JSON 形式のデータを使うことによって、とてもシンプルかつ正確に複数の値を渡すことができます。この利用方法については、本編の続編である「JSON 関数事始め - その 2 -」で説明します。

3) の利用方法は、FileMaker の新しい関数やスクリプトステップから返されてくる JSON 形式のデータを使うというものです。例えば、バージョン 19 で追加された ComputerModel 関数や [FileMaker Data API を実行] スクリプトステップは、実行結果を JSON 形式のデータ（JSON オブジェクト）で返します。また、[FileMaker Data API を実行] スクリプトステップは、リクエストも JSON オブジェクトで指定します。



いずれの利用方法でも、JSON 形式のデータを利用することで、とても柔軟かつ便利に処理ができるようになります。

FileMakerのJSON関数

FileMaker で JSON を取り扱うための関数（JSON 関数）は全部で 6 個あります。関数の機能を「取得する」、「作る」、「削除する」そして「整形する」の 4 種類に分けると、「取得する」関数が 3 つ、その他の種類にそれぞれ 1 つずつ JSON 関数が用意されています。

これらの関数については、本編の後のセクションと、続編の「JSON 関数事始め - その 2 -」でご紹介します。

JSON 関数

	関数	返される値
取得する	JSONGetElement	JSON データで、オブジェクト名、配列索引、またはパスで指定された要素のクエリーを実行します。
取得する	JSONListKeys	オブジェクト名、配列索引、またはパスで指定された要素に対する JSON データ内のオブジェクト名 (キー) または配列索引の一覧を表示します。
取得する	JSONListValues	オブジェクト名、配列索引、またはパスで指定された要素に対する JSON データ内の値の一覧を表示します。
作る	JSONSetElement	オブジェクト名、配列索引、またはパスで指定された JSON データ内の要素を追加または変更します。
削除する	JSONDeleteElement	オブジェクト名、配列索引、またはパスで指定された JSON データ要素を削除します。
整形する	JSONFormatElements	JSON データ内の要素を読みやすい形に書式設定します。

それではまず、JSON とはなにか、についてみてみましょう。

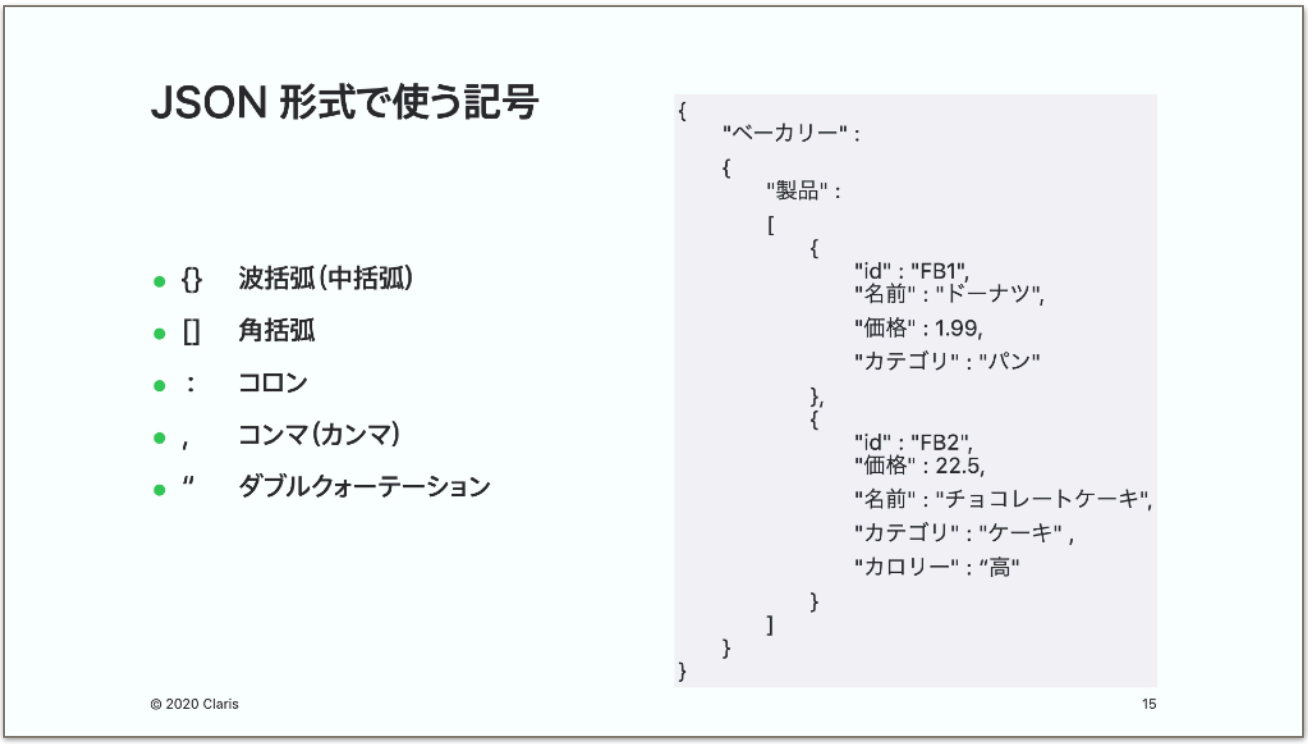
JSONの基本

FileMaker で JSON 形式のデータを扱う前に、まず、JSON の基本を押さえておきましょう。

JSON形式で使う記号

JSON では、次の 5 種類の記号を使用してデータを記述します。

- {} （中カッコ、波括弧） - 1 つ以上のキーと値のペアをくくります。
- [] （角カッコ、角括弧） - コンマで区切った値をくくって配列を表します。
- : （コロン） - キーと値を区切ってペア（要素）を作ります。
- , （コンマ） - 要素を区切ります。
- ” （ダブルクォーテーション） - 文字列をくくります。



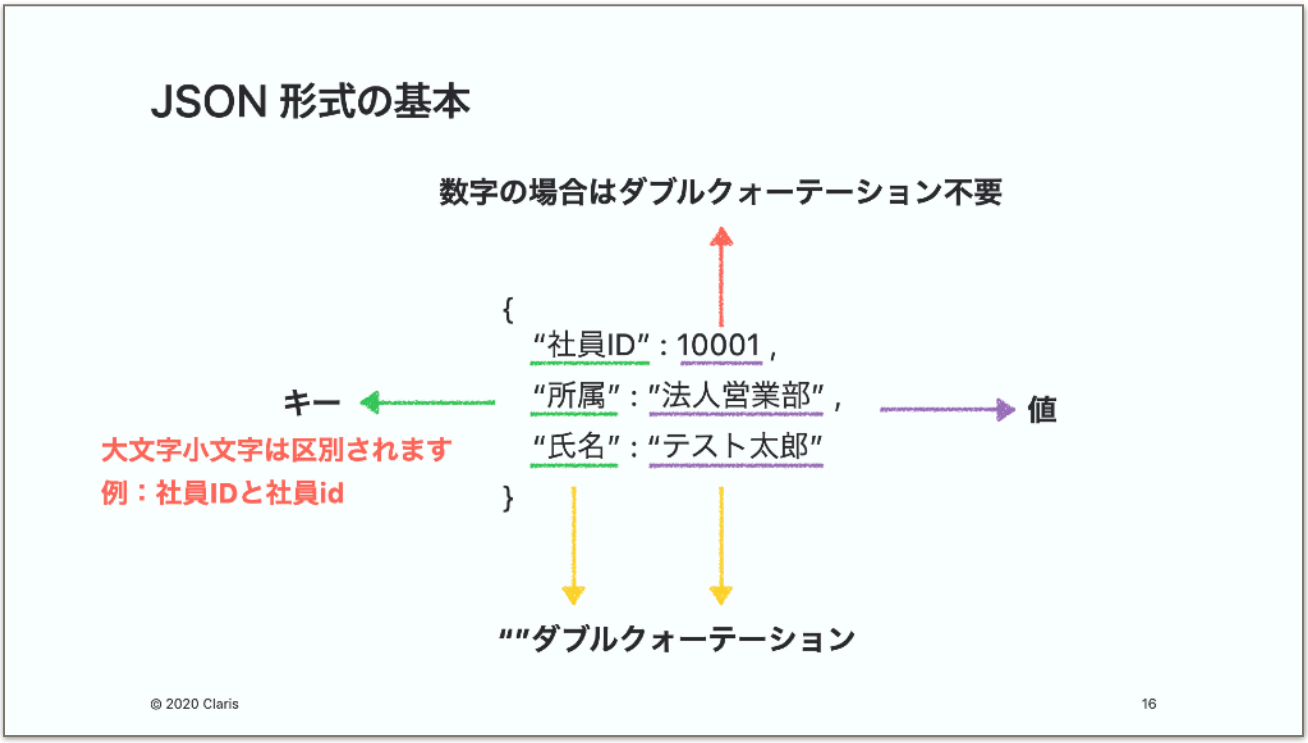
JSONの構造

JSON では、ひとくくりのデータを「オブジェクト」といいます。オブジェクトは、キーと値のペアをコロンで区切ったもの（要素）をゼロ個以上列挙し、中カッコでくくって表現します。

{ ”<キー1>” : <値1>, “<キー名2>” : <値2>, ...}

キーは文字列で、必ず「"」（ダブルクォーテーション）で囲む必要があります。なお、FileMaker ではアカウントや変数名など、大文字と小文字を区別しない場合がありますが、JSON では大文字と小文字は区別されますので注意してください。

値には、文字列の他、数字、オブジェクト、配列などを指定することができます。文字列以外の場合はダブルクォーテーションで囲む必要はありません。詳しくは、json.org を参照してください。



JSONのデータ型

JSON には次のような 6 種類のデータ型があります。

JSON のデータ型	説明
文字列	「”」（ダブルクォーテーション）でくくった文字列
数値	整数、浮動小数点数
オブジェクト	順序付けされていないキーと値のペアの集まり
配列	角括弧でくくられたデータのシーケンス
真偽値	真偽を表す値で、true と false（すべて小文字）
ヌル値	値がないことを示す表記で、null（すべて小文字）

ご覧のとおり、JSON のデータ型には FileMaker のフィールドタイプと対応しないものがあります。また、文字列や数値など、一見対応するように見えるものも厳密には FileMaker のフィールドタイプと仕様が異なることがあるので、JSON データから FileMaker にデータを取り込む場合や FileMaker のデータを JSON データに変換する場合には注意が必要です。これらについては、続編の「JSON 関数事始め - その 2 -」で説明します。

JSON形式を組み立てる

いよいよ、FileMaker で JSON 形式のデータを使ってみましょう。最初に、FileMaker の JSON 関数を使って JSON 形式のデータを作成してみます。

JSONSetElement 関数

FileMaker で JSON 形式のデータを作成、組み立てるには、JSONSetElement 関数を使用します。JSONSetElement 関数の構文は、次の形式です。

構文
JSONSetElement (json ; キーまたは索引またはパス ; 値 ; タイプ)

使用例

JSONSetElement (“{}” ; “布マスク” ; “2” ; JSONString)
JSONSetElement (JSON ; “在庫マスク[0].布マスク.価格” ; 350 ; JSONNumber)

引数

- 第 1 引数：json

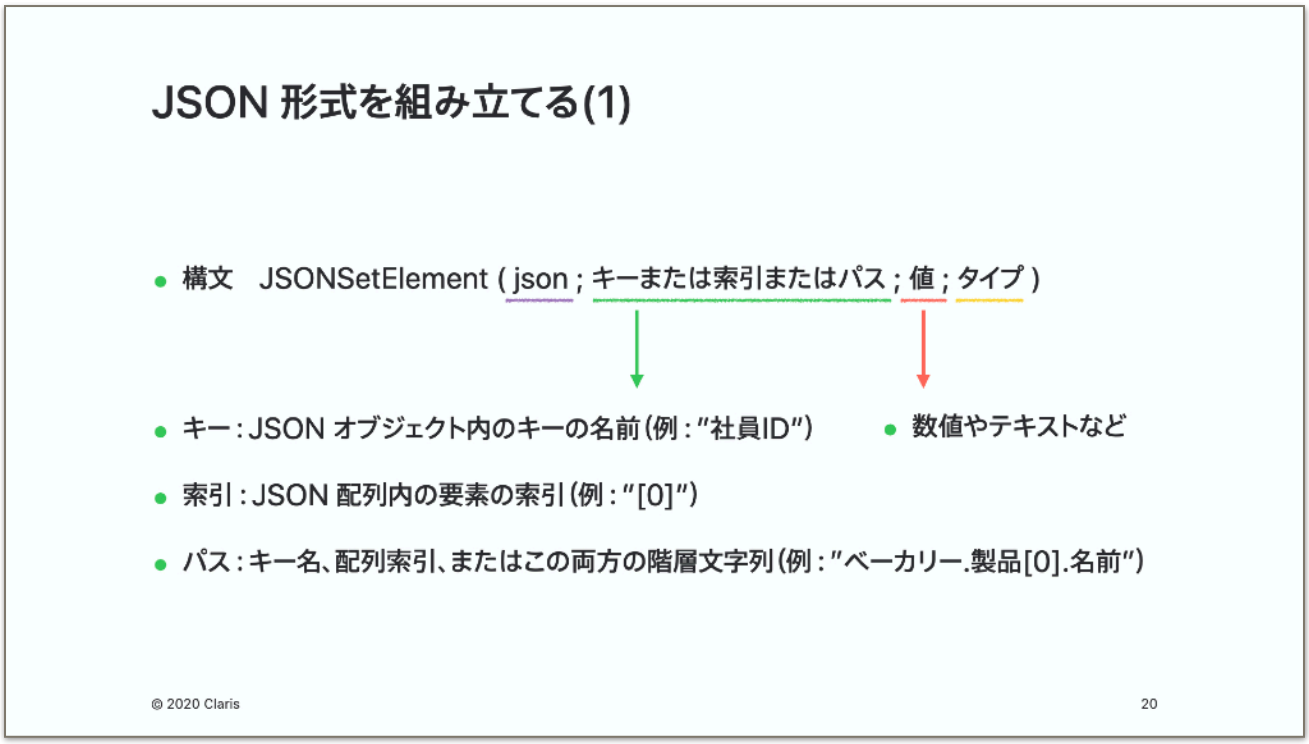
  JSON オブジェクトを指定します。初めて JSON 形式のデータを作成する場合は、“{}” または “”（空白）を指定します。なお、””（空白）を指定した場合は "{}" が自動的に補完されます。

- 第 2 引数：キーまたは索引またはパス

  キーは JSON オブジェクトの中のキー、索引は JSON 配列内の要素の索引、パスはキー名、配列要素、またはこの両方の階層文字列です。

- 第 3 引数：値

  数値、テキスト、または JSON データを含む任意の式またはフィールドを指定します。



- 第 4 引数：タイプ

  第 3 引数の値を JSON データに設定するときの JSON のデータ型を指定します。 次の表に示す 7 種類のうちのいずれかのタイプの名前（「JSONString」など）か、名前の後ろの括弧内の数値（「1」など）を使って指定することができます。表の「出力タイプ」は、対応する JSON のデータ型です。

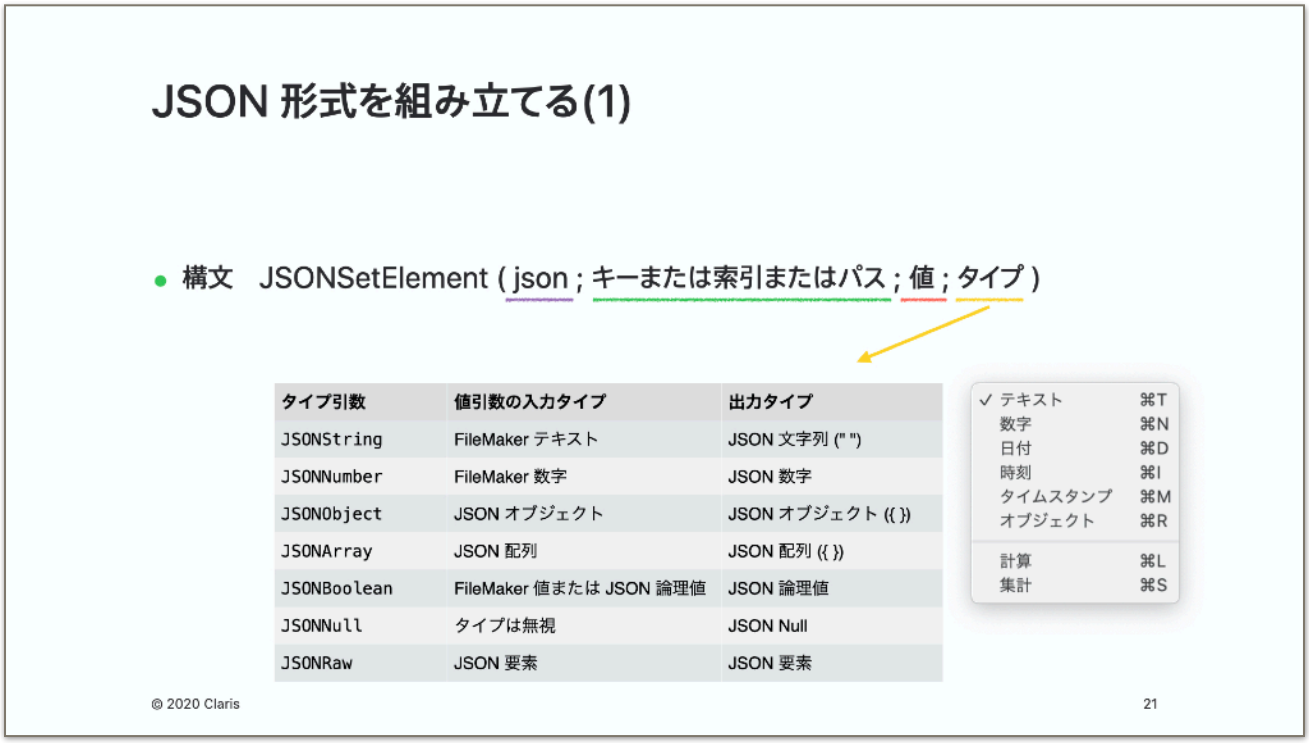
タイプ引数（第 4 引数）	値引数（第 3 引数）の入力タイプ	出力タイプ （対応する JSON のデータ型）
JSONString (1)	FileMaker テキスト	JSON 文字列 (" ")
JSONNumber (2)	FileMaker 数字	JSON 数字
JSONObject (3)	JSON オブジェクト	JSON オブジェクト ({ })
JSONArray (4)	JSON 配列	JSON 配列 ({ })
JSONBoolean (5)	FileMaker 値または JSON 論理値	JSON 論理値
JSONNull (6)	タイプは無視	JSON Null
JSONRaw (0)	JSON 要素	JSON 要素

戻り値

この関数は、JSON オブジェクトまたは配列をテキストで返します。

JSONSetElement 関数のタイプ引数と FileMaker のフィールドタイプ

ここでもう少し、JSONSetElement 関数で指定するタイプ引数について説明します。



JSONString はテキストを表すタイプで、最もよく使われるものです。JSONString の値はダブルクォーテーションでくくって指定します。

JSONNumber は数値を表すタイプで、これもよく使われます。JSON の数値型が扱うことのできる整数（10 進法表記）や浮動小数点数も扱うことができます。ただし、FileMaker では非常に広い範囲の数値（10 ^ -400 から 10 ^ 400 までと、それと同じ範囲の負の値）がサポートされていますが、JSONNumber では 18 桁を超えると精度が失われるので注意が必要です。

JSONObject は JSON オブジェクトを表すタイプです。FileMaker のフィールドタイプ「オブジェクト」とは意味が違い、画像やファイルなどを指定することはできません。JSONObject と次の JSONArray は、JSON データに入れ子にすることができます。

JSONArray は配列を表すタイプです。配列の中には、文字列や数値だけでなく、JSON オブジェクトや配列など、JSON 形式で扱うことのできるすべての値を設定することができます。JSON 配列については、後ほど例を見てみましょう。

JSONBoolean は論理値を表すタイプです。タイプに JSONBoolean を指定すると、JSONSetElement 関数は値引数を FileMaker の論理関数のように評価して、値として小文字の「true」あるいは「false」を設定します。例えば、値引数に「True」や「1」を指定すると「true」が設定され、「False」や「0」を指定すると「false」が設定されます。なお、ダブルクォーテーションでくくられた文字列（例えば「"True"」）を指定すると、「false」が設定されます。

JSONRaw をタイプに指定すると、値引数が JSON パーサで判定されて、文字列あるいは数値が値として設定されます。ただし、文字列と数字が混在する場合には意図しない結果が返ってくることがあるので、このタイプを使う場合は注意が必要です。

以上のうち、JSONString と JSONNumber はそれぞれ、FileMaker のフィールドタイプの「テキスト」と「数字」に対応付けて考えることができます。

しかし、ご存知のとおり FileMaker には、テキスト、数字以外にも、日付、時刻、タイムスタンプ、オブジェクトといったフィールドタイプがあります。これらのフィールドタイプのデータは、JSON 形式ではどのように扱うのでしょうか？

これらのデータは、JSON 関数では JSONString、つまり文字列として取り扱います。ファイルや画像データも、Base64 フォーマットにエンコードして JSONString として扱います。

なお、JSONString として受け取ったデータを FileMaker の世界のフィールドタイプで扱うには、GetAs 系のテキスト関数によって型変換してから使用します。

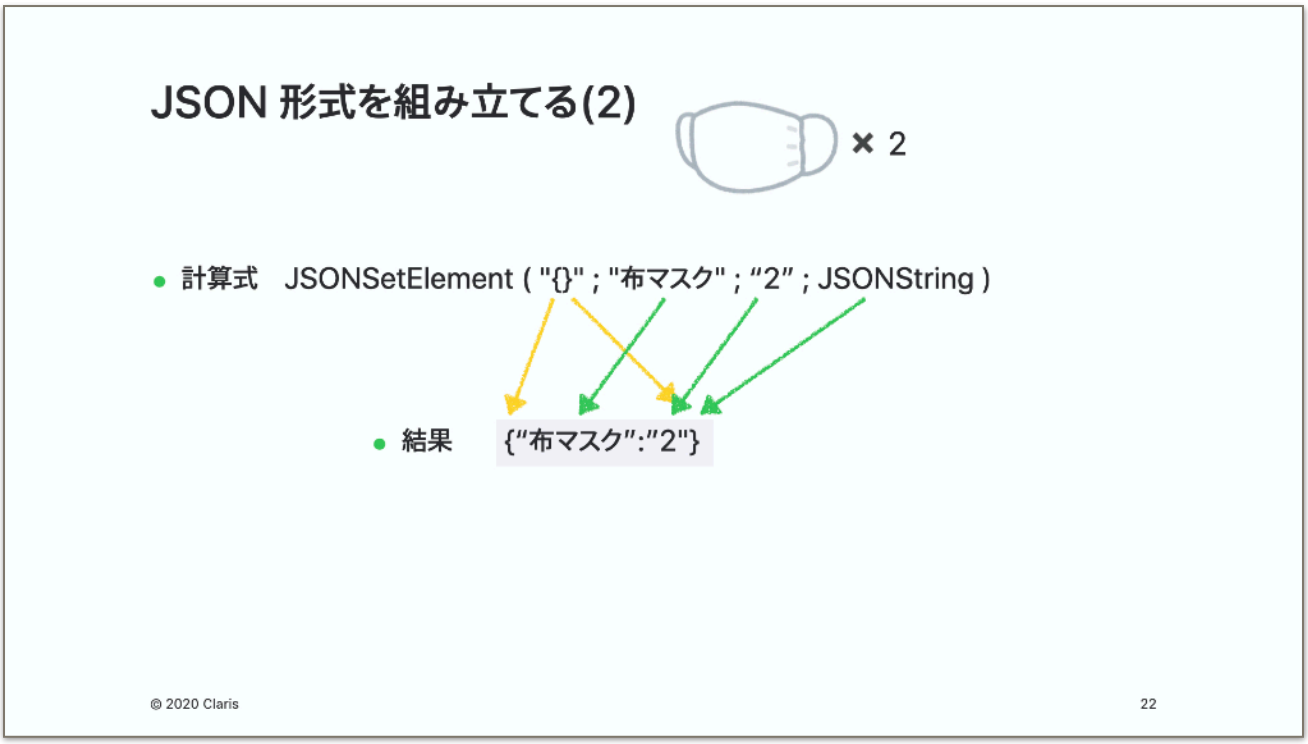
それでは、JSONSetElement 関数の使用例を見てきましょう。

JSONSetElement 関数の使用例 (1)

あるとき「布マスク」が「2 枚」支給されました。これを管理する JSON 形式のデータを作ることを考えます。

計算式とその結果は次のようになります。作成する JSON データは最初のデータなので、計算式の第 1 引数には「"{}"」を指定します。第 2 引数の「"布マスク"」はキーとなる名前です。第 3 引数として値となる「"2"」を、第 4 引数として値のタイプ「JSONString」と指定しています。この結果として、「"布マスク" : "2"」というキーと値のペア（要素）を 1 つだけ持つ、最もシンプルな JSON オブジェクトが得られました。

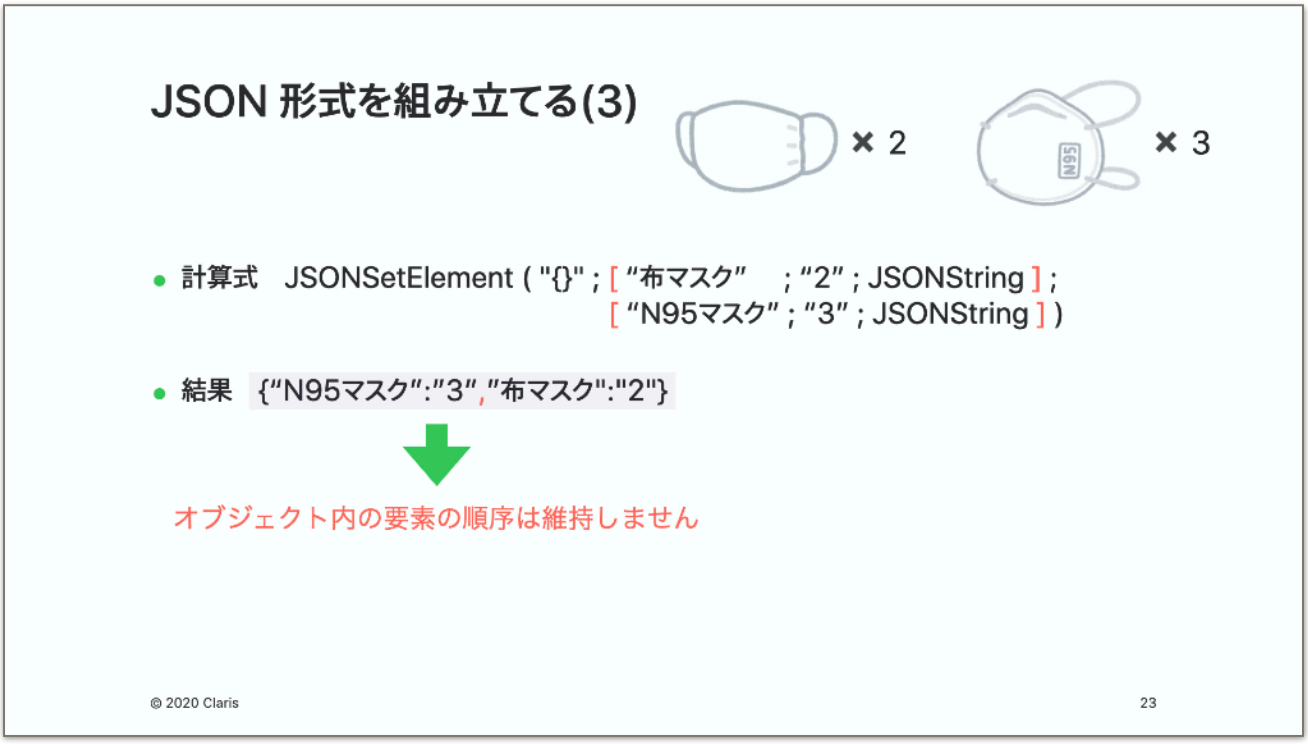
なお、作成した JSON オブジェクトを使うスクリプトや他のアプリケーションで JSON オブジェクト内の値を使った四則演算をする処理がなければ、この例のように、数字の値に対して「JSONNumber」タイプではなく、扱いやすい「JSONString」タイプを使っても問題ありません。



JSONSetElement 関数の使用例 (2)

JSONSetElement 関数は、第 2, 3, 4 引数を角カッコでくくって、複数個の要素（キーと値のペア）を同時に設定することができます。

次の例では、「布マスクが2枚」と「N95 マスクが 3 枚」に対応する引数を角カッコでくくって 2 つ同時に指定し、結果としてコンマで区切られた 2 つの要素からなる JSON 形式のデータが返っています。



ここで、JSONSetElement 関数で指定した「布マスク」と「N95 マスク」の要素の順序が、結果の JSON データでは入れ替わっていることに注意してください。そもそも、JSON オブジェクトの中では要素の順序に意味はないので、JSONSetElement 関数の生成する JSON データでも要素の順序は維持されません。JSON データの中では要素に順序はないと考えてください。

なお、JSON データの中で値（文字列、数値やJSON オブジェクトそのものなど）に順序を持たせるためには、配列を使います。これについては、後の JSONSetElement 関数の使用例 (4) で紹介します。

JSONSetElement 関数の使用例 (3)

続いて、「在庫マスク」という箱の中にいろいろなマスクが管理されている場合を考えます。

この場合は、キーとして「在庫マスク.布マスク」というパス文字列を指定します。これは、「在庫マスク」の中に「布マスク」があるという構造を意味しています。そして、生成された JSON データには、キー「在庫マスク」に対応する値として、中カッコでくくられた新たな JSON オブジェクト「{ “布マスク” : “2” } 」が出来ています。

だんだん JSON データが複雑になって読みづらくなってきました。

これを、次の JSONFormatElements 関数を使って解決します。

JSONFormatElements 関数

JSONFormatElements 関数は、引数で指定されたテキスト（JSON データ）を解析し、タブ文字と改行コードを追加することによって、人間に読みやすい形に整形して返します。

構文

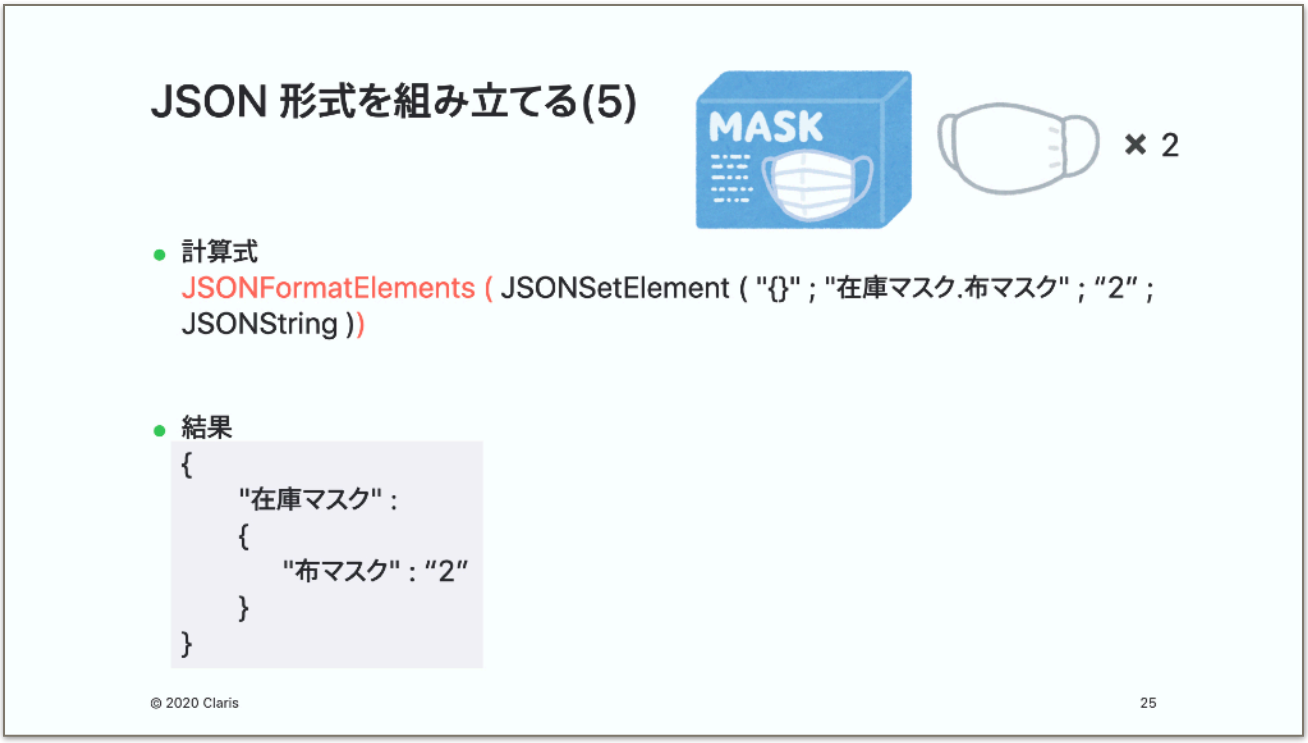
JSONFormatElements (json)

使用例

JSONFormatElements ("{\"在庫マスク\" : {\"布マスク\" : 2, \"花柄マスク\" : 3 }}")
※この式は、次のようなテキストを返します。

```
{
        "在庫マスク" :
        {
                "布マスク" : 2,
                "花柄マスク" : 3
        }
}
```

この関数の引数に JSONSetElement 関数を指定すると、次のようにJSONSetElement 関数の結果の JSON データにタブ文字と改行が追加されて見やすくなります。

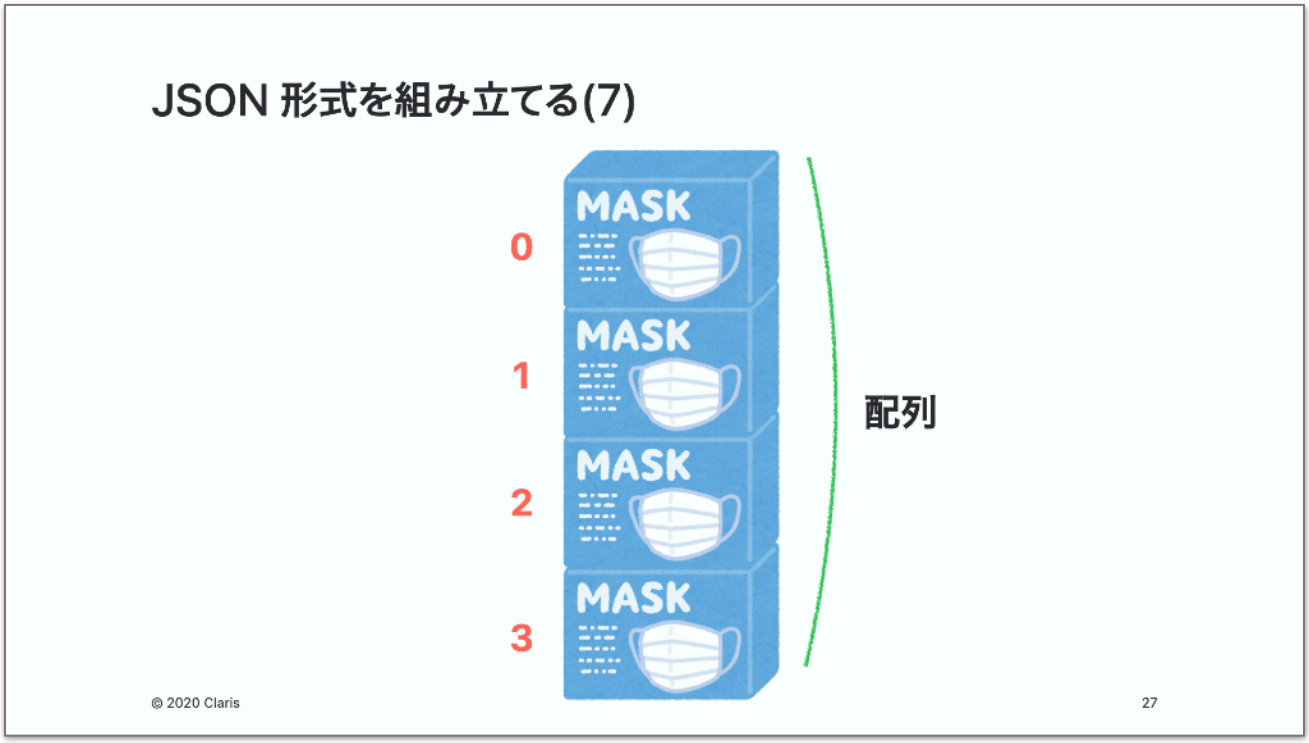


以降のJSON データの表記はすべて、このように整形した形で記載しますが、計算式の記述を簡潔にするため JSONFormatElements 関数の記載は省略していますので、ご了承ください。

JSONSetElement 関数の使用例 (4)

これまで「在庫マスク」という箱は 1 つだけを考えていましたが、「在庫マスク」が複数個、例えば古い順に管理されている場合を考えます。

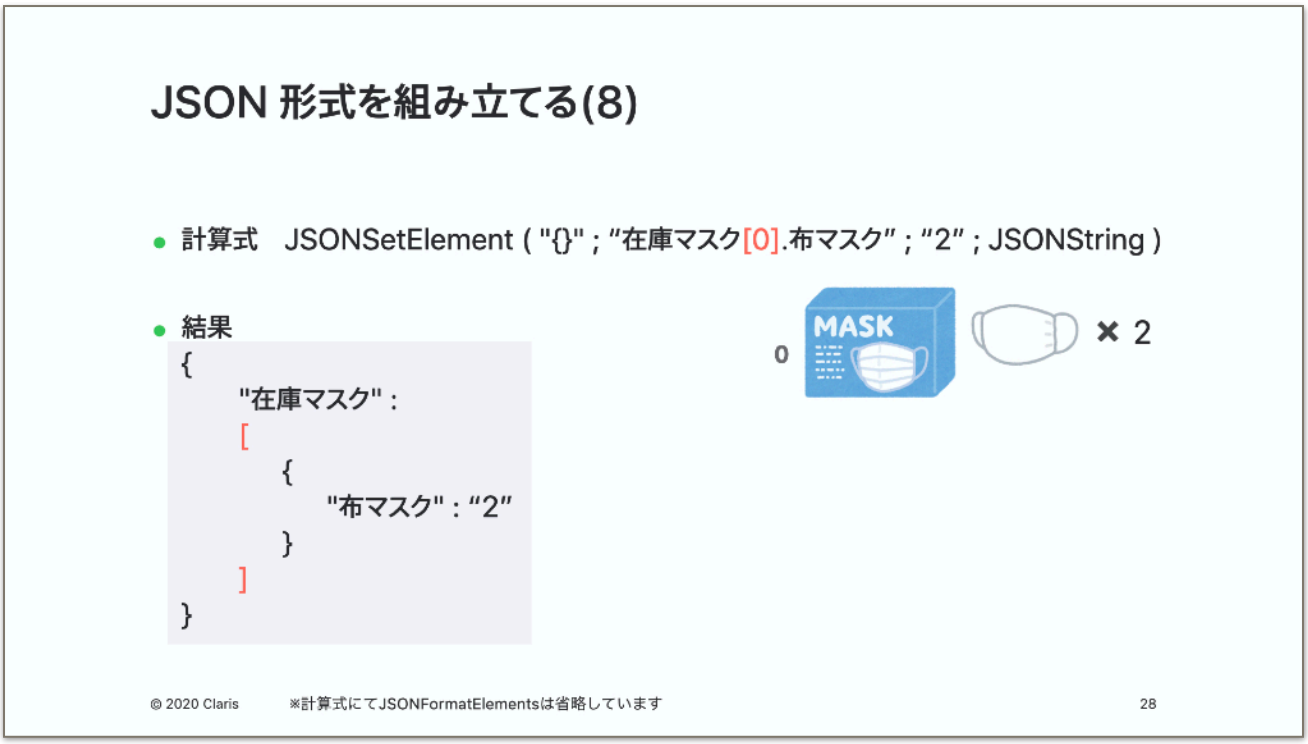
JSON 形式のデータの中で順序を持った値を取り扱うには、配列を使います。配列中の要素の順序は、JSON オブジェクト中の要素とは違って、順序が勝手に入れ替わったりはしません。また、索引（配列の何番目かを表す数）を使ってアクセスすることができます。このとき、索引の番号は「0」から始まることに注意してください。



では、配列を使って複数の「在庫マスク」を管理する JSON データを生成してみましょう。

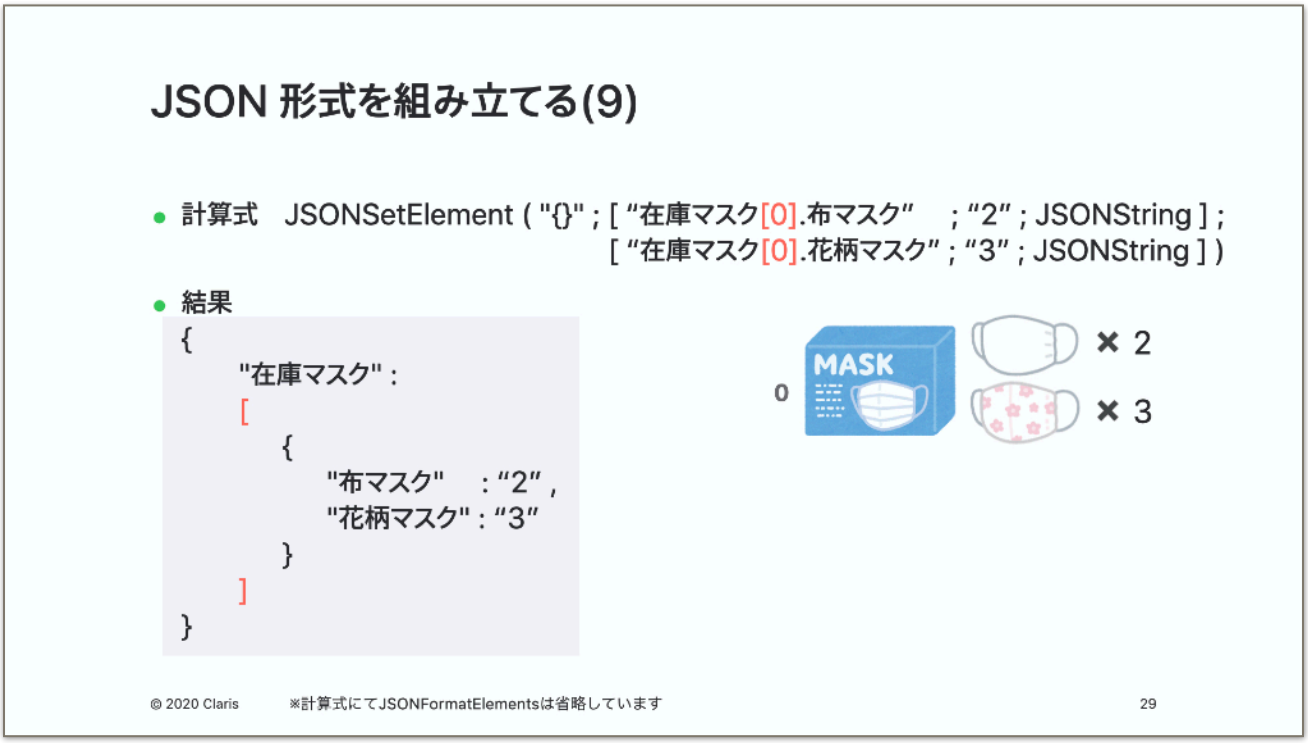
次の例は、「在庫マスク」が 1 つだけあって、中に「布マスク」が「2 枚」ある場合です。ただし、前の使用例 (3) の場合とは異なり、今後「在庫マスク」の箱の数が増えていくことが想定されていたとします。このため、「在庫マスク」が 1 つだけでも、配列を使った JSON データを生成します。配列の中身は 1 個なので、配列の索引は「0」になります。

このときの JSONSetElement 関数の結果は、使用例 (3) の結果とは異なり、キー「在庫マスク」に対応する値が、角カッコでくくられた配列になっていることがわかります。



JSONSetElement 関数の使用例 (5)

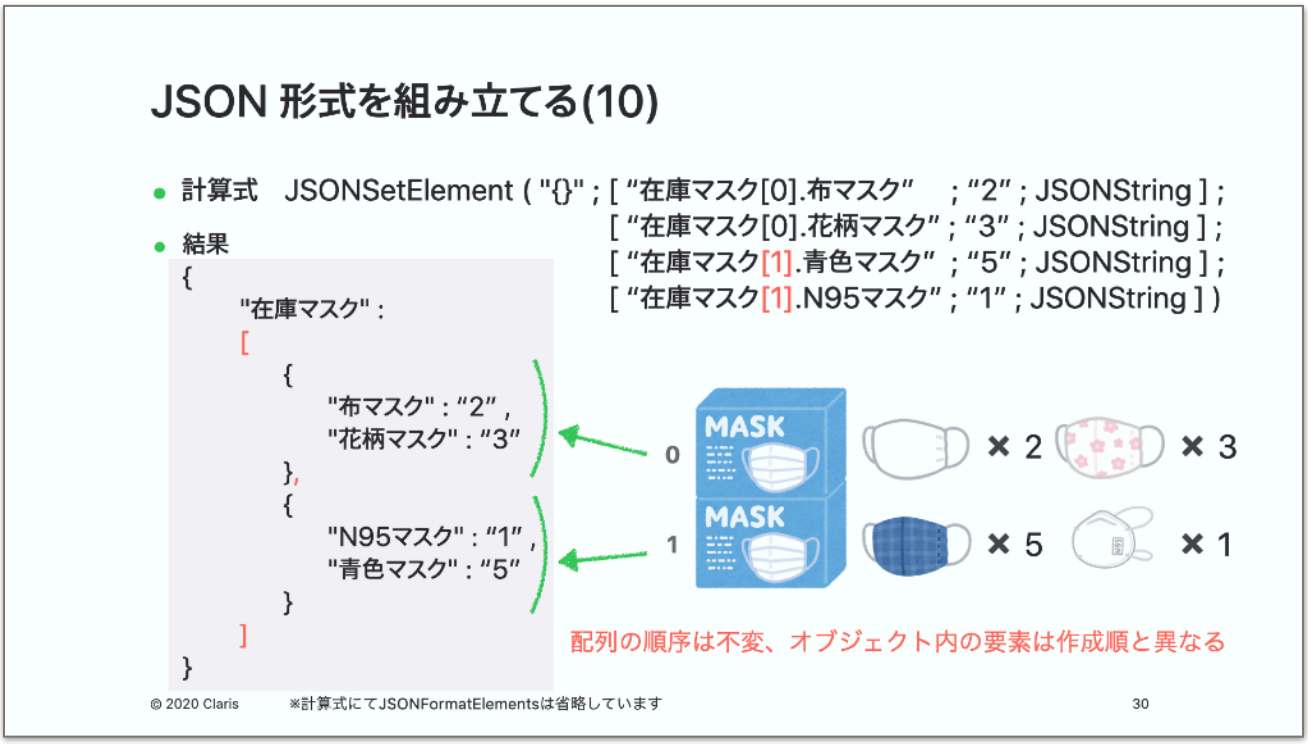
次に、「在庫マスク」が 1 つだけあって、中に「布マスク」が「2 枚」、「花柄マスク」が「3 枚」ある場合を考えます。この場合は、次のように JSONSetElement 関数を記述します。



結果は、キー「在庫マスク」の値が配列である要素を持つ JSON オブジェクトとなり、その配列の最初（0 番目）の JSON オブジェクトに、キーがそれぞれ「布マスク」と「花柄マスク」の 2 つの要素が入っています。

JSONSetElement 関数の使用例 (6)

今度は「在庫マスク」を 2 つ持つ場合を考えます。次の図のように、1 つ目の「在庫マスク」には「布マスク」が「2 枚」と「花柄マスク」が「3 枚」、2 つ目の「在庫マスク」には「青色マスク」が「5 枚」と「N95 マスク」が「1 枚」入っているとします。1 つ目の「在庫マスク」は配列の 0 番目、2 つ目の「在庫マスク」は配列の 1 番目に指定します。



この結果は、キー「在庫マスク」の値が 、2 つの JSON オブジェクトが入った配列である要素を持つ JSON オブジェクトになります。

なお、「在庫マスク」の配列の中で、「布マスク」と「花柄マスク」の要素から成る JSON オブジェクト（在庫マスク[0]）は、「青色マスク」と「N95 マスク」の要素から成る JSON オブジェクト（在庫マスク[1]）よりも前にありますが、後者の JSON オブジェクト（在庫マスク[1]）の中では「青色マスク」と「N95 マスク」の要素の順序が、JSONSetElement 関数で指定した順序と異なっています。繰り返しになりますが、JSON オブジェクト内の要素に順序はないので、このような結果になっても気にしてはいけません。

JSON データの配列の記法

FileMaker における JSON データの配列を FileMaker Pro ヘルプなどで見ると、2 つの記法があることに気がつかれた方もいるかもしれません。配列の角カッコの後にパスを表す「.」（ピリオド）がある場合と無い場合です。

例えば、次の 2 つの式の第 2 引数は、どちらも同じ「クラリス病院」の「在庫マスク」配列の最初の配列要素の「布マスク」を意味していますが、2 つ目の式には角カッコの後に「.」（ピリオド）がありません。これらを FileMaker Pro のデータビューアで確認すると、実は、同じ結果（{“クラリス病院”:{“在庫マスク”:[{“布マスク”:”2”}]}}）が返されます。

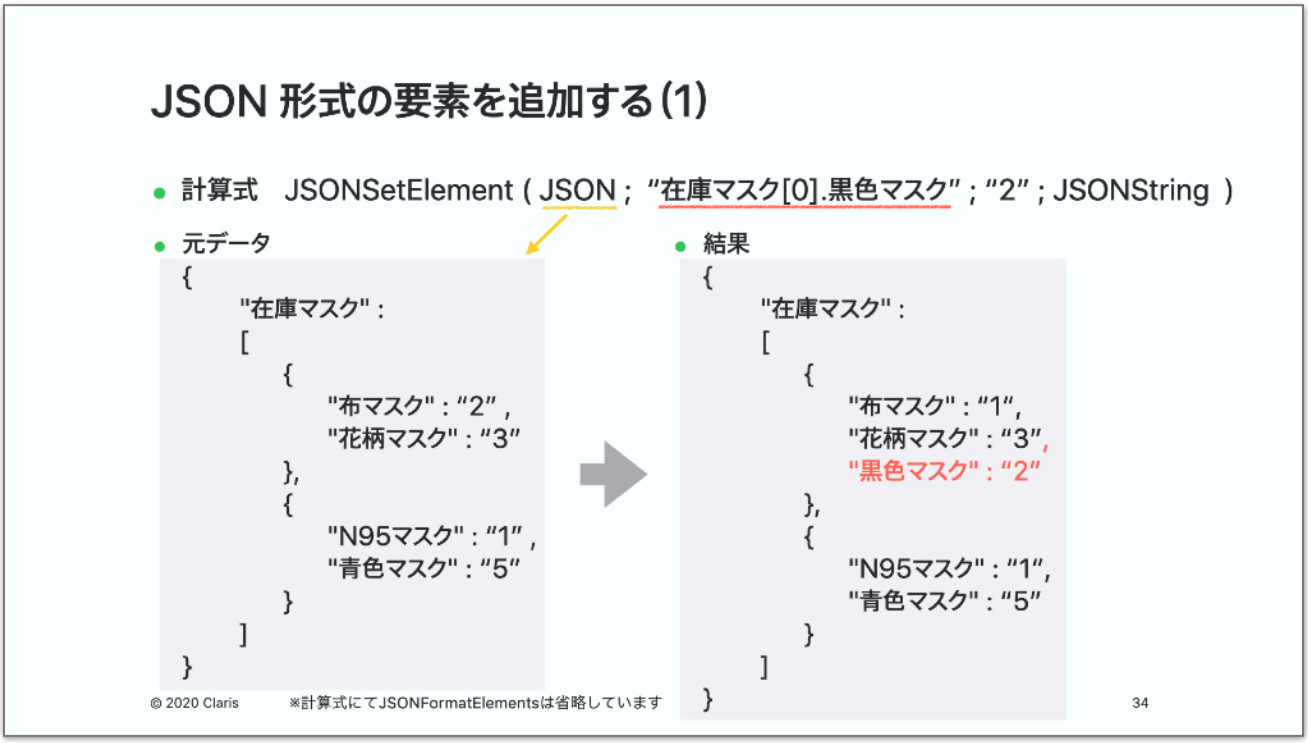
つまり、FileMaker ではどちらの記法も利用できるということです。

JSON形式を変更する

JSON 関数で JSON 形式のデータを生成したら、次は、その JSON データに要素を追加したり、内容を変更したり、要素を削除したりしてみましょう。

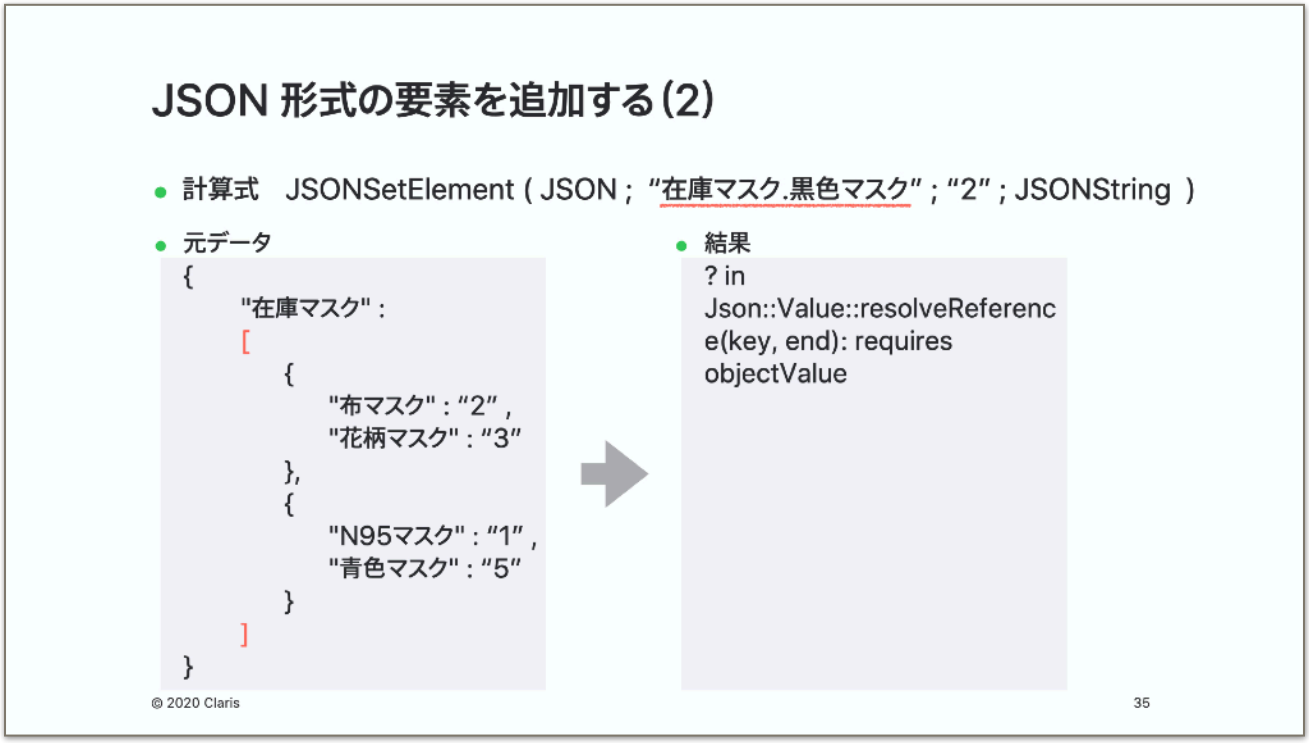
要素を追加する

JSON データに要素を追加するには、JSON データを最初に作成したときと同じ、JSONSetElement 関数を使います。JSONSetElement 関数は、第 1 引数（図では「JSON」）に空白あるいは “{}” の代わりに JSON オブジェクトが指定されていたら、その JSON オブジェクトに、第 2 引数以下で指定された要素を追加します。次の例では、「在庫マスク」の最初（0 番目）の配列要素である JSON オブジェクトに「2」枚の「黒色マスク」を追加しています。



ただし、元データ（第 1 引数の JSON オブジェクト）に第 2 引数で指定する JSON 構造が存在しない場合にはエラーになります。

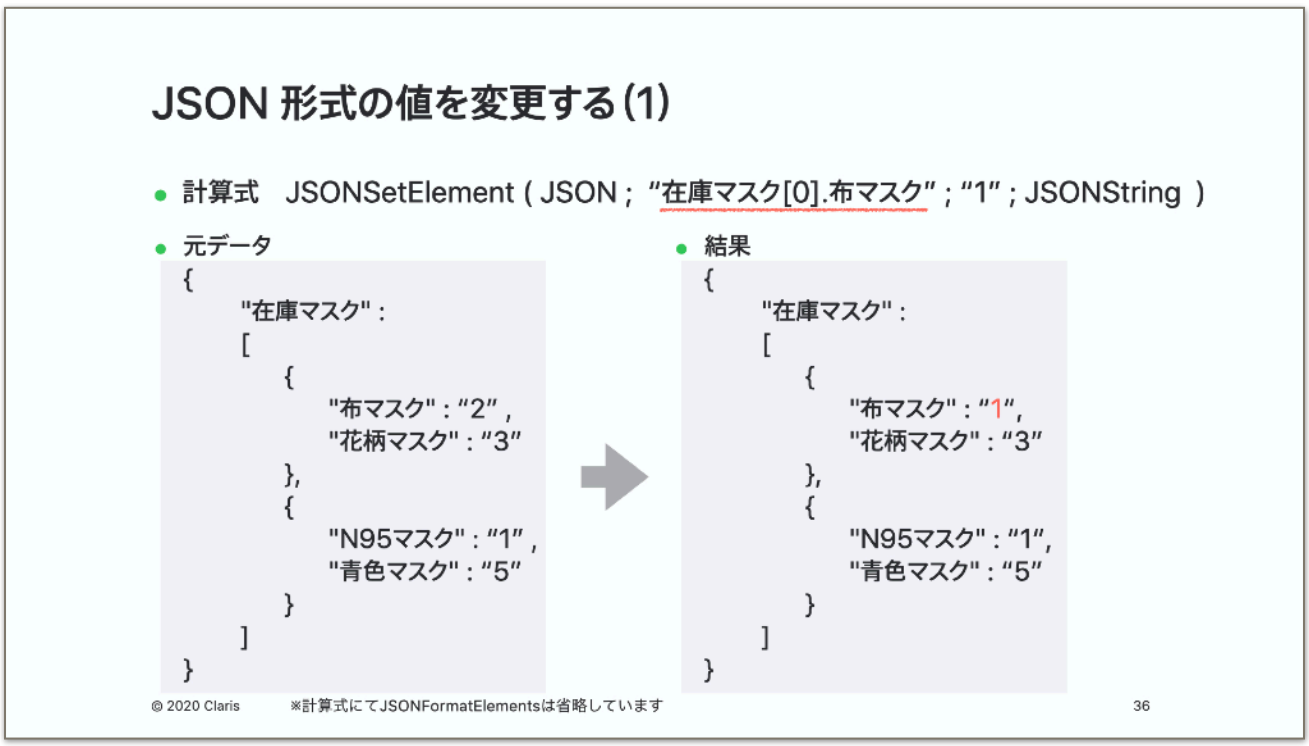
次の例では、元データのJSON オブジェクトの「在庫マスク」の値は配列なのに、第 2 引数で配列の指定（索引番号）を忘れてしまっています。このため、結果として「?」マークに続いてエラーメッセージが返っています。



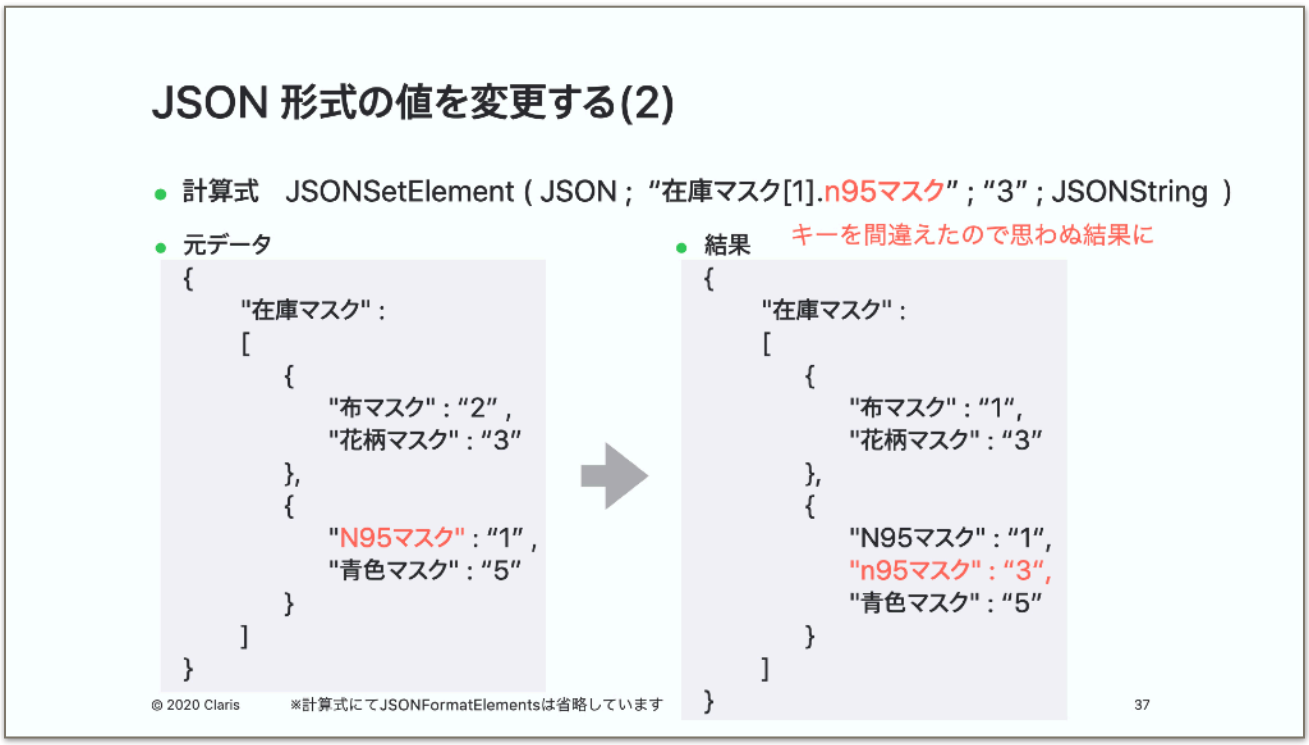
値を変更する

次は、JSON データの値を変更する場合を考えます。例えば、今までの例で「布マスク」の在庫が残り 1 枚になってしまった場合です。

この場合も、要素を追加する場合と同様に、JSONSetElement 関数を使います。第 2 引数に既存の構造に対応したパスを指定すると、その値が第 3 引数の値で上書きされます。



このときも、変更する対象（第 2 引数）を正しく指定しないと、思わぬ結果を招くことがあるので注意が必要です。次の例は、2 つ目の「在庫マスク」の中の「N95 マスク」の在庫数を「3」に変



更するつもりだったところ、キーの「N95 マスク」の最初の「N」を間違えて小文字の「n」にしてしまっています。このため、元の「N95 マスク」の値は変更されず、小文字の「n」から始まる「n95 マスク」の要素が「在庫マスク[1] 」オブジェクトに新しく追加されてしまいました。

このように、JSON の世界では大文字と小文字が区別されるので、文字列を記述する際は注意しましょう。

要素を削除する

最後に、JSON オブジェクトから要素を削除する場合を考えます。この場合は、JSONDeleteElement 関数を使用します。

JSONDeleteElement 関数は次のような構文を取ります。

構文

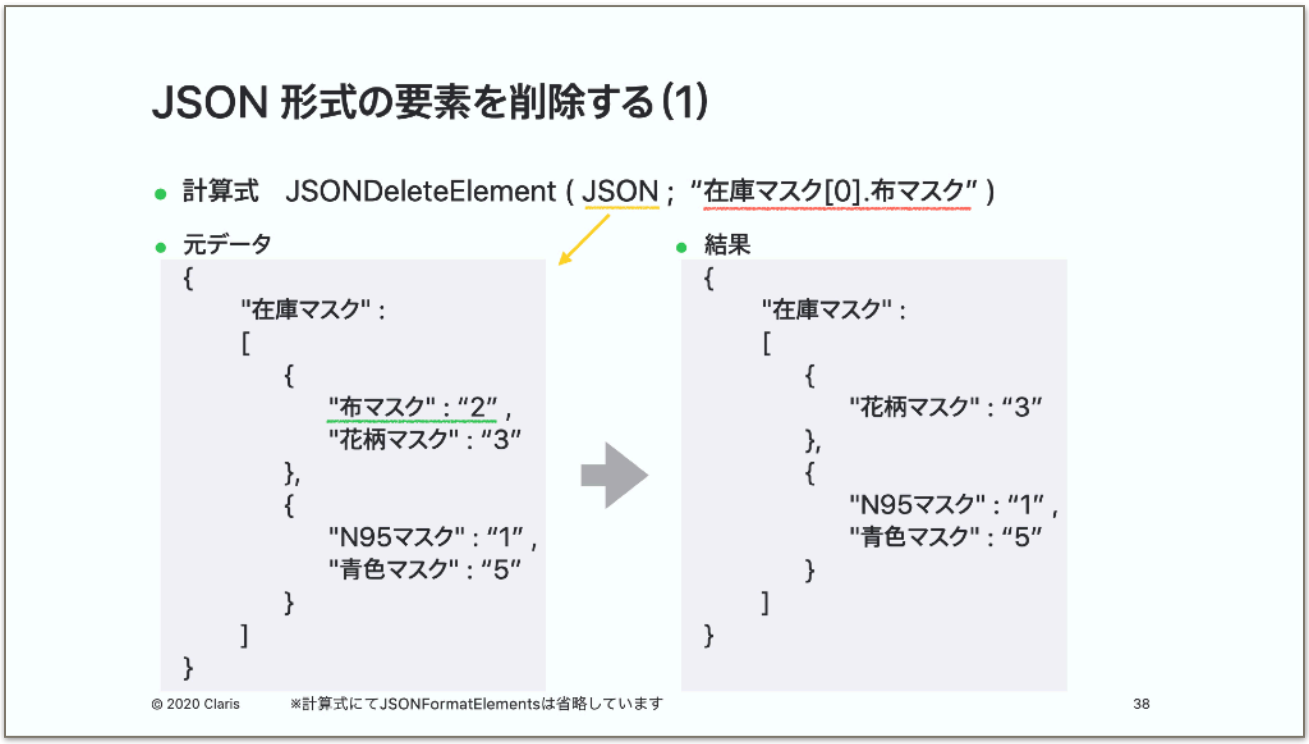
JSONDeleteElement (json ; キーまたは索引またはパス)

使用例

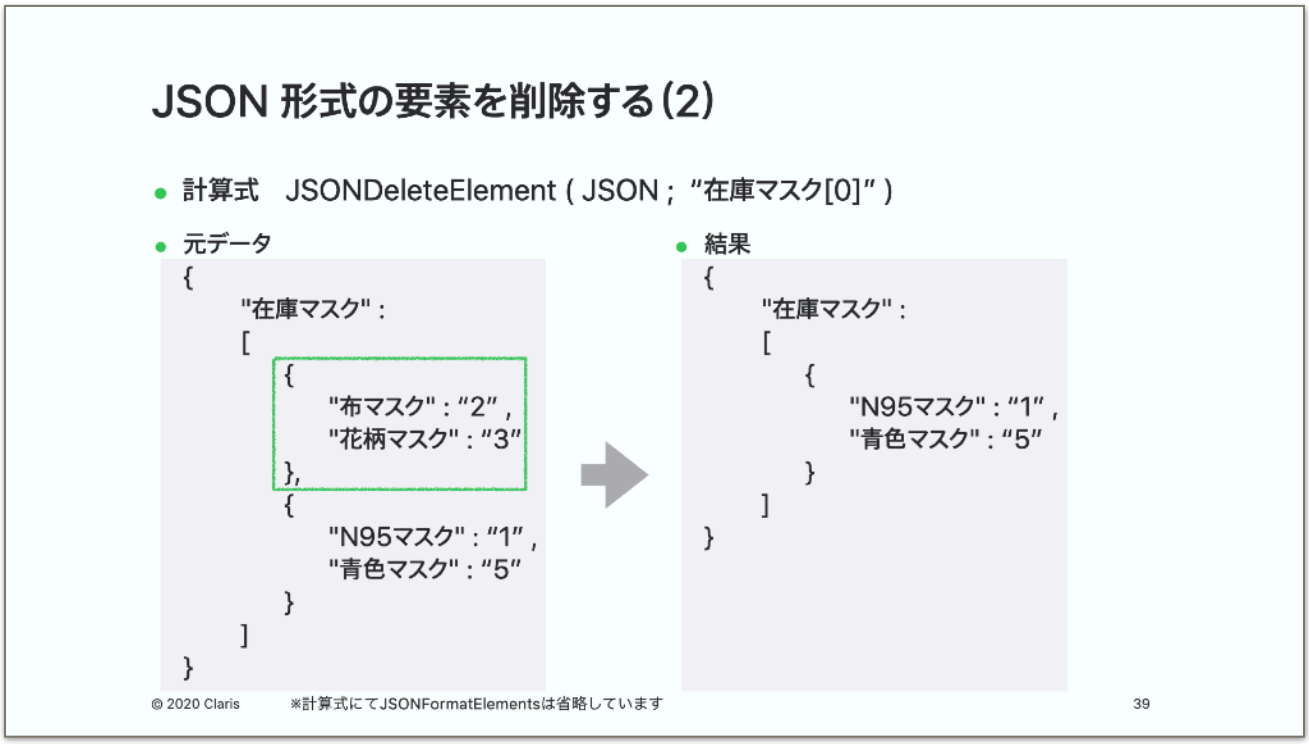
JSONDeleteElement ($JSON ; “布マスク”)

JSONDeleteElement 関数の第 1 引数には、JSON 形式のデータが入っている FileMaker のフィールドや変数を指定し、第 2 引数には削除したい要素のキーまたは索引またはパスを指定します。このとき、値を指定する必要はありません。

次の例は、「在庫マスク」配列の中の最初（0 番目）の配列要素の JSON オブジェクトから、「布マスク」の要素を削除しています。



また次の例は、「在庫マスク」配列の最初の配列要素である JSON オブジェクト全体を削除しています。



なお、配列の要素そのものを削除した場合、残った配列要素に対応する索引番号が変わるので、注意してください。上の例では、当初は「在庫マスク[1]」でアクセスしていた2番めの配列要素が、最初の要素が削除されたことによって「在庫マスク[0]」になります。

JSON形式を利用する

最後に、これまで学習した JSON 関数をカスタム App で利用してみましょう。

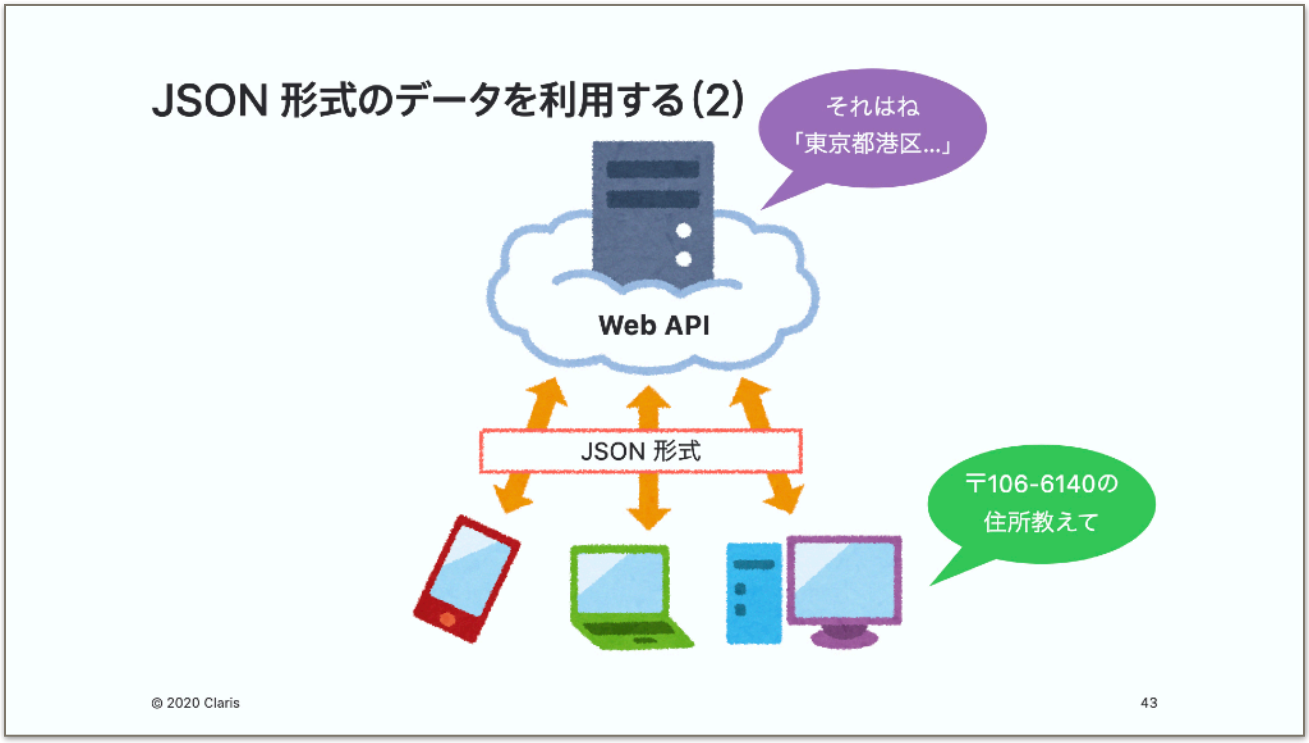
顧客情報を管理する

顧客の情報を管理するカスタム App （顧客情報 App）を考えます。この顧客情報には、郵便番号を含む顧客の住所データが入っているとします。

顧客情報 App は、新規顧客情報をユーザが簡単に入力できるように、ユーザが郵便番号を入力するだけで、約 12 万件のレコードを持つ郵便番号マスタ（郵便番号マスタ App）を参照して自動的にデータベースに住所が入力されるようになっています。

ところが、実は郵便番号そのものが日本郵便によって月に一回更新されています。つまり、顧客情報 App が参照している郵便番号マスタも、月に一回、誰かが更新しないと正しい住所データが取得できない、ということになります。

そこで、マスタデータの面倒なメンテナンス作業をする代わりに、常に最新の郵便番号データを提供してくれる Web API を利用することを考えます。昨今のWeb API とのデータ送受信には JSON 形式のフォーマットが使われることが多いため、FileMaker でも Web API から受け取ったデータを JSON 関数を利用して取り扱うことができます。



以下の例では、次の Web API をカスタム App から利用する場合を考えます。

- 郵便番号／住所／緯度経度データサービス Heart Rails Geo API[1]
  郵便番号による住所検索 API:　https://geoapi.heartrails.com/api.html#postal

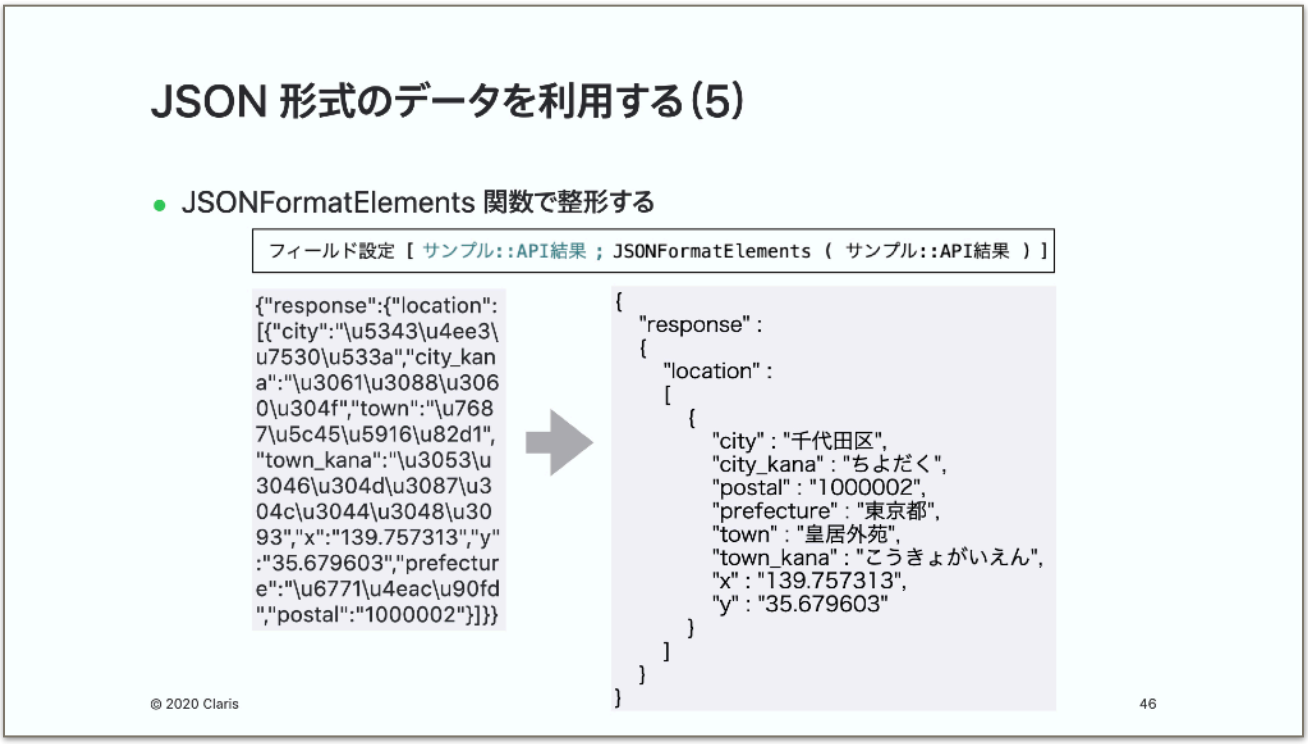
WEB APIからJSONデータを取得する

FileMaker で Web API からデータを取得するには、[URL から挿入] スクリプトステップを使います。このスクリプトステップは、指定した URL から取得したデータをフィールドや変数に入力します。

次の例は、[URL から挿入] スクリプトステップを使って郵便番号による住所検索 API に郵便番号「1000002」の住所をリクエストするスクリプトと、その実行結果（レスポンス）です。「サンプル::API結果」に入力された実行結果は、 JSON 形式になっていることはわかりますが、
キー "location" に続く配列要素の値は UTF エンコーディングされていて、（ほとんどの）人間には読むことができません。

[1] https://geoapi.heartrails.com

ここで、この実行結果を JSONFormatElements 関数に入力してみます。これによって、先程の実行結果にはタブ文字と改行が追加され、JSON 形式の構造がわかりやすくなっただけでなく、配列要素の値もデコードされて普通の人間にも読めるようになりました。



続いて、取得した JSON データから必要なデータを取り出します。

JSONGetElement 関数

JSON データから必要なデータ（値）を取り出すには、JSONGetElement 関数を使います。

構文

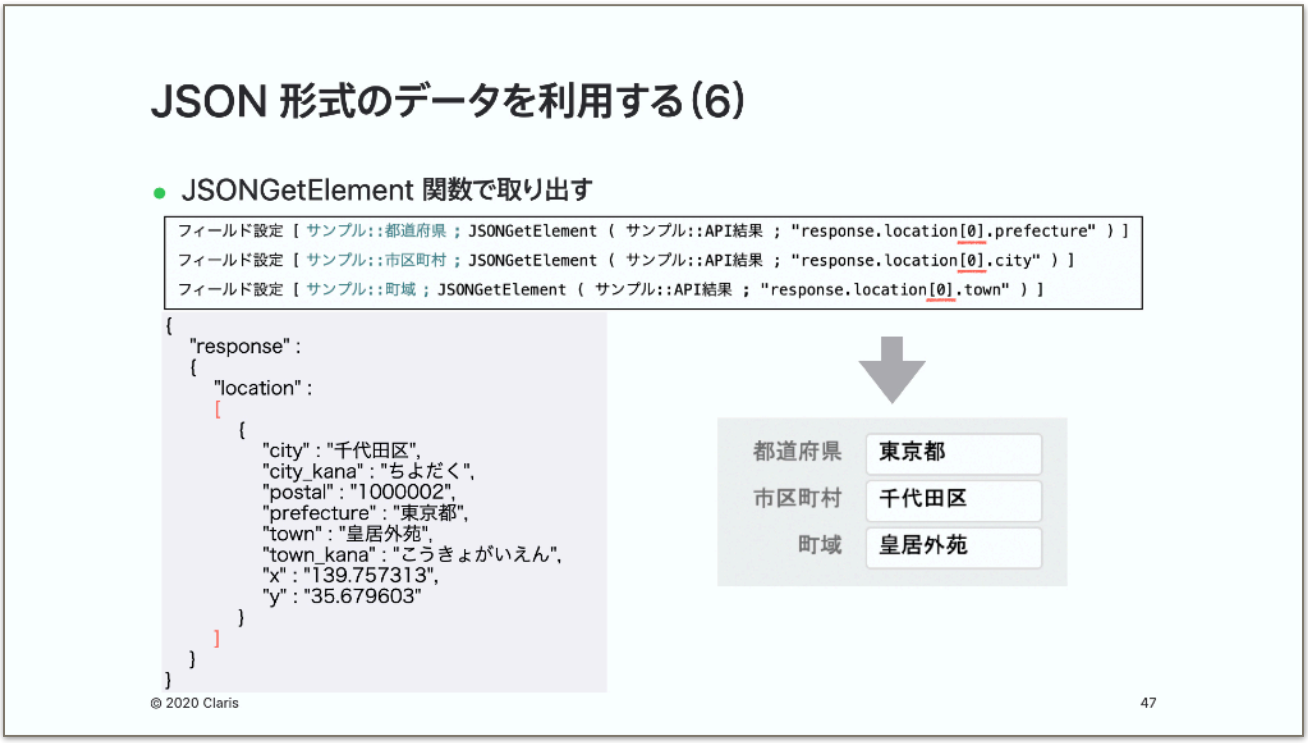
JSONGetElement (json ; キーまたは索引またはパス)

使用例

JSONGetElement ($JSON ; “布マスク”)

この関数は、第 1 引数に指定された JSON データから、第 2 引数で指定されたキーまたは索引またはパスに対応する値を取り出します。

次の例では、郵便番号による住所検索 API のレスポンスである JSON データから、都道府県（”prefecture”）、市区町村（”city”）、町域（”town”）のキーに対応する値を取り出し、「サンプル」テーブルのそれぞれのフィールドに設定しています。ここで、元のJSON データ中の住所データ（”location”）の値は配列になっているので、配列索引として「0」を指定していることに注意してください。

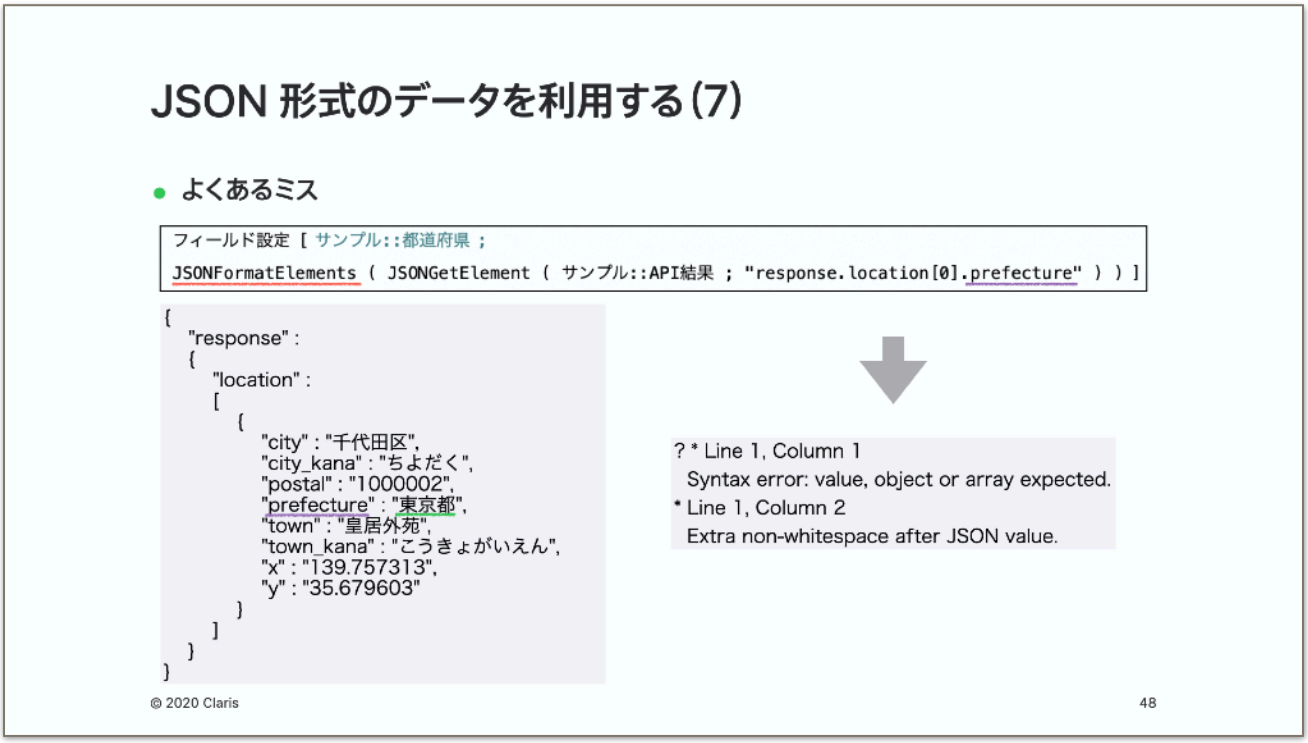


なお、こういったスクリプトを書くとき、最初はデータビューアを使って JSON データを目で確認しながら試行錯誤することをお勧めします。特に、Web API の仕様がはっきりとわからない場合など、まずはリクエストして返ってきた値をデータビューアで確認しながらスクリプトを作っていくことは、手慣れた FileMaker ユーザの間でもよく使われる手法です。

このとき、データビューアで JSONFormatElements 関数を使うと、計算式の結果の JSON データを整形された形で確認することができるので便利です。多くの場合は、JSONFormatElements 関数を入れたままの式をスクリプトにそのまま使っても問題なく動作するので、データビューアで動作を確認した式をそのままスクリプトに利用することができます。

ただし、エラーが起こることがあります。

次の例のスクリプトは、右下のように「Syntax error（構文エラー）」が起こっています。このスクリプト内の JSONGetElement 関数は、結果が「response.location[0].prefecture」の値、つまり都道府県名である「東京都」といったテキストであって、JSON データではありません。このスクリプトでは、JSON オブジェクトを引数とする JSONFormatElements 関数に、JSON オブジェクトではなくテキストを渡しているので、「Syntax error」となってしまったのです。



なお、この例の計算式の JSONFormatElements 関数の部分は、データビューアで実行しても「Syntax error」になります。安易に式を使い回す前に、まずデータビューアで結果を確認することが重要です。

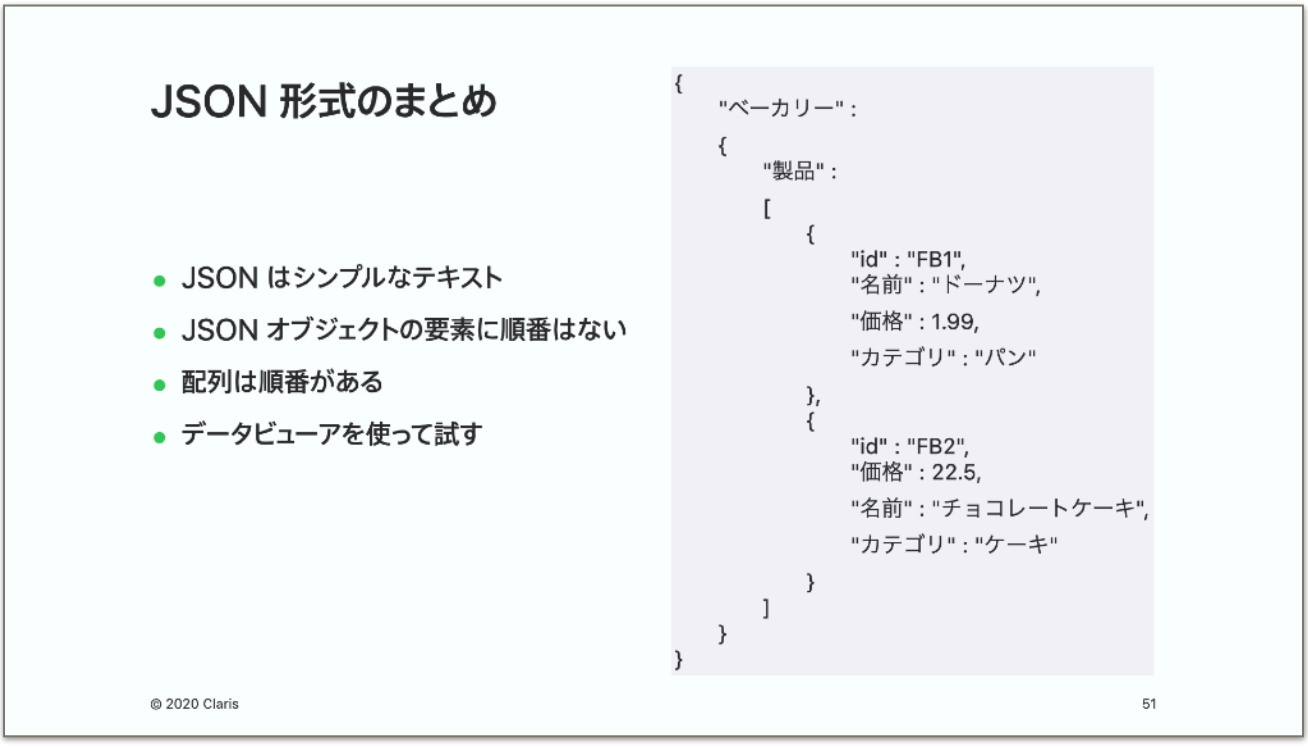
JSONFormatElements 関数はとても便利な関数ですが、結果が JSON 形式のデータフォーマットではない場合にエラーが出るということと、JSONFormatElement 関数を使った式を使い回すときには注意が必要、ということを覚えておいてください。

その1のまとめ

本編では、FileMaker で JSON 形式のデータを活用していくための「基礎」として、JSON 形式の基本的な知識と FileMaker の基本的な JSON 関数の使い方をみてきました。

ここまでご覧になってきておわかりのように、JSON 形式はとてもシンプルなテキストによるデータフォーマットです。キーと値のペアでデータを表現し、その順序に意味はありません。唯一、配列を使用する場合のみ、データの順序に意味を持たせることができます。このような特徴を利用して様々な構造のデータを表現することができるので、みなさんもいろいろな場面で JSON データを使ってみてください。

FileMaker で JSON 形式のデータを使い始める際に重要なのは、データビューアを使うことです。最初は思った結果が出ないかもしれませんが、データビューアを使って JSON 関数を使った計算式を試しながらスクリプトを作成していくことによって、JSON と JSON 関数に慣れ親しみ、より理解を深めていくことができるでしょう。その結果として、JSON を利用する際の「ルール」を覚えてしまえば、JSON 形式のデータと JSON 関数をとても便利に使うことができるようになっているはずです。



なお、本編には続編「JSON 関数事始め - その 2 -」も用意しています。さらに一歩進んだ JSON の利用方法を「その 2」で確認してください。

参考資料

1. FileMaker Pro 19 ヘルプ
   https://help.claris.com/ja/pro-help/#page/FMP_Help%2Findex.html

2. 「FileMaker 16 の新機能『JSON 関数』」、株式会社イエスウィキャン、May 2017.
   https://ywc.com/filemaker/?p=3908

3. Krueger, S., "JSON, You Deserve [Arrays]," Claris Engage 2020, Sept. 2020.
   https://youtu.be/ZpyMs9_VJ7w

4. Kos, M., "FileMaker JSON Serialization and Deserialization Gotchas," Soliant Consulting, Inc., Aug. 2019.
   https://www.soliantconsulting.com/blog/filemaker-json-serialization/
   （邦訳）nomfjmt, 「FileMaker <<>> JSON データ変換時の注意点」、
   https://notonlyfilemaker.com/2019/09/filemaker-json-serialization/ (Not Only FileMaker)

5. Krueger, S. A., "FileMaker 16's New {JSON} Functions," LuminFire Brilliant Solutions, Sept. 2017.
   https://luminfire.com/2017/09/30/filemaker-16s-new-json-functions/

6. Introducing JSON
   https://www.json.org/

7. 「FileMaker JSON 関数事始め - その 2 - 」, Claris International Inc.
   https://downloads.claris.com/kk/downloads/pdf/JA_FileMaker_JSON_2.pdf

8. Claris オンデマンド Web セミナー「はじめての JSON 関数　その 1」, Claris International Inc.
   https://downloads.claris.com/kk/video/webseminar/20200929_fm19_json_1.mp4
   https://youtu.be/Syobq2XSZaM