JSON であって JSON ではない!?

オブジェクト

Object is not JSON by Laurent Esnault

#4DSummit

オブジェクト is not != JSON

オブジェクト型が好まれる理由

- 値を**いくつでも**まとめて受け渡すことができる
- ポインター**不要**
- プロセス変数を列挙する代わりに情報を**類別されたクラス**として扱うことができる
- 番号ではなく**名前**で引数が参照できる
- 簡単かつ安全に引数が**追加**できる
- 自然なテキスト形式

JS Inspired

#4DSummit

プロパティに対するアクセス

- **OB Get** (object ; propertyName {; type})

  どんなプロパティ名でも良い
  型を指定することができる

- **object.**propertyName

  プロパティ名に**制約**あり
  型の変換はしない

- **object**[propertyName]

  どんなプロパティ名でも良い
  型の変換はしない

- **WP GET ATTRIBUTES**

  アシスト付き

#4DSummit

4D Write Pro オブジェクトにアクセスする4つの方法

Four Different ways to Use Objects with 4D Write Pro!

https://blog.4d.com/four-different-ways-to-use-objects-with-4d-write-pro/

#4DSummit

C_OBJECT 変数 is not != オブジェクト

- **変数**は名札のついた容器みたいなもの

- 容器の中に**値**が入っている

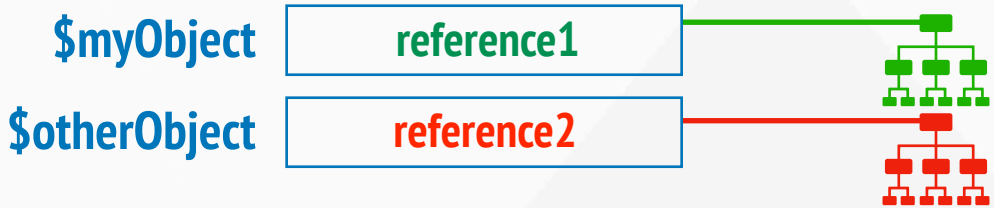
- あるコマンドは**変数**を受け取り，あるコマンドは**値**を受け取る
- メソッドが受け取ることができるのは**値**のみ

オブジェクトの参照

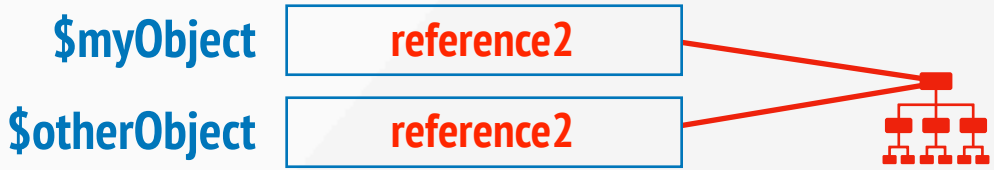
変数を代入すれば参照がコピーされる

$myObject:=**New object**

$otherObject:=**New object**



$myObject:=$otherObject



オブジェクトのプロパティ is not != 変数

- 変数を受け取るコマンドにプロパティを渡すことはできない

✖ **BLOB TO VARIABLE** ($blob ; a.b.c)

- プロパティをポインターで指定することはできない

✖ ->a.b.c

OB SET の特殊性

- **OB SET** ($myObject ; "code" ; 1)

渡されているのは**変数**　**reference**

- **OB SET** (myGetObject() ; "code" ; 1)

渡されているのは**値**　**reference**

OB SET に空の変数/フィールドが渡された場合には**新規オブジェクト**が代入される

$myObject.code:=1 にこの特殊なルールはない

Null および Undefined

- **Undefined** は『**存在しない**』
    - 存在しないモノのプロパティにアクセスすると **Undefined** が返される
- **Null** は『**存在するが値を知ることはできない**（状態）』
    - ヌルであるモノのプロパティにアクセスを試みると**エラー**が返される
- 空の **C_OBJECT, C_COLLECTION** および **C_POINTER** は**Null**であるとみなされる（値）
    - 未定義の変数（**Undefined**）と混同してはいけない
- = および # 演算子は **Null** と **Undefined** を同等のものとして評価する
    - If (a.b = **Null**)
- **Null** には**状態・値・関数**という側面がある

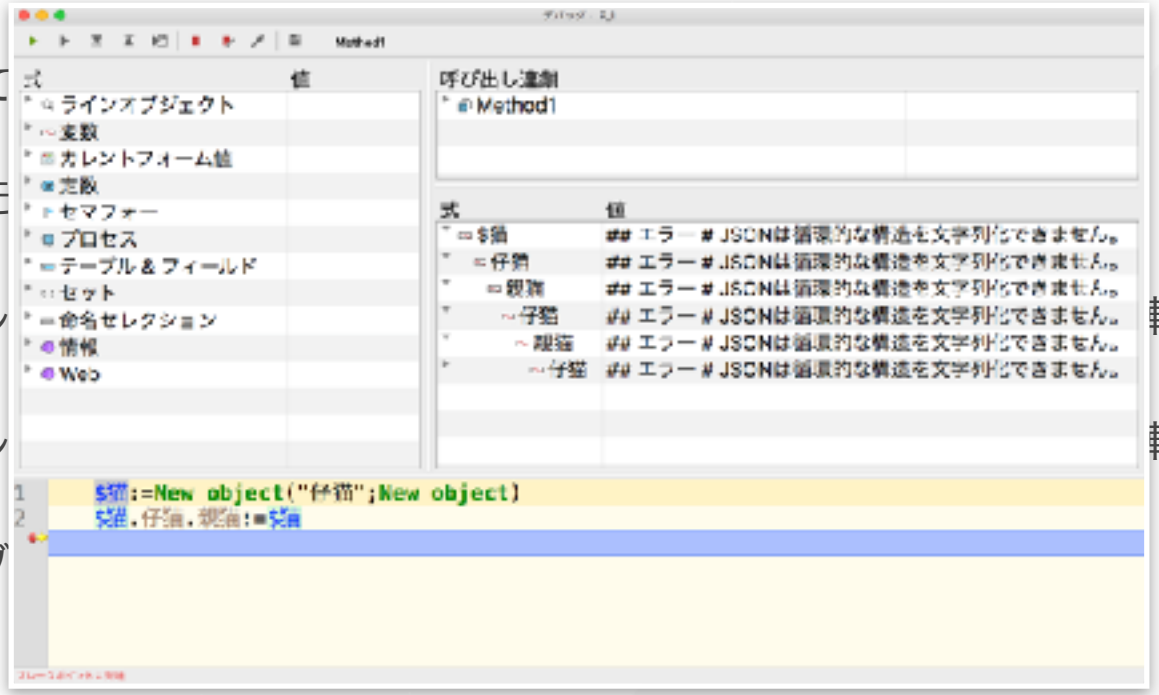
未定義を怖がらなくても良い理由
Don’t be Afraid of Undefined Values

https://blog.4d.com/object-notation-improvement-after-customer-feedback/

#4DSummit

循環オブジェクト

- それ自体を直接的あるいは間接的に参照しているオブジェクト
- 文字列として
- 確保したメモ
- **v16**: フィール 転送は**できない**
- **v17**: フィール 転送が**できる**
- **v17**: 同じオブ



バリアント型としてのプロパティ

- Null や Undefined を含めてどんなタイプでも扱うことができる
- タイプを気にせずにコードが記述できる

```
$0.value:=$1.value+$2.value
$0->:=$1->+$2->
```

- タイプチェックをランタイムまで先延ばしできる
- 実行速度と堅牢性の点では劣っているが知っていると便利

Non-JSON タイプ

- **BLOB**: 利用不可
- **Integer**: 利用不可
- **Date**: 日付 **16R6** **Stringify:**
- **Time**: 秒数 **v17**

!2018-10-22!

JS String type without time zone:0 "2018-10-21T15:00:00.000Z"

String type with time zone:1 "2018-10-22T00:00:00.000Z"

Date type:2 "2018-10-22"

- **Pointer**: オブジェクト **Stringify:** ポインターの値
- **Image**: オブジェクト **Stringify: [object Picture]**
- **ORDA**: オブジェクト **Stringify: [object DataStore] [object Entity] [object EntityCollection]**
- **Write Pro** および **View Pro**: オブジェクト **Stringify:** ドキュメントのプロパティ

JSON スキーマ

- **JSON Parse**は構文（シンタックス）を検証する
- **JSON Validate**は文法（グラマー）を検証する
- 外部ドキュメントを取り込む前のエラーチェックに有効
- **DTD XSD SET ARRAY/OB GET ARRAY** よりも扱いやすい

C_COLLECTION

- そのまま受け取ったり返したりすることができる配列のようなもの
- 要素の型は不揃いであっても構わない
- サイズを超えた位置に要素が追加されたときは自動的にリサイズ
- **OB SET ARRAY/OB GET ARRAY** がすでに使用していた

C_COLLECTION

	配列	コレクション
宣言と定義	`ARRAY TEXT($arr;n)`	`C_COLLECTION($col)` `$col:=New collection`
タイプ	一律	混在
代入	`$arr{$i}:="hello"`	`$col[$i]:="hello"`
サイズ	`Size of array($arr)`	`$col.length`
添字	0 または 1 から Size of array まで	0 から length-1 まで

コレクションメソッド

average	clear	combine	concat	copy	count
countValues	distinct	equal	every	extract	fill
filter	find	findIndex	indexOf	indices	insert
join	lastIndexOf	map	max	min	orderBy
orderByMethod	pop	push	query	reduce	remove
resize	reverse	shift	slice	some	sort
sum	unshift				

JS Inspired

コレクションメソッド

- 追加: push insert unshift resize fill
- 削除: pop shift remove clear
- 並び替え: orderBy orderByMethod sort reverse
- 計算: countValues count sum average min max reduce
- 比較: equal
- 複製/挿入: copy concat combine slice distinct
- 検索: indexOf find filter findIndex lastIndexOf query indices every some
- 取り出し: extract map
- 文字列変換: Split string join

JS Inspired

#4DSummit

大幅に拡張されたコレクション管理メソッド

A Wide Range of Possibilities to Manage your Collections

https://blog.4d.com/a-wide-range-of-possibilities-to-manage-your-collections/

#4DSummit

For each

➜コレクションの各要素に対して繰り返す

```
For ($i;0;$collection.length-1)
 MESSAGE($collection[$i])
End for
```

For each

➜コレクションの各要素に対して繰り返す

```
For ($i;0;$collection.length-1)
 MESSAGE($collection[$i])
End for

For each ($element;$collection)
 MESSAGE($element)
End for each
```

For each

➡コレクションの各要素に対して繰り返す

➡オブジェクトの各プロパティに対して繰り返す

```
For each ($propertyName;$object)
 MESSAGE($propertyName)
End for each
```

For each

➡コレクションの各要素に対して繰り返す

➡オブジェクトの各プロパティに対して繰り返す

➡セレクションの各エンティティに対して繰り返す

```
For each ($entity;$entitySelection)
 MESSAGE($entity.name)
End for each
```

For each

➡コレクションの各要素に対して繰り返す

➡オブジェクトの各プロパティに対して繰り返す

➡セレクションの各エンティティに対して繰り返す

```
For each ($element;$collection) While($element#"")
 MESSAGE($element)
End for each
```

For each

➡コレクションの各要素に対して繰り返す

➡オブジェクトの各プロパティに対して繰り返す

➡セレクションの各エンティティに対して繰り返す

```
For each ($element;$collection) Until($element="")
 MESSAGE($element)
End for each
```