



# プロトタイプ型オブジェクト指向 JavaScriptの利点を Webアプリケーションに活かす

群馬大学医学部附属病院  
システム統合センター  
鳥飼 幸太

Gunma University Hospital SIC, Confidential

## 目次

- 1 : なぜ「今」JavaScriptなのか
- 2 : プロトタイプオブジェクト指向とは
- 3 : M言語との共通性
- 4 : FHIR応用の利点
- 5 : JSFiddleの紹介
- 6 : ハンズオン : JSFiddle上でのWebアプリ製作と動作確認
  - 6.1: JSFiddleの使い方
  - 6.2: 基本的なHTMLコードの記述
  - 6.3: JavaScriptの呼出し
  - 6.4: オンラインライブラリの呼出し
  - 6.5: FHIR exampleからひな形を構成する
  - 6.6: Graph.jsのデータ構造
  - 6.7: Graph.jsでデータを取得する
  - 6.8: jExcelのデータ構造
  - 6.9: jExcelでデータを表示する

<https://jsfiddle.net/5fh3o06s/45/>

Gunma University Hospital SIC, Confidential

# なぜ「今」JavaScriptなのか



1995年にWebブラウザであったネットスケープ内のスクリプト言語として誕生  
1997年の標準化開始から、近年言語としての品質が急速に高まった

| 西暦   | 出来事                              |
|------|----------------------------------|
| 1990 | WWW誕生                            |
| 1994 | Mosaic Communication創業           |
| 1995 | JavaScriptが搭載される                 |
| 1998 | Google創業                         |
| 2004 | Facebook創業                       |
| 2009 | Node.jsが開発される (JSONサポート)         |
| 2010 | OracleがSun Microsystemsを買収       |
| 2015 | ECMAScript2015が制定<br>(クラス・モジュール) |



名前は同じだが、その内容が急速に変化している、5年前に制定された新しい技術

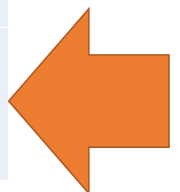
Gunma University Hospital SIC, Confidential

## プロトタイプ型オブジェクト指向（ECMA2015で策定）とは



- ・「クラスの管理者がいなくなった後のコード開発問題」に解決法を与えるもの

| クラスベース (Java) とプロトタイプベース (JavaScript) のオブジェクトシステムの比較              |                                                                                                           |
|-------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| クラスベース (Java)                                                     | プロトタイプベース (JavaScript)                                                                                    |
| クラスとインスタンスは異なる実体です。                                               | すべてのオブジェクトは別のオブジェクトを継承できます。                                                                               |
| クラス定義を用いてクラスを定義します。また、コンストラクターメソッドを用いてクラスをインスタンス化します。             | コンストラクター関数を用いて一連のオブジェクトを定義および作成します。                                                                       |
| new 演算子を用いて単一のオブジェクトを生成します。                                       | 同様です。                                                                                                     |
| 既存のクラスのサブクラスを定義するクラス定義を用いて、オブジェクト階層を構築します。                        | コンストラクター関数に結びつけられたプロトタイプとしてオブジェクトを代入することで、オブジェクト階層を構築します。                                                 |
| クラスチェーンに従ってプロパティを継承します。                                           | プロトタイプチェーンに従ってプロパティを継承します。                                                                                |
| クラス定義が、クラスから作られた全インスタンスすべてのプロパティを定義します。実行時に動的にプロパティを追加することはできません。 | コンストラクター関数またはプロトタイプによって、一連の初期化されたプロパティが指定されます。個々のオブジェクトやオブジェクトのセット全体へ動的にプロパティを追加したり、それらからプロパティを削除したりできます。 |



[https://developer.mozilla.org/ja/docs/Web/JavaScript/Guide/Details\\_of\\_the\\_Object\\_Model](https://developer.mozilla.org/ja/docs/Web/JavaScript/Guide/Details_of_the_Object_Model)

Gunma University Hospital SIC, Confidential

## M言語(IRIS)との共通性



### JSON リテラル・コンストラクタの使用

ダイナミック・エンティティは、`DynamicObject` または `DynamicArray` のインスタンスです。これらのクラスは、JSON のデータ操作を Cache にシームレスに統合するように設計されています。標準の `%New()` メソッドを使用してこれらのクラスのインスタンスを作成できますが、ダイナミック・エンティティは、これよりさらに柔軟で直感的な一連のコンストラクタをサポートしています。`JSON リテラル・コンストラクタ` を使用すると、JSON 文字列を変数に直接代入することでダイナミック・エンティティを作成できます。例えば、以下のコードは `DynamicObject` と `DynamicArray` の空のインスタンスを作成します。

```
set dynamicObject = {}
set dynamicArray = []
write dynamicObject.!.dynamicArray

3@%L library.DynamicObject
1@%L library.DynamicArray
```

`%New()` コンストラクタとは異なり、リテラル・コンストラクタの `{}` と `[]` は JSON フォーマットの文字列を引数として受け取ることができます。例えば、以下のコードは `prop1` という名前のプロパティを持つダイナミック・オブジェクトを作成します。

```
set dynamicObject = {"prop1":"a string value"}
write dynamicObject.prop1

a string value
```

実際には、JSON リテラル・コンストラクタの `{}` と `[]` を使用して、任意の有効な JSON 記列構造またはオブジェクト構造を指定できます。簡単に言うと、有効な JSON リテラル文字列はすべて、評価結果がダイナミック・エンティティとなる有効な Cache 式でもあります。

### ・IRISのDynamicObjectは JSONのプロトタイプ型オブジェクト指向そのものである

## オブジェクトとプロパティ

JavaScript のオブジェクトは、自身に関連付けられたプロパティを持ちます。オブジェクトのプロパティは、オブジェクトに関連付けられている変数と捉えることができます。オブジェクトのプロパティは、オブジェクトに属するものという点を除けば、基本的に通常の JavaScript 変数と同じようなものです。オブジェクトのプロパティは、オブジェクトの特性を定義します。オブジェクトのプロパティには、単純なドット表記でアクセスします。

```
1 | objectName.propertyName
```

すべての JavaScript の変数と同じく、オブジェクト名 (通常の変数にもなります) とプロパティ名では、大文字と小文字は厳密に区別されます。プロパティに値を代入することでプロパティを定義する事ができます。以下のようにして、`myCar` という名前のオブジェクトを作成し、`make`、`model`、`year` という名前のプロパティを付与することができます

```
1 | var myCar = new Object();
2 | myCar.make = 'Ford';
3 | myCar.model = 'Mustang';
4 | myCar.year = 1969;
```

上記の例は、**オブジェクト初期化子**を使用して作成することもできます。オブジェクト初期化子は、中括弧 (`{}`) で囲まれたオブジェクトのプロパティ名と関連する値の 0 個以上のペアをカンマで区切ったリストです。

```
1 | var myCar = {
2 |   make: 'Ford',
3 |   model: 'Mustang',
4 |   year: 1969
5 | };
```

オブジェクトに割り当てられていないプロパティは `undefined` です (`null` ではありません)。

Gunma University Hospital SIC, Confidential

## プログラミングの構成要素



<https://skillhub.jp/blogs/183>

ほぼ全てのプログラムが備えている機能：

変数名を付けた変数（数字や文字を格納）

関数（特定の処理を関数名として呼ぶ）

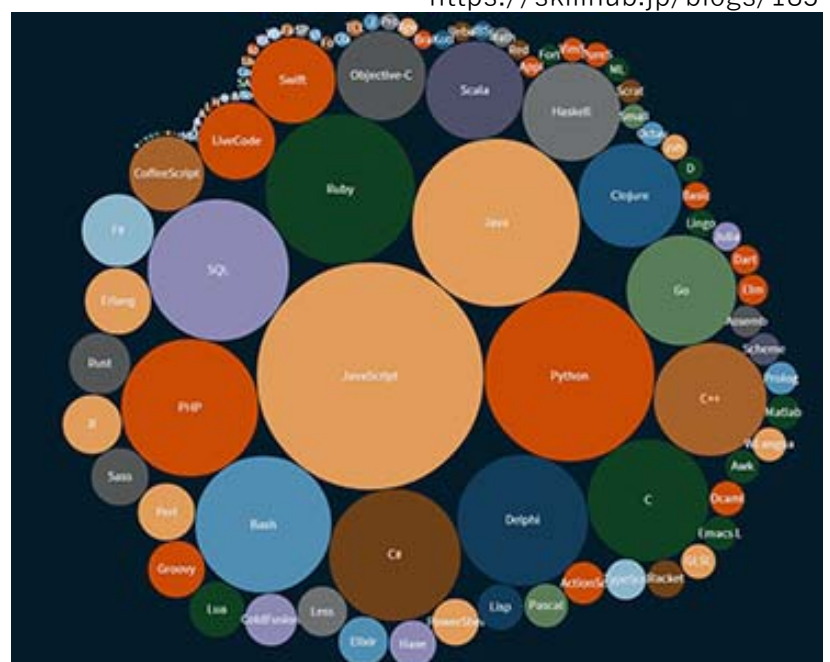
繰り返し（1億回でも繰り返せる）

条件分岐（場合分けの処理を記述する）

入出力（画面、ファイルシステム、デバイス）

外部呼出し（APIと呼ばれる）

素朴な疑問：多言語は真に必須なのか？



プログラム名とその比率

Gunma University Hospital SIC, Confidential

# Web開発の問題点



最小構成のプログラムでも「Webサーバ」を意識しなければならない

- ・非同期通信
- ・ポートの概念
- ・URIの概念

ユーザーインターフェースにHTML、CSSを必要とする

- ・JavaScriptを含め文法が3つ存在する

サーバサイドJavaエンジニアにとってはこれまで（品質の低い）言語であり、習得コストも高かった

これらの特徴のため、RAD（Rapid Application Development）には向かなかった

Gunma University Hospital SIC, Confidential

# Web開発の利点



世界中のサーバに置いてあるライブラリを利用できる

スクリプト言語と標準化がもたらす高い移植性

処理速度の遅さをハードウェアやメニーコアがカバーし、実用的に

クラス概念などの言語の洗練により、チームコーディングが容易に

→エンジニアは「世界中で使われるjsライブラリの開発者」となる道と、  
「これらライブラリを組み合わせる現場に必要なサービスアプリケーションを  
迅速に提供する開発者」となる道とが生じてきた



Gunma University Hospital SIC, Confidential



## FHIR-JSONを応用する直接的な利点



- ・ Webページで公開されている例を直接プロトタイプとして使うことができる  
→DICOM Conformance Statementのように、コードと「別に」仕様書を（エンジニア自身が）記述する仕事から解放される
- 仮に仕様が合わなくても、現地レベルで（つまり、コード生成の後ろの段で）修正モジュールを記述して通せばよい→作業性の改善

Gunma University Hospital SIC, Confidential

## FHIRサイトでの一例



The screenshot shows the HL7 FHIR website. The top navigation bar includes links for Home, Getting Started, Documentation, Resources, Profiles, Extensions, Operations, and Terminologies. The main content area is titled 'Welcome to FHIR®' and includes a 'First time here?' section with links to executive summary, developer's introduction, clinical introduction, and architect's introduction. Below this is a 'Technical Corrections' section. The 'Level 1 Basic framework' section includes a 'Foundation' box with links to Base Documentation, XML, JSON, Data Types, and Extensions. The 'Level 2 Supporting implementation and binding to external specifications' section includes four boxes: 'Implementer Support' (Downloads, Version Mgmt, Use Cases, Testing), 'Security & Privacy' (Security, Consent, Provenance, AuditEvent), 'Conformance' (StructureDefinition, CapabilityStatement, ImplementationGuide, Profiling), and 'Terminology' (CodeSystem, ValueSet, ConceptMap, Terminology Svc).

The diagram shows a 'Diagnostics' section with a red box around the word 'Observation'. The text 'クリック' (Click) is written next to the red box. Below the red box, the text 'Observation, Report, Specimen, ImagingStudy, Genomics, Specimen, ImagingStudy, etc.' is listed.

Gunma University Hospital SIC, Confidential

## Exampleを開く



The screenshot shows the HL7 FHIR website (https://www.hl7.org/fhir/observation.html) for Release 4. The 'Examples' tab is highlighted in the navigation bar. Below the navigation bar, there is a table with the following content:

| Orders and Observations Work Group                                                                                                        | Maturity Level: N | Normative (from v4.0.0) | Security Category: Patient | Compartments: Device, Encounter, Patient, Practitioner, RelatedPerson |
|-------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------------------|----------------------------|-----------------------------------------------------------------------|
| This page has been approved as part of an <a href="#">ANSI</a> standard. See the <a href="#">Observation Package</a> for further details. |                   |                         |                            |                                                                       |

Measurements and simple assertions made about a patient, device or other subject.

### 10.1.1 Scope and Usage

This resource is an [event resource](#) from a FHIR workflow perspective - see [Workflow](#).

Observations are a central element in healthcare, used to support diagnosis, monitor progress, determine baselines and patterns and even capture demographic characteristics. Most observations are simple name/value pair assertions with some metadata, but some observations group other observations together logically, or even are multi-component observations. Note that the [DiagnosticReport](#) resource provides a clinical or workflow context for a set of observations and the Observation resource is referenced by DiagnosticReport to represent laboratory, imaging, and other clinical and diagnostic data to form a complete report.

Gunma University Hospital SIC, Confidential

## 使いたい形式に近いExampleを選択



The screenshot shows the HL7 FHIR website (https://www.hl7.org/fhir/observation-examples.html) for Release 4. The 'Examples' tab is highlighted in the navigation bar. Below the navigation bar, there is a table with the following content:

| Orders and Observations Work Group                                   | Maturity Level: N/A | Standards Status: Informative | Security Category: Patient | Compartments: Device, Encounter, Patient, Practitioner, RelatedPerson |
|----------------------------------------------------------------------|---------------------|-------------------------------|----------------------------|-----------------------------------------------------------------------|
| The following Observations show examples of laboratory observations: |                     |                               |                            |                                                                       |

- Real-world patient - glucose
- Real-world patient - erythrocyte
- Real-world patient - hemoglobin
- Real-world patient - base excess
- Real-world patient - CO2
- Real-world patient - creatinine
- Real-world patient - estimated gfr
- Real-world patient - staphylococcus
- Specimen Reject Example
- An "ask at order" response using the datatype for the result value
- Simple blood typing example:

Gunma University Hospital SIC, Confidential

## JSON形式を選択



← → ↻ 🔍 <https://www.hl7.org/fhir/observation-example-f001-glucose.html>

**HL7 FHIR** Release 4

Home Getting Started Documentation Resources Profiles Extensions Operations Terminologies

Diagnostics > Observation > Example Instance

This page is part of the FHIR Specification (v4.0.1: R4 - Mixed Normative and STU). This is the current published version. For a full list of available versions, see the [Directory of published versions](#)

### Observation-example-f001-glucose

|                                    |                     |                               |                                                                       |
|------------------------------------|---------------------|-------------------------------|-----------------------------------------------------------------------|
| Orders and Observations Work Group | Maturity Level: N/A | Standards Status: Informative | Compartments: Device, Encounter, Patient, Practitioner, RelatedPerson |
|------------------------------------|---------------------|-------------------------------|-----------------------------------------------------------------------|

This is the narrative for the resource. See also the [XML](#), [JSON](#) or [turtle](#) format. This example conforms to the [profile Observation](#).

**Generated Narrative with Details**

id: f001

identifier: 6323 (OFFICIAL)

status: final

code: Glucose [Moles/volume] in Blood (Details : {LOINC code '15074-8' = 'Glucose [Moles/volume] in Blood', given as 'Glucose [Moles/volume] in Blood'})

subject: P. van de Heuvel

effective: 02/04/2013 9:30:10 AM --> (ongoing)

issued: 03/04/2013 3:30:10 PM

performer: A. Langeveld

value: 6.3 mmol/l (Details: UCUM code mmol/L = 'mmol/L')

interpretation: High (Details : {http://terminology.hl7.org/CodeSystem/v3-ObservationInterpretation code 'H' = 'High', given as 'High'})

クリック

Gunma University Hospital SIC, Confidential

## JSON部分をコピーする



← → ↻ 🔍 <https://www.hl7.org/fhir/observation-example-f001-glucose.json.html>

**HL7 FHIR** Release 4

Home Getting Started Documentation Resources Profiles Extensions Operations Terminologies

Diagnostics > Observation > Example Instance

This page is part of the FHIR Specification (v4.0.1: R4 - Mixed Normative and STU). This is the current published version. For a full list of available versions, see the [Directory of published versions](#)

### Observation-example-f001-glucose.json

|                                    |                     |                               |                                                                       |
|------------------------------------|---------------------|-------------------------------|-----------------------------------------------------------------------|
| Orders and Observations Work Group | Maturity Level: N/A | Standards Status: Informative | Compartments: Device, Encounter, Patient, Practitioner, RelatedPerson |
|------------------------------------|---------------------|-------------------------------|-----------------------------------------------------------------------|

Raw JSON (canonical form + also see [JSON Format Specification](#))

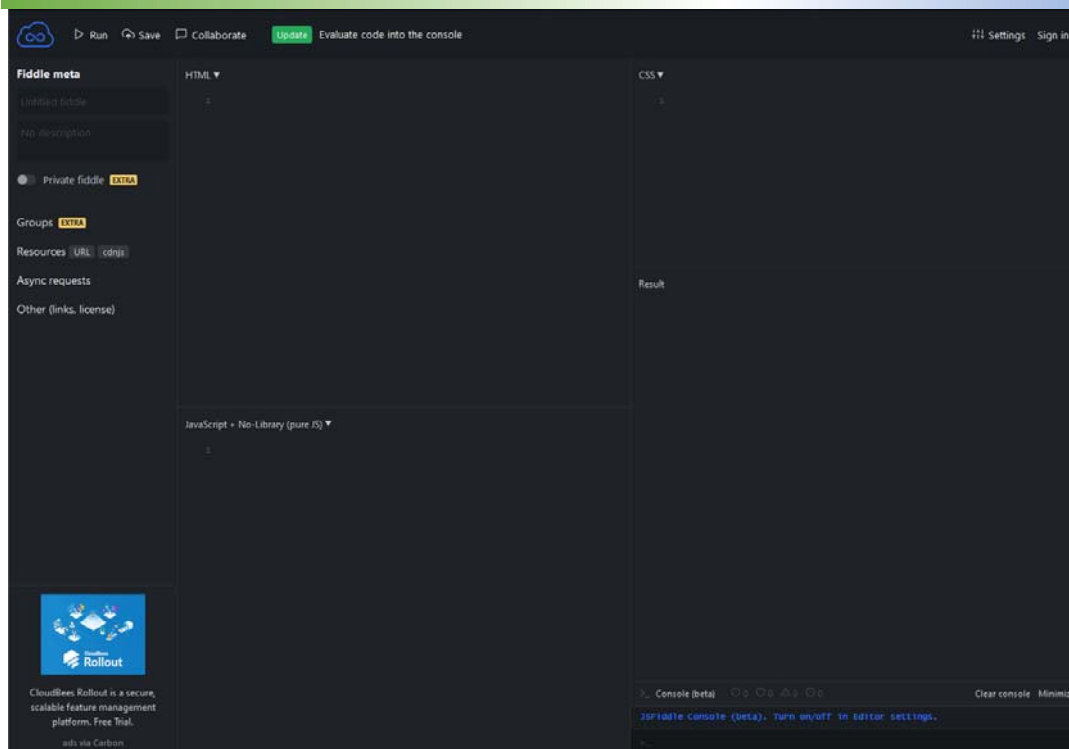
Real-world patient - glucose

```
{
  "resourceType": "Observation",
  "id": "f001",
  "text": {
    "status": "generated",
    "div": "<div xmlns='http://www.w3.org/1999/xhtml'><p><b>Generated Narrative with Details</b></p><p><b>id</b>: f001</p><p><b>identifier</b>: 6323 (OFFICIAL)</p><p><b>status</b>: final</p><p><b>code</b>: Glucose [Moles/volume] in Blood (Details : {LOINC code '15074-8' = 'Glucose [Moles/volume] in Blood', given as 'Glucose [Moles/volume] in Blood'})</p><p><b>subject</b>: <a>P. van de Heuvel</a></p><p><b>effective</b>: 02/04/2013 9:30:10 AM --> (ongoing)</p><p><b>issued</b>: 03/04/2013 3:30:10 PM</p><p><b>performer</b>: <a>A. Langeveld</a></p><p><b>value</b>: 6.3 mmol/l (Details: UCUM code mmol/L = 'mmol/L')</p><p><b>interpretation</b>: High (Details : {http://terminology.hl7.org/CodeSystem/v3-ObservationInterpretation code 'H' = 'High', given as 'High'})</p><p><b>ReferenceRanges</b>: <table><tr><td><b>Low</b></td><td><b>High</b></td><tr><td><b>3.1 mmol/l</b></td><td><b>5.6 mmol/l</b></td></tr></table></p></div>"
  },
  "identifier": [
    {
      "use": "official",
      "system": "http://www.bmc.nl/zorgportal/identifiers/observations",
      "value": "6323"
    }
  ]
}
```

コピー

Gunma University Hospital SIC, Confidential

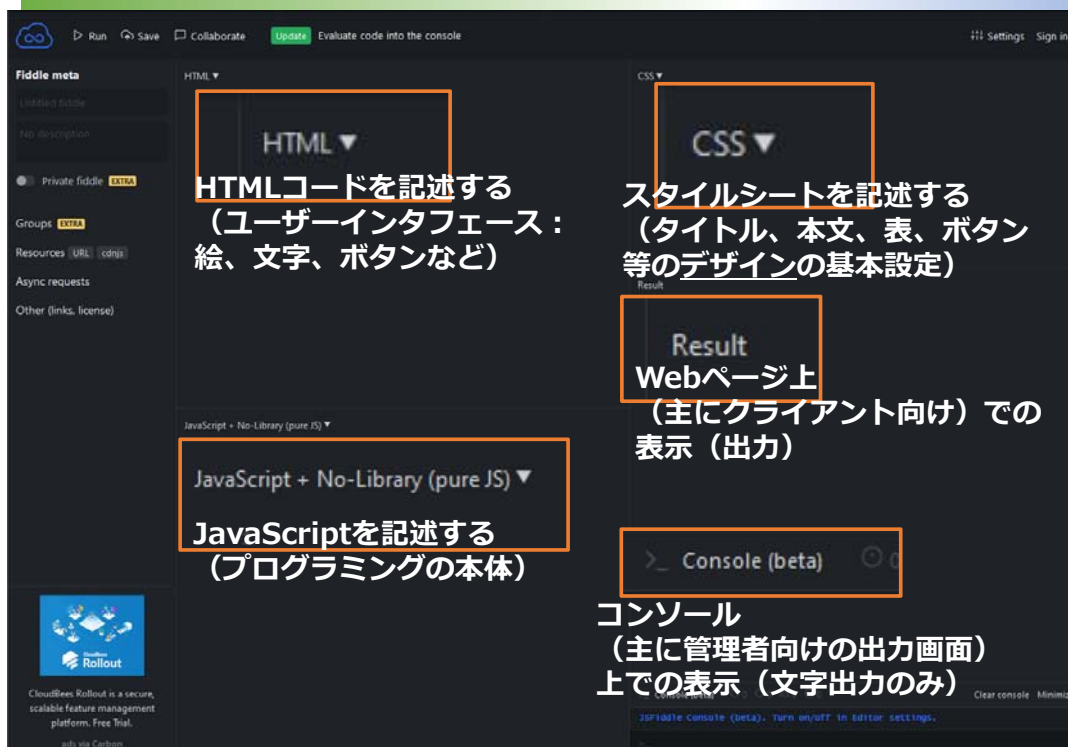
# JSFiddleの起動



<https://jsfiddle.net/>  
利用にあたり  
アカウント作成や  
ログインは不要

uma University Hospital SIC, Confidential

## JSFiddle: 画面の使い方



uma University Hospital SIC, Confidential



## JavaScript部にコピーしたデータを貼り付けて変数化



```
JavaScript + No-Library (pure JS) ▼
1 var fhirdata = {
2   "resourceType": "Observation",
3   "id": "f001",
4   "text": {
5     "status": "generated",
6     "div": "<div xmlns=\\"http://www.w3.org/1999/xhtml\\"><p><b>Generated Narrative with Details</b></p><p><b>id</b>: f001</p><p><b>identifier</b>: 6323 (OFFICIAL)</p><p><b>status</b>: final</p><p><b>code</b>: Glucose [Moles/volume] in Blood <span>(Details : {LOINC code '15074-8' = 'Glucose [Moles/volume] in Blood', given as 'Glucose [Moles/volume] in Blood'})</span></p><p><b>subject</b>: <a>P. van de Heuvel</a></p><p><b>effective</b>: 02/04/2013 9:30:10 AM --&gt; (ongoing)</p><p><b>issued</b>: 03/04/2013 3:30:10 PM</p><p><b>performer</b>: <a>A. Langeveld</a></p><p><b>value</b>: 6.3 mmol/l<span> (Details: UCUM code mmol/L = 'mmol/L')</span></p><p><b>interpretation</b>: High <span>(Details : {http://terminology.hl7.org/CodeSystem/v3-ObservationInterpretation code 'H' = 'High', given as 'High'})</span></p><h3>ReferenceRanges</h3><table><tr><td><b>Low</b></td><td><b>High</b></td></tr><tr><td>*</td><td>3.1 mmol/l<span> (Details: UCUM code mmol/L = 'mmol/L')</span></td><td>6.2 mmol/l<span> (Details: UCUM code mmol/L = 'mmol/L')</span></td></tr></table></div>"
7 }
>_ Console (beta) 0 0 0 0
```

var fhirdata = の後にペーストし、最後の}の後に ; を記述すれば「ひな形」のデータが出来上がる

Gunma University Hospital SIC, Confidential

## プロパティの表示



```
JavaScript + No-Library (pure JS) ▼
1 var fhirdata = {
2   "resourceType": "Observation",
3   "id": "f001",
4   "text": {
5     "status": "generated",
6     "div": "<div xmlns=\\"http://www.w3.org/1999/xhtml\\"><p><b>Generated Narrative with Details</b></p><p><b>id</b>: f001</p><p><b>identifier</b>: 6323 (OFFICIAL)</p><p><b>status</b>: final</p><p><b>code</b>: Glucose [Moles/volume] in Blood <span>(Details : {LOINC code '15074-8' = 'Glucose [Moles/volume] in Blood', given as 'Glucose [Moles/volume] in Blood'})</span></p><p><b>subject</b>: <a>P. van de Heuvel</a></p><p><b>effective</b>: 02/04/2013 9:30:10 AM --&gt; (ongoing)</p><p><b>issued</b>: 03/04/2013 3:30:10 PM</p><p><b>performer</b>: <a>A. Langeveld</a></p><p><b>value</b>: 6.3 mmol/l<span> (Details: UCUM code mmol/L = 'mmol/L')</span></p><p><b>interpretation</b>: High <span>(Details : {http://terminology.hl7.org/CodeSystem/v3-ObservationInterpretation code 'H' = 'High', given as 'High'})</span></p><h3>ReferenceRanges</h3><table><tr><td><b>Low</b></td><td><b>High</b></td></tr><tr><td>*</td><td>3.1 mmol/l<span> (Details: UCUM code mmol/L = 'mmol/L')</span></td><td>6.2 mmol/l<span> (Details: UCUM code mmol/L = 'mmol/L')</span></td></tr></table></div>"
7 }
8 console.log(fhirdata.id);
>_ Console (beta) 0 0 0 0
```

https://jsfiddle.net/o6e9ya7g/1/

```
>_ Console (beta) 0 2 0 0 0 0 Clear console Minimize
JSFiddle Console (beta). Turn on/off in Editor settings.
"Running fiddle"
"Running fiddle"
"f001"
"Running fiddle"
"f001"
>_
```

コンソール出力関数は  
console.log(...);  
ここに、Exampleに入っているidを呼び出すと、f001が表示される

Gunma University Hospital SIC, Confidential

## プロパティを書き換える



```
JavaScript + No-Library (pure JS)
--
51     "display": "High"
52   }
53 }
54 ],
55 ],
56 "referenceRange": [
57   {
58     "low": {
59       "value": 3.1,
60       "unit": "mmol/l",
61       "system": "http://unitsofmeasure.org",
62       "code": "mmol/L"
63     },
64     "high": {
65       "value": 6.2,
66       "unit": "mmol/l",
67       "system": "http://unitsofmeasure.org",
68       "code": "mmol/L"
69     }
70   }
71 ];
72
73 fhirdata.id = "Chiba Tarou";
74 console.log(fhirdata.id);
75
```

https://jsfiddle.net/o6e9ya7g/2/

Console (beta) 3 0 0 0 0 Clear console Minimize

JSFiddle Console (beta). Turn on/off in Editor settings.

Running fiddle

Running fiddle

"f001"

Running fiddle

"f001"

Running fiddle

"Chiba Tarou"

idが上書きされる

Gunma University Hospital SIC, Confidential

## プロパティを動的に追加する（プロトタイプ型オブジェクトの特徴）



```
JavaScript + No-Library (pure JS)
--
54   }
55 ],
56 "referenceRange": [
57   {
58     "low": {
59       "value": 3.1,
60       "unit": "mmol/l",
61       "system": "http://unitsofmeasure.org",
62       "code": "mmol/L"
63     },
64     "high": {
65       "value": 6.2,
66       "unit": "mmol/l",
67       "system": "http://unitsofmeasure.org",
68       "code": "mmol/L"
69     }
70   }
71 ];
72 };
73
74 fhirdata.id = "Chiba Tarou";
75 console.log(fhirdata.id);
76 fhirdata.specialcomment = "Extremely Good";
77 console.log(fhirdata.specialcomment);
78
```

https://jsfiddle.net/o6e9ya7g/3/

Console (beta) 5 0 0 0 0 Clear console Minimize

JSFiddle Console (beta). Turn on/off in Editor settings.

Running fiddle

Running fiddle

"f001"

Running fiddle

"f001"

Running fiddle

"Chiba Tarou"

Running fiddle

"Chiba Tarou"

"Extremely Good"

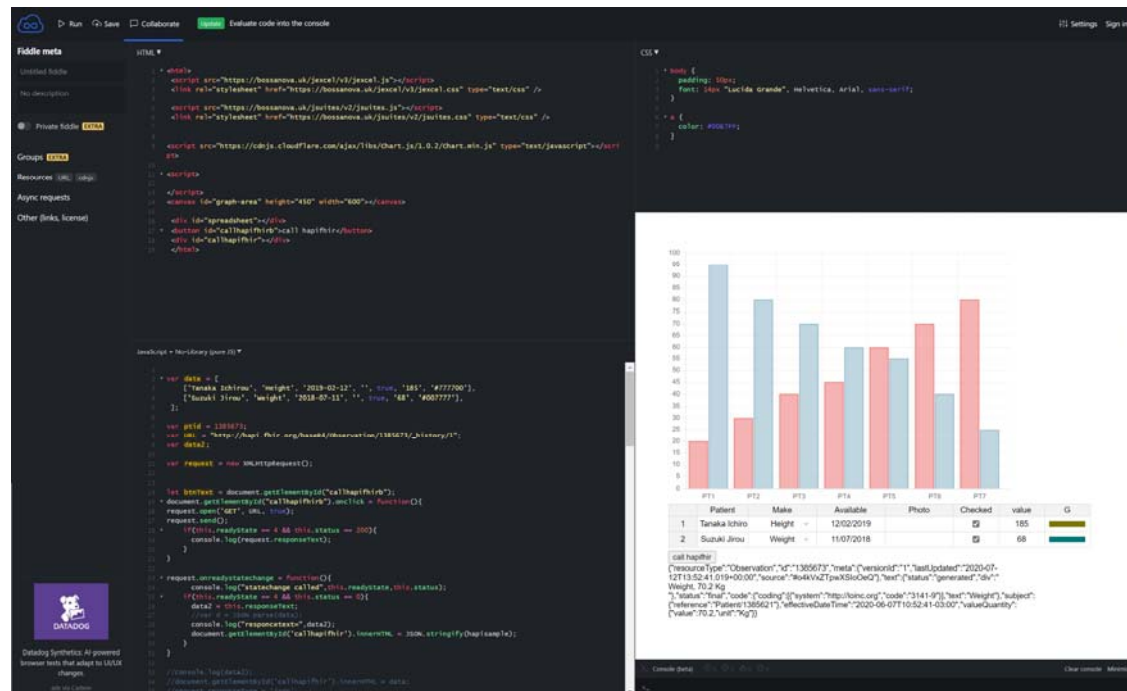
それまでに定義していなかったプロパティを動的に格納・使用できる

Gunma University Hospital SIC, Confidential

# JSFiddleサンプル(Graph.js, jExcel)



<https://jsfiddle.net/5fh3o06s/45/>



## Graph.jsのデータ設定



JavaScript + No-Library (pure JS) ▼

```
37
38
39 // ▼上記のグラフを描画するための記述
40 window.onload = function(){
41     var ctx = document.getElementById("graph-area").getContext("2d");
42     window.myBar = new Chart(ctx).Bar(barChartData);
43 }
125
126 // ▼グラフの中央
127 var barChartData = {
128     labels: ["PT1", "PT2", "PT3", "PT4", "PT5", "PT6", "PT7"],
129     datasets: [
130         {
131             fillColor: "rgba(240,128,128,0.6)", // 塗りつぶし色
132             strokeColor: "rgba(240,128,128,0.9)", // 枠線の色
133             highlightFill: "rgba(255,64,64,0.75)", // マウスが載った際の塗りつぶし色
134             highlightStroke: "rgba(255,64,64,1)", // マウスが載った際の枠線の色
135             data: [ 20, 30, 40, 45, 60, 70, 80 ] // 各棒の値(カンマ区切りで指定)
136         },
137         {
138             fillColor: "rgba(151,187,205,0.6)",
139             strokeColor: "rgba(151,187,205,0.9)",
140             highlightFill: "rgba(64,96,255,0.75)",
141             highlightStroke: "rgba(64,96,255,1)",
142             data: [ 95, 80, 70, 60, 55, 40, 25 ]
143         }
144     ]
145 };
146
```

<https://jsfiddle.net/j92vhk4y/>

data:欄のデータを  
代入で書き換えて表示を行う

## Graph.jsのデータ設定



JavaScript + No-Library (pure JS) ▼

```
1 // ▼上記のグラフを描画するための記述
2 window.onload = function(){
3   var ctx = document.getElementById("graph-area").getContext("2d");
4   barChartData.datasets[0].data = [10, 15, 20, 25, 30, 35, 40];
5   window.myBar = new Chart(ctx).Bar(barChartData);
6
7 };
8
```

```
125
126 // ▼グラフの中身
127 var barChartData = {
128   labels : ["PT1", "PT2", "PT3", "PT4", "PT5", "PT6", "PT7"],
129   datasets : [
130     {
131       fillColor : "rgba(240,128,128,0.6)", // 塗りつぶし色
132       strokeColor : "rgba(240,128,128,0.9)", // 枠線の色
133       highlightFill : "rgba(255,64,64,0.75)", // マウスが載った際の塗りつぶし色
134       highlightStroke : "rgba(255,64,64,1)", // マウスが載った際の枠線の色
135       data : [ 20, 30, 40, 45, 60, 70, 80 ] // 各棒の値(カンマ区切りで指定)
136     },
137     {
138       fillColor : "rgba(151,187,205,0.6)",
139       strokeColor : "rgba(151,187,205,0.9)",
140       highlightFill : "rgba(64,96,255,0.75)",
141       highlightStroke : "rgba(64,96,255,1)",
142       data : [ 95, 80, 70, 60, 55, 40, 25 ]
143     }
144   ]
145 };
146
```

<https://jsfiddle.net/j92vhk4y/1/>

**data:欄のデータを  
代入で書き換えて表示を行う**

## jExcelのデータ設定



HTML ▼

```
1 <html>
2 <script src="https://bossanova.uk/jexcel/v3/jexcel.js"></script>
3 <link rel="stylesheet" href="https://bossanova.uk/jexcel/v3/jexcel.css" type="text/css" />
4
5 <script src="https://bossanova.uk/jsuities/v2/jsuities.js"></script>
6 <link rel="stylesheet" href="https://bossanova.uk/jsuities/v2/jsuities.css" type="text/css" />
7
8
9 <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/1.0.2/Chart.min.js" type="text/javascript"></script>
10
11 <script>
12
13 </script>
14 <canvas id="graph-area" height="450" width="600"></canvas>
15
16 <div id="spreadsheet"></div>
17 <button id="callhapifhir">call hapifhir</button>
18 <div id="callhapifhir"></div>
19 </html>
```

<https://jsfiddle.net/j92vhk4y/2/>

**source:欄のデータを  
代入で書き換えて表示を行う  
この記述部がJSONファイルで読み込んでもよい**

```
jexcel(document.getElementById('spreadsheet'), {
  data:data,
  columns: [
    {
      type: 'text',
      title: 'Patient',
      width:90
    },
    {
      type: 'dropdown',
      title: 'Make',
      width:120,
      source:[
        "Temperature",
        "Height",
        "Weight",
        "BloodType",
        "Age",
        "Name"
      ]
    },
    {
      type: 'calendar',
      title: 'Available',
      width:120
    },
    {
      type: 'image',
      title: 'Photo',
      width:120
    },
    {
      type: 'checkbox',
      title: 'checked',
      width:80
    },
    {
      type: 'numeric',
      title: 'value',
      mask: '#,##,00',
      width:80,
      decimal: ','
    },
    {
      type: 'color',
      width:80,
      render: 'square'
    }
  ]
});
```



## ボタンの配置



```
HTML ▼
1 <html>
2 <script src="https://bossanova.uk/jexcel/v3/jexcel.js"></script>
3 <link rel="stylesheet" href="https://bossanova.uk/jexcel/v3/jexcel.css" type="text/css" />
4
5 <script src="https://bossanova.uk/jsuities/v2/jsuities.js"></script>
6 <link rel="stylesheet" href="https://bossanova.uk/jsuities/v2/jsuities.css" type="text/css" />
7
8
9 <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/1.0.2/chart.min.js" type="text/javascript"></script>
10
11 <script>
12
13 </script>
14 <canvas id="graph-area" height="450" width="600"></canvas>
15
16 <div id="spreadsheet"></div>
17 <button id="callhapifhir">call hapifhir</button>
18 <div id="callhapifhir"></div>
19 </html>
```

```
13
14 let btnText = document.getElementById("callhapifhirb");
15 document.getElementById("callhapifhirb").onclick = function(){
16 request.open('GET', URL, true);
17 request.send();
18 if(this.readyState == 4 && this.status == 200){
19 console.log(request.responseText);
20 }
21 }
```

<button>タグでidを置く  
JavaScript中でdocument.getElementByIdでidをコール

Gunma University Hospital SIC, Confidential

## FHIRサーバへの接続



```
7 var ptid = 1385673;
8 var URL = "http://hapi.fhir.org/baseR4/observation/1385673/_history/1";
9 var data2;
10
11 var request = new XMLHttpRequest();
12
13
14 let btnText = document.getElementById("callhapifhirb");
15 document.getElementById("callhapifhirb").onclick = function(){
16 request.open('GET', URL, true);
17 request.send();
18 if(this.readyState == 4 && this.status == 0){
19 console.log(request.responseText);
20 }
21 }
22
23 request.onreadystatechange = function(){
24 console.log("statechange called",this.readyState,this.status);
25 if(this.readyState == 4 && this.status == 0){
26 data2 = this.responseText;
27 //var d = JSON.parse(data);
28 console.log("responsetext=",data2);
29 document.getElementById('callhapifhir').innerHTML = JSON.stringify(hapisample);
30 }
31 }
32 }
```

<https://jsfiddle.net/5fh3o06s/45/>

XMLHttpRequest()でURLをコールする  
request.sendでHAPI serverにアクセスする

Gunma University Hospital SIC, Confidential



# 1 : サーバアクセスの基本となるフレームHTMLを作成する

## HTMLの構造

```
<html>
<!-- ①HTMLのスタイル等設定箇所 -->
<head>
  <meta charset="UTF-8">
  <title>FHIRTEST Step 1</title>
  <style>body {
    background: #eeeeee;
  }</style>
<!-- ②サーバに接続しFHIRデータを処理する箇所 -->
<script type="text/javascript" language="javascript">
  function onClick() {
    var target = document.getElementById("output2");
    var txt = document.forms.id_form1.id_textBox2.value;
    var outtext = txt + "です";
    target.innerHTML = outtext;
  }
</script>
</head>
<br>
<br>
<!-- ③画面出力を行う箇所 -->
<body>
  <form name="form1" id="id_form1" action="">
    <input name="textBox1" id="id_textBox2" type="text" value="" />
    <input type="button" value="Exec" onclick="onClick();" />
  </form>
  <br>
  <div id="output2"></div>
<br>
</body>
</html> [EOF]
```

**HTMLタグ：一番外側にくる構造**

**HEADタグ：スタイル、コードを格納**

**SCRIPTタグ：コード実体を記述**

**BODYタグ：コードにより出力されるHTMLのターゲット位置を指定**

Gunma University Hospital SIC, Confidential

# 1 : サーバアクセスの基本となるフレームHTMLを作成する



## HTMLの構造

```
<html>
<!-- ①HTMLのスタイル等設定箇所 -->
<head>
  <meta charset="UTF-8">
  <title>FHIRTEST Step 1</title>
  <style>body {
    background: #eeeeee;
  }</style>
<!-- ②サーバに接続しFHIRデータを処理する箇所 -->
<script type="text/javascript" language="javascript">
  function onClick() {
    var target = document.getElementById("output2");
    var txt = document.forms.id_form1.id_textBox2.value;
    var outtext = txt + "です";
    target.innerHTML = outtext;
  }
</script>
</head>
<br>
<br>
<!-- ③画面出力を行う箇所 -->
<body>
  <form name="form1" id="id_form1" action="">
    <input name="textBox1" id="id_textBox2" type="text" value="" />
    <input type="button" value="Exec" onclick="onClick();" />
  </form>
  <br>
  <div id="output2"></div>
<br>
</body>
</html> [EOF]
```

**③関数onClick()の動作を記述  
ここではtextBox2の値を読み込み**

**①画面から入力されたテキストボックスの内容はJavaScriptからid\_textbox2として認識される**

**②ボタンクリックで、関数onClick()をコール**

**④document.getElementById("output2").innerHTMLを設定**

Gunma University Hospital SIC, Confidential

## 2 : JavaScriptでサーバにアクセスする



### サンプル

```
<html>
<head>
  <meta charset="UTF-8">
  <title>FHIRTEST Step 2</title>
  <style>body {
    background: #eeeeee;
  }</style>
  <script type="text/javascript" language="javascript">
    function onClick() {
      target = document.getElementById("output");

      var xhr = new XMLHttpRequest(); // HTTPサーバへの接続インスタンスを作成
      xhr.responseType = "text"; // サーバからの応答をテキスト形式に指定
      xhr.open("GET", "http://iris4h-fhirapi.japaneast.cloudapp.azure.com:52773/csp/sqltothir/fhir/observation/pulse/1001", true); // RESTのGETコマンドで、リソースを指定する
      xhr.send(); // サーバに送信

      xhr.onreadystatechange = function() { // サーバから応答があったときに実行される箇所
        if (xhr.readyState==4 && xhr.status==200) { // readyState=4(DONE (操作完了) かつstatus=200(OK (要求通りの処理がされた)
          var data = JSON.parse(xhr.responseText); // !!重要 JSON形式を構造体として解釈する これ以降はJSONタグをプロパティとして記述できる
          //追加した部分
          //変更した部分
          var outtext = xhr.responseText + '<br> patient id = ' + data.id + '<br>'; // data.idと記述されている"id"はJSON内のidタグを指す
          //変更した部分

          target.innerHTML = outtext; //読み取ったタグの値をHTMLとして出力
        }
      }
    }
  </script>
</head>
<body>
  <div id="output"></div>
</body>
</html>
```

- ①XMLHttpRequestのインスタンスを作成
- ②テキスト形式を指定
- ③GET/REST構文で欲しいリソースを指定
- ④構文をサーバに送信

読みだしたファイルの表示位置

## 3 : データを取得しJSON形式でメモリに格納する



### サンプル

```
<html>
<head>
  <meta charset="UTF-8">
  <title>FHIRTEST Step 2</title>
  <style>body {
    background: #eeeeee;
  }</style>
  <script type="text/javascript" language="javascript">
    function onClick() {
      target = document.getElementById("output");

      var xhr = new XMLHttpRequest(); // HTTPサーバへの接続インスタンスを作成
      xhr.responseType = "text"; // サーバからの応答をテキスト形式に指定
      xhr.open("GET", "http://iris4h-fhirapi.japaneast.cloudapp.azure.com:52773/csp/sqltothir/fhir/observation/pulse/1001", true); // RESTのGETコマンドで、リソースを指定する
      xhr.send(); // サーバに送信

      xhr.onreadystatechange = function() { // サーバから応答があったときに実行される箇所
        if (xhr.readyState==4 && xhr.status==200) { // readyState=4(DONE (操作完了) かつstatus=200(OK (要求通りの処理がされた)
          var data = JSON.parse(xhr.responseText); // !!重要 JSON形式を構造体として解釈する これ以降はJSONタグをプロパティとして記述できる
          //追加した部分
          //変更した部分
          var outtext = xhr.responseText + '<br> patient id = ' + data.id + '<br>'; // data.idと記述されている"id"はJSON内のidタグを指す
          //変更した部分

          target.innerHTML = outtext; //読み取ったタグの値をHTMLとして出力
        }
      }
    }
  </script>
</head>
<body>
  <div id="output"></div>
</body>
</html>
```

- ①XMLHttpRequestのインスタンスを作成
- ②テキスト形式を指定
- ③GET/REST構文で欲しいリソースを指定
- ④構文をサーバに送信

- ① 応答が200(正常)であることを条件とし割り込み実行
- ② 変数名dataに応答テキストをJSON形式でパース入力

①例えばJSONのラベル内に"id"が存在する場合、これをdata.idのようにプロパティとして呼び出せる  
→FHIRで医療情報のラベル名が固定されることで、コードの再利用が非常に楽になることに注目する

読み出したファイルの表示位置

## 4 : Graph.jsライブラリを参照する



Graph.jsの元コードでは、<body>内に<script>タグを置き、その中でChartインスタンスを生成しながらデータを割り当てている

```
<script>
var ctx = document.getElementById('myChart').getContext('2d');
var myChart = new Chart(ctx, {
  type: 'line',
  data: {
    labels: ['M', 'T', 'W', 'T', 'F', 'S', 'S'],
    datasets: [
      {
        label: 'apples',
        data: [12, 19, 3, 17, 6, 3, 7],
        backgroundColor: "rgba(153,255,51,0.4)"
      }, {
        label: 'oranges',
        data: [2, 29, 5, 5, 2, 3, 10],
        backgroundColor: "rgba(255,153,0,0.4)"
      }
    ]
  }
});
</script>
```

データの内容はJSON形式で記述されていることがわかる

→何らかの構造体データを作り、それをJSON化するか、またはJSONフォーマット自体を生成して自動化

## 4 : Graph.jsライブラリを参照する



```
function onShowChart() {
  var chartheader = document.createElement("script");
  chartheader.innerHTML = `var ctx = document.getElementById('myChart').getContext('2d');` + Chartの実体
  var myChart = new Chart(ctx, {
    type: 'line',
    data: {
      labels: ['M', 'T', 'W', 'T', 'F', 'S', 'S'],
      datasets: [
        {
          label: 'apples',
          data: [12, 19, 3, 17, 6, 3, 7],
          backgroundColor: "rgba(153,255,51,0.4)"
        }, {
          label: 'oranges',
          data: [2, 29, 5, 5, 2, 3, 10],
          backgroundColor: "rgba(255,153,0,0.4)"
        }
      ]
    }
  });
  target = document.getElementById("outchartsript");
  target.appendChild(chartheader);
}

</script>
</head>
<br>
<br>
<body>
  <form name="form1" id="id_form1" action="">
    <input name="textBox1" id="id_textBox1" type="text" value="" />
    <input type="button" value="Exec" onclick="onButtonClick();" />
  </form>
  <form name="form2" id="id_form2" action="">
    <input name="textBox2" id="id_textBox2" type="text" value="" />
    <input type="button" value="Exec" onclick="onShowChart();" />
  </form>
  <br>
  <div id="output"></div>
  <br>
  <a href="/contact.html" class="btn">これはボタンです。</a>
</body>

<body>
  <script src="https://cdn.jsdelivr.net/npm/chart.js@2.1.4/Chart.min.js"></script>
  <canvas id="myChart"></canvas>
  <div id="outchartsript"></div>
</body>
```

ボタンと実装例

Graphタグ : この中にGraph.jsの実体を記述する



## 4 : Graph.jsライブラリを参照する



### サンプル

```
function onShowChart(){
  var charthead = document.createElement("script");
  charthead.innerHTML = `
    var ctx = document.getElementById('myChart').getContext('2d');
    var myChart = new Chart(ctx, {
      type: 'line',
      data: {
        labels: ['M', 'T', 'W', 'T', 'F', 'S', 'S'],
        datasets: [
          {
            label: 'apples',
            data: [12, 19, 3, 17, 6, 3, 7],
            backgroundColor: 'rgba(153,255,51,0.4)'
          },
          {
            label: 'oranges',
            data: [2, 29, 5, 2, 3, 10],
            backgroundColor: 'rgba(255,153,0,0.4)'
          }
        ]
      }
    });
  `;
  target = document.getElementById("outchartscrip");
  target.appendChild(charthead);
}
```

出力自体がJavascriptとして動くことを指定

X-Yを指定した2次元グラフ

折れ線グラフ

X軸の要素ラベル

Y軸 1つ目の数値列  
表示色の指定 (黄緑)

Y軸 2つ目の数値列  
表示色の指定 (オレンジ)

出力自体がJavascriptとして記述されることを指定

## 5 : 条件式を用いて取り出す



JSON構造体を単純に読み出す関数 (JSON配列の再帰呼び出しのアルゴリズムを理解する好例)

```
function plotres(response, prefix) {
  for (var key in response) {
    if (typeof response[key] == "object") {
      if (Array.isArray(response[key])) {
        // 配列の場合は forEach で要素ごとに再帰呼び出し
        response[key].forEach(function(item) {
          plotres(item, prefix+" "+key);
        });
      } else {
        // 連想配列はそのまま再帰呼び出し
        plotres(response[key], prefix+" "+key);
      }
    } else {
      // 配列や連想配列でなければキーの値を表示
      console.log(prefix+" "+key+": "+response[key]);
    }
  }
}
```

## 6 : Graph.jsのデータ入力形式に合わせてデータフォーマットを作成する



```
function onClick() {  
    var linkText5 = "https://hapi.fhir.org/baseDstu3/Observation?patient=172385&pretty=true";  
    target = document.getElementById("output");  
  
    var xhr = new XMLHttpRequest();  
    xhr.responseType = 'text';  
    xhr.open('GET', linkText5, true);  
    xhr.send();  
  
    let insertLabels = '';  
    let insertData = '';  
  
    xhr.onreadystatechange = function() {  
        if (xhr.readyState==4 && xhr.status==200) {  
  
            var data = JSON.parse(xhr.responseText);  
            //要素のめわり data['entry'].length  
  
            var extractdata; ;  
            extractdata += '<br>';  
  
            for ( var i in data.entry ) {  
                if (data.entry[i].resource.code.text == 'Heart Rate'){  
                    insertLabels += "" + data.entry[i].resource.effectiveDateTime + "<br>";  
                    insertData += "" + data.entry[i].resource.valueQuantity.value + "<br>";  
                    if(i < (data['entry'].length - 1)){  
                        insertLabels += "<br>";  
                        insertData += "<br>";  
                    }  
                }  
                extractdata += data.entry[i].resource.id + "-" + data.entry[i].resource.code.text + "-" + data.entry[i].resource.valueQuantity.value + "<br>"; //['resource']['medicationCodeableConcept']['text']  
            }  
  
            var outtext = "<br> patient id = " + data.entry[0].resource.subject.reference + "<br> elements.No = " + data['entry'].length + "<br>" + extractdata + "<br>" + xhr.responseText;  
            target.innerHTML = outtext;  
  
            plotres(data, '');  
            var insertLabel; ;  
            var insertData; ;  
            //日付  
            insertLabel = "labels: [" + insertLabels + "],";  
            //数値  
            insertData = "data: [" + insertData + "],";  
  
            console.log(insertLabels);  
            console.log(insertData);  
  
            onShowChart2(insertLabel, insertData);  
            onShowChart3(insertLabel, insertData);  
        }  
    }  
};
```

**FHIRデータのcode.textから“Heart Rate”に該当する場合にその内容を読み込み  
→他のデータに応用可能**

## 7 : Graph.jsの表示フォーマットを整える



表示項目を追加する方法（このフォーマット自体をライブラリ化してもよい）

```
var extractdata; ;  
extractdata += '<br>';  
  
for ( var i in data.entry ) {  
    if (data.entry[i].resource.code.text == '体温'){  
        insertLabels += "" + data.entry[i].resource.effectiveDateTime + "<br>";  
        insertData += "" + data.entry[i].resource.valueQuantity.value + "<br>";  
        if(i < (data['entry'].length - 1)){  
            insertLabels += "<br>";  
            insertData += "<br>";  
        }  
    }  
    if (data.entry[i].resource.code.text == 'Respiratory Rate'){  
        insertLabels2 += "" + data.entry[i].resource.effectiveDateTime + "<br>";  
        insertData2 += "" + data.entry[i].resource.valueQuantity.value + "<br>";  
        if(i < (data['entry'].length - 1)){  
            insertLabels2 += "<br>";  
            insertData2 += "<br>";  
        }  
    }  
    extractdata += data.entry[i].resource.id + "-" + data.entry[i].resource.code.text + "-" + data.entry[i].resource.valueQuantity.value + "<br>"; //['resource']['medicationCodeableConcept']['text']  
};  
  
var outtext = "<br> patient id = " + data.entry[0].resource.subject.reference + "<br> elements.No = " + data['entry'].length + "<br>" + extractdata + "<br>" + xhr.responseText;  
target.innerHTML = outtext;
```

**体温と記述されたテキストを読み込み**

**Respiratory Rateと記述されたテキストを読み込み**

## 7 : Graph.jsの表示フォーマットを整える



### 異なるグラフを表示する方法

```
function onShowChart2(insertLabel, insertData){
    var charheader = document.createElement("script");
    charheader.innerHTML = "var ctx = document.getElementById('myChart').getContext('2d');"+
        "var myChart = new Chart(ctx, { "+
        "    type: 'bar',"+
        "    data: { "+
        "        //このラベル領域を差し替える
        "        insertLabel,
        "        datasets: [{ "+
        "            label: 'patient Heart Rate',"+
        "            //このデータ領域を差し替える
        "            insertData,
        "            backgroundColor: 'rgba(153,255,51,0.4)'"
        "        }],
        "    });";
    target = document.getElementById("outchartscript");
    target.appendChild(charheader);
}

function onShowChart3(insertLabel, insertData){
    var charheader = document.createElement("script");
    charheader.innerHTML = "var ctx = document.getElementById('myChart2').getContext('2d');"+
        "var myChart = new Chart(ctx, { "+
        "    type: 'line',"+
        "    data: { "+
        "        //このラベル領域を差し替える
        "        insertLabel,
        "        datasets: [{ "+
        "            label: 'patient Respiratory Rate',"+
        "            //このデータ領域を差し替える
        "            insertData,
        "            backgroundColor: 'rgba(255,153,0,0.4)'"
        "        }],
        "    });";
    target = document.getElementById("outchartscript");
    target.appendChild(charheader);
}
```

棒グラフの設定

折れ線グラフの設定

## 8 : 総合テスト



お疲れさまでした。  
コードは動きましたか？



## 参考資料

サンプル

