初学者向けFHIRハンズオンセミナー
PythonでFHIR!
-Webアプリを作ろう-

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

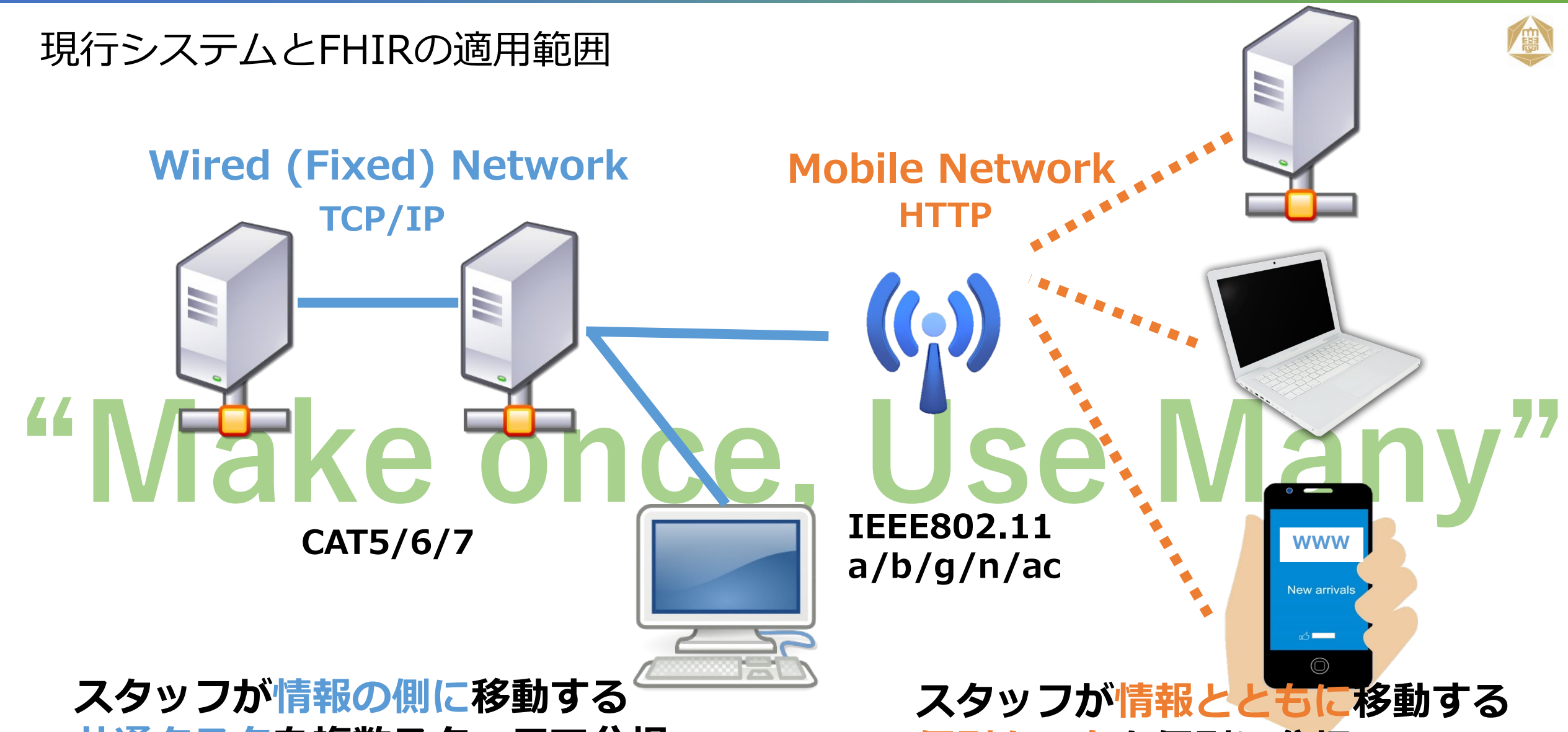
このチュートリアルの目的

- **HL7 FHIRの構成要素について、基本的なコーディングスキルを身につける**
  - HL7 FHIR、JSON、RESTとは何か
  - Pythonを利用したコーディングの基本
- **RESTとPythonライブラリを利用した簡単なWebサービスを作る**
  - RESTを利用してFHIRサーバからデータを取得する
  - 上記をWebサービスとして実装する
- **上記の取り組みを通して、HL7 FHIRの活用方法について具体的な活用方法・メリットについて具体的実感を得られる**

本日の流れ

- **HL7 FHIRについて（20分）**
  - 医療情報ユーザーから見たFHIRアプリケーションの利点
  - FHIRとはどんな規格か、REST, JSONについて
- **ハンズオン ～Python＋flaskで自作カルテ解析サーバを作ろう！～（約90分）**
  - 環境準備・確認（Pythonバージョンの確認、各種ライブラリのインストール）
  - ハンズオン1：PythonでRESTによりFHIRの患者データを取得しよう！
  - ハンズオン2：flaskを利用して解析結果をローカル環境に出力しよう！
- **質疑（10分）**

現行システムとFHIRの適用範囲



スタッフが情報の側に移動する
共通タスクを複数スタッフで分担
部門化の流れとともに浸透
共通の端末

スタッフが情報とともに移動する
個別タスクを個別に分担
チーム医療・情報の電子化、即時化
目的ごとのデバイス

HL7 FHIR （ファイア）とは

Fast **H**ealthcare **I**nteroperability **R**esource

- 米国のHL7協会が開発した医療情報交換のための新しい標準仕様（規格）
- 日本では、日本HL7協会、日本医療情報学会NeXEHRS研究会日本実装検討ＷＧなどが普及にむけて活動しはじめている

ＨＬ７ ver.2 （ 1980年代～ ） ISO 27931
主として医療情報システム同士のオーダ（検査や処方などの指示情報）か数値検査結果の連携手順と連携データの規格

準拠

厚労省標準規格
- HS012 臨床検査データ交換規約
- HS016 放射線データ交換規約
- HS022 処方データ交換規約
- HS026 SS-MIX標準化ストレージ

ＨＬ７ ver.3 （ 1995あたり～）
特に医療文書データの標準 HL7 CDA
医療全般の情報（画像やゲノム以外）をカバーして
多目的に利用できる「データ記述方法」の規格
医療文書データの標準(HL7CDA)は比較的使われている

準拠

厚労省標準規格（HL7CDAに準拠するもの）
- HS007 患者診療情報提供書／電子診療データ提供書
- HS008 診療情報提供書（電子紹介状）
- HS032 HL7 CDAに基づく退院時サマリー規約

厚労省医政局 電子処方箋 CDA 記述仕様
厚労省保険局 健診・特定保健指導の電子的なデータ標準様式

ＨＬ７ FHIR

仕様が複雑で実装時に多様性が生じるHL７ver3に対して、簡単な実装を重視して、規格策定が進んでいる

まだまだ発展途上の段階の部分が多い

準拠

2020年度厚生労働科学研究（特定研究）で策定作業中
- FHIR準拠 電子処方箋規格（仮称）
- FHIR準拠 特定健診・健診データ規約（仮称）
- FHIR準拠 退院時サマリー規約（仮称）
- FHIR準拠 患者診療情報提供書規約（仮称）

これまでの規格と何が違うのか

- FHIRが優れている理由

誰でも参加しやすい、参入しやすい

  - **実装性に強力にフォーカス：速く、簡単に実装できる**
    （複数の開発者がたった1日で簡単なインターフェイスを構築できた例もある）
    クイックスタートを可能とする多くの実装ライブラリが準備されている

既存の資産も活用できる

  - **革新的な相互運用性**：ベースとなるリソースは用意されているが、既に利用されているプロファイルやエクステンション、用語集などを採用できる
  - **既存のHL7 v.2、CDAとは相互互換**があり、**両方から発展的に活用できる**

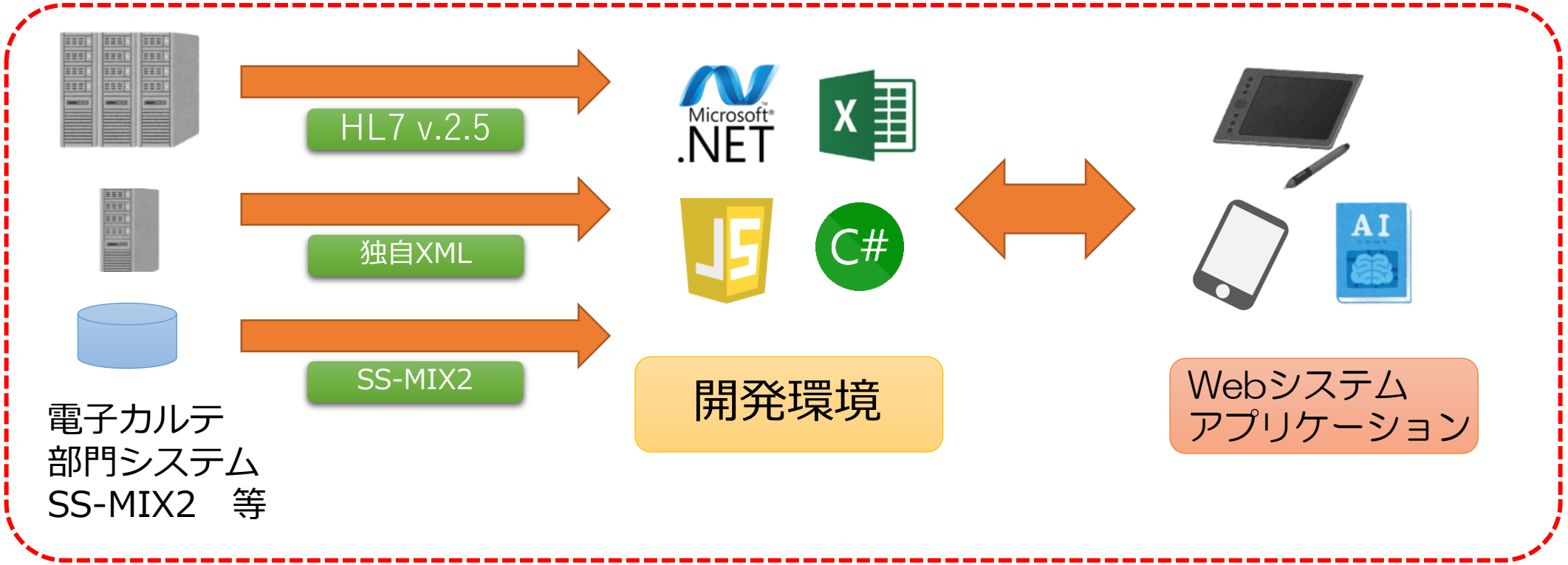
Web標準＝医療特化の脱却

  - **RESTfulアーキテクチャ**、メッセージとドキュメントを使用したシームレスな情報交換で技術開発社にとって学習障壁が低い

HL7 FHIR ポータルサイト「2.17.1 Why FHIR is better」を一部抜粋し翻訳（https://www.hl7.org/fhir/summary.html）

HL7 FHIRの導入が医療情報分野にもたらすもの

- **サービスを現場に導入するまで（従来）**

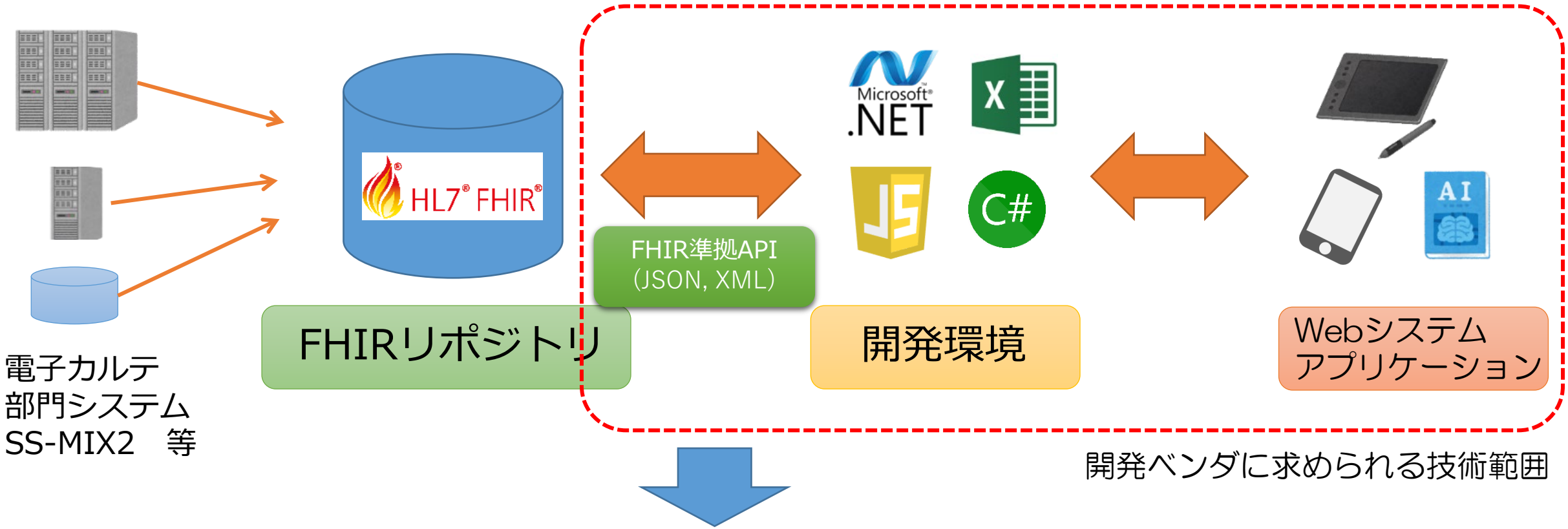


開発ベンダに求められる技術範囲

連携元のデータ構造や医療情報の通信規格も理解しなければならず、医療情報関係の開発経験の少ないベンダにとっては参入の障壁はとても高い

HL7 FHIRの導入が医療情報分野にもたらすもの

・サービスを現場に導入するまで（FHIRを利用）

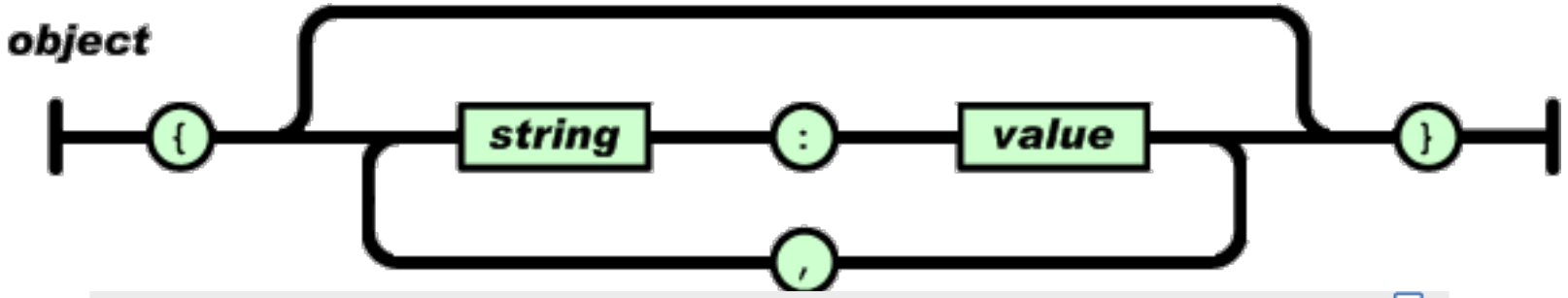


「医療情報のデータ交換には独特のアプローチが必要」という既成概念を覆す可能性
→より広範なベンダの参入を促し、ひいてはより良いサービス構築につながる

使うもの１：JSON（Web標準のデータ定義言語）

- **JSON=JavaScript Object Notation**
- **JavaScriptの文法に沿っています**
- **内容１：名前/値**
- **内容２：配列**

https://www.json.org/json-ja.html



例
{ “食べ物” : “うな重”}

```
{ "name"    : "John Smith",
  "sku"     : "20223",
  "price"   : 23.95,
  "shipTo" : { "name" : "Jane Smith",
               "address" : "123 Maple Street",
               "city" : "Pretendville",
               "state" : "NY",
               "zip"   : "12345" },
  "billTo" : { "name" : "John Smith",
               "address" : "123 Maple Street",
               "city" : "Pretendville",
               "state" : "NY",
               "zip"   : "12345" }
}
```

https://www.ibm.com/support/knowledgecenter/ja/SS9H2Y_7.5.0/com.ibm.dp.doc/json_jsonexamples.html

使うもの１：JSON（Web標準のデータ定義言語）

Label　　Value

```
{ "name"    : "John Smith",
  "sku"     : "20223",
  "price"   : 23.95,
  "shipTo"  : { "name" : "Jane Smith",
                "address" : "123 Maple Street",
                "city" : "Pretendville",
                "state" : "NY",
                "zip"   : "12345" },
  "billTo"  : { "name" : "John Smith",
                "address" : "123 Maple Street",
                "city" : "Pretendville",
                "state" : "NY",
                "zip"   : "12345" }
}
```

← 文字列型

← 数値型

← JSONデータの入れ子（連想配列）

https://www.ibm.com/support/knowledgecenter/ja/SS9H2Y_7.5.0/com.ibm.dp.doc/json_jsonexamples.html

使うもの2：REST

- **REST(REpresentational State Transfer)**
  - Web開発において最も一般的な通信技術の1つ
  - **リソース**と呼ばれる同じ性質を持つ情報の断片（例：Patient, Observation）と、**リソースを一意に識別するURI**で表されるアドレスを持つ

RESTで使用するURLの例

"http://hapi.fhir.org/baseDstu3/Patient/1646554/_history/1?_format=json"

サーバアドレス（サーバごと固定される部分）　リソース名とID　引数（取得するデータの条件指定）

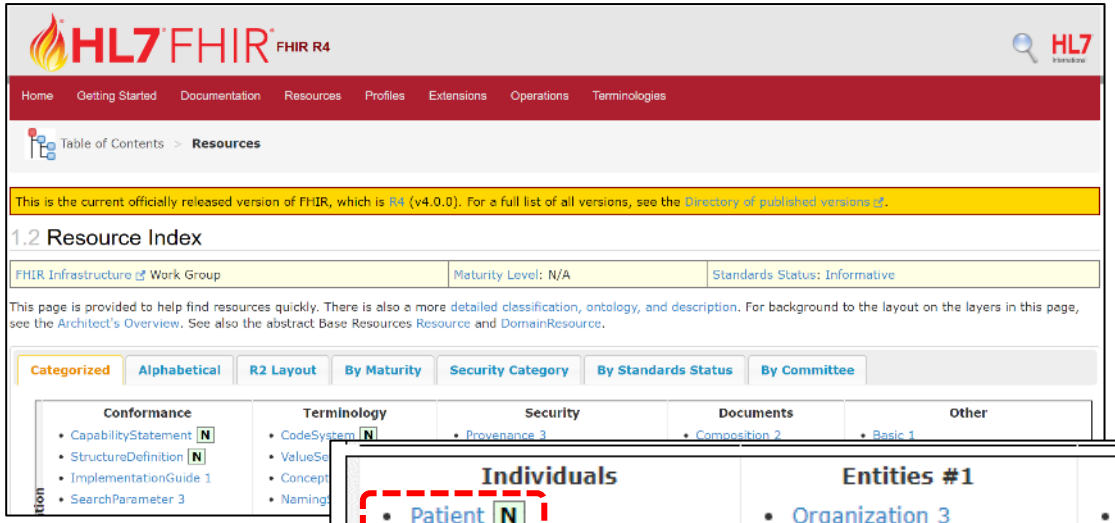
RESTサービスの設計原則

1. **アドレス指定可能なURIで公開されていること**
2. **インターフェース(HTTPメソッドの利用)の統一がされていること**
3. **ステートレスであること**
4. **処理結果がHTTPステータスコードで通知されること**

処理	HTTPメソッド	CRUD操作
登録	POST	CREATE
取得	GET	READ
更新	PUT	UPDATE
削除	DELETE	DELETE

リソースとは

- **FHIRリポジトリ（FHIR準拠のデータ格納庫（データベース））に保存される、一定のデータ構造をもった医療情報のかたまりそのもの**



Normativeとされたもの以外は、
FHIR Maturity Modelのレベルを表す。
値が大きいほど成熟度が高い。

Patient
（患者基本情報）
のリソース

Observation
（検査）
のリソース

Base

Individuals	Entities #1	Entities #2	Workflow	Management
Patient N	Organization 3	Substance 2	Task 2	Encounter 2
Practitioner 3	OrganizationAffiliation 0	BiologicallyDerivedProduct 0	Appointment 3	EpisodeOfCare 2
PractitionerRole 2	HealthcareService 2	Device 2	AppointmentResponse 3	Flag 1
RelatedPerson 2	Endpoint 2	DeviceMetric 1	Schedule 3	List 1
Person 2	Location 3		Slot 3	Library 2
Group 1			VerificationResult 0	

Clinical

Summary	Diagnostics	Medications	Care Provision	Request & Response
AllergyIntolerance 3	Observation N	MedicationRequest 3	CarePlan 2	Communication 2
AdverseEvent 0	Media 1	MedicationAdministration 2	CareTeam 2	CommunicationRequest 2
Condition (Problem) 3	DiagnosticReport 3	MedicationDispense 2	Goal 2	DeviceRequest 1
Procedure 3	Specimen 2	MedicationStatement 3	ServiceRequest 2	DeviceUseStatement 0
FamilyMemberHistory 2	BodyStructure 1	Medication 3	NutritionOrder 2	GuidanceResponse 2
ClinicalImpression 0	ImagingStudy 3	MedicationKnowledge 0	VisionPrescription 2	SupplyRequest 1

10.1.3 Resource Content

Structure | UML | XML | JSON | Turtle | R3 Diff | All

Structure

Name	Flags	Card.	Type	Description & Constra
Observation	I N		DomainResource	Measurements and sim + Rule: dataAbsentRea Observation.value[x] is + Rule: If Observation. Observation.componen the code SHALL NOT b Elements defined in An text, contained, extens
identifier	Σ	0..*	Identifier	Business Identifier for
basedOn	Σ	0..*	Reference(CarePlan | DeviceRequest | ImmunizationRecommendation | MedicationRequest | NutritionOrder | ServiceRequest)	Fulfills plan, proposal o
partOf	Σ	0..*	Reference(MedicationAdministration | MedicationDispense | MedicationStatement | Procedure | Immunization | ImagingStudy)	Part of referenced event
status	?! Σ	1..1	code	registered | preliminary | final | amended + ObservationStatus (Required)
category		0..*	CodeableConcept	Classification of type of observation Observation Category Codes (Preferred)
code	Σ	1..1	CodeableConcept	Type of observation (code / type) LOINC Codes (Example)
subject	Σ	0..1	Reference(Patient | Group | Device | Location)	Who and/or what the observation is about
focus	Σ TU	0..*	Reference(Any)	What the observation is about, when it is not about the subject of record
encounter	Σ	0..1	Reference(Encounter)	Healthcare event during which this observation is made
effective[x]	Σ	0..1		Clinically relevant time/time-period for observation
effectiveDateTime			dateTime	
effectivePeriod			Period	

Observationリソースの構造（階層構造）
米国HL7協会がリソースの定義を行ったもの
→**これとは別に日本版仕様（JP-CORE）が検討されている（後述）**

それぞれのリソースへのアクセスは、
RESTのURIで特定することができる
→**URIさえわかればHTTPでデータを取得したり、登録・修正したりできる**

各国のFHIR定義と実装ガイド

- **FHIRは実装性を主眼にして開発された規格である**
  - **80%のシステムが使用するであろう項目を標準で搭載**
  - **逆に残りは自由に実装できる**
- **各国が自国の制度に合ったFHIRの標準仕様を策定している**
  - **米国ではUS COREを策定**
  - **日本もJP COREを策定中**
- **それぞれの仕様を説明するために実装ガイドを公開している**
  - **JP COREの実装ガイドは今まさに作成中**

https://simplifier.net/guide/jpfhirjp/FHIRJP

日本版の仕様(JP-CORE)は検討段階

医療情報学会課題研究会NeXEHRS研究会が主催し、大学・企業から延べ100名超が参加
まもなく実装ガイド（α版）が公開される予定

HL7®FHIR® 日本実装検討WG

日本医療情報学会NeXEHRS研究会（正式名称：次世代健康医療記録システム共通プラットホーム課題研究会）

ようこそNeXEHRS研究会
HL7® FHIR®日本実装検討WGへ

[２０２１年WG日程]
第２０回全体WGミーティング案内　10月26日(火)　13:00～15:00　（Web会議形式）ZOOMアクセス情報はSlack(hl7fhir-jp-wg.slack.com)　#generalを参照ください

終了　■第14回WGミーティング　2021年1月25日（月）16:00～18:00 ★Web ZOOMでのみ
終了　■第15回WGミーティング　2021年3月25日（木）10:00～12:00 ★Web ZOOMでのみ
終了　■第16回WGミーティング　2021年5月27日（木）10:00～12:00 ★Web ZOOMでのみ
終了　■第17回WGミーティング　2021年6月30日（木）14:30～16:30 ★Web ZOOMでのみ
終了　■第18回WGミーティング　2021年8月25日(水)13:00～15:00 ★Web ZOOMでのみ
終了　■第19回全体WGミーティング案内　9月22日（水）16:00～18:00 ★Web ZOOMでのみ

■ HL7FHIR® HL7 FHIR Jp Core 実装ガイド
▶ 日本実装のためのImplementationGuideを作成中で、部分公開しています。
▶ FHIR JPーCORE Profile 作成中

■ HL7 FHIR® 医療関連文書FHIR準拠仕様の公開
https://std.jpfhir.jp/ をご覧ください。

FHIR®© FHIR実装ガイド

FHIRJP　Guidances　FHIRContents　Security　HowTo

HL7 FHIR Jp Core 実装ガイド ＜作成中＞

このドキュメントはHL7 FHIR実装ワーキンググループで作業中の実装ガイドです。日本HL7協会が承認するものではありませんので、実装や利用は全て自己責任で行ってください。

目次

- FHIRJP
  - Guidances
    - GeneralGuidance
    - Cardinality
    - Handling of non-existent data
    - Character Encoding
    - Search
  - FHIRContents
    - Profiles
    - Extensions
    - SearchParametersAndOperations
    - Terminology
    - CapabilityStatements
  - Security
  - HowTo

概要

- 1. ガイダンス: Jp Coreでの全体に関わる規則や注意事項を記載しています。
  - 1.1. 総合ガイダンス
  - 1.2. CardinaryとMust Supportの組み合わせ

HL7 FHIR日本実装検討WG

https://jpfhir.jp/

HL7 FHIR　医療関連文書FHIR記述仕様

https://std.jpfhir.jp/

- **令和２年度厚生労働科学特別研究事業の研究班により作成**
- **電子処方箋や健康診断結果報告書、退院時サマリ、診療情報提供書などのFHIR記述仕様書案が掲載されている**
- **JP COREもこの仕様書案と整合性を保って策定されている**

実際のデータ構造

(例)電子処方箋FHIR仕様

- 複数のリソースの組み合わせで構成されている
  - 文書全体はBundleリソースに包含されており、その下に全てのリソースがentryの形で登録される
  - Bundleのtypeを”document”にすることで、FHIR Documentという文書形式で記述できる
  - Compositionリソース･･･特定の文書情報を表すためのリソースの基本セットを構成する

- 構成要素に何を記述するべきかは、仕様書内で定義されている

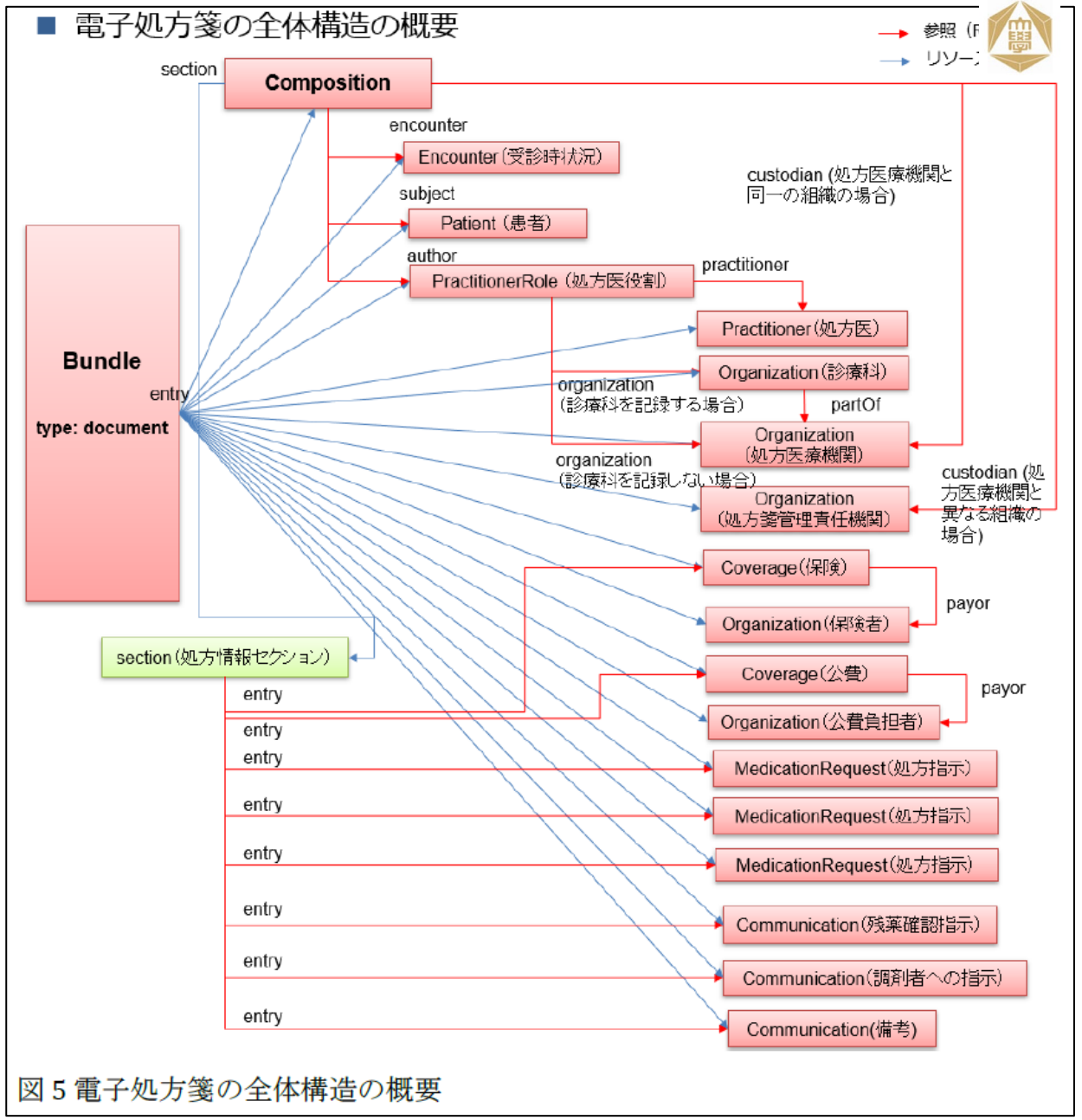


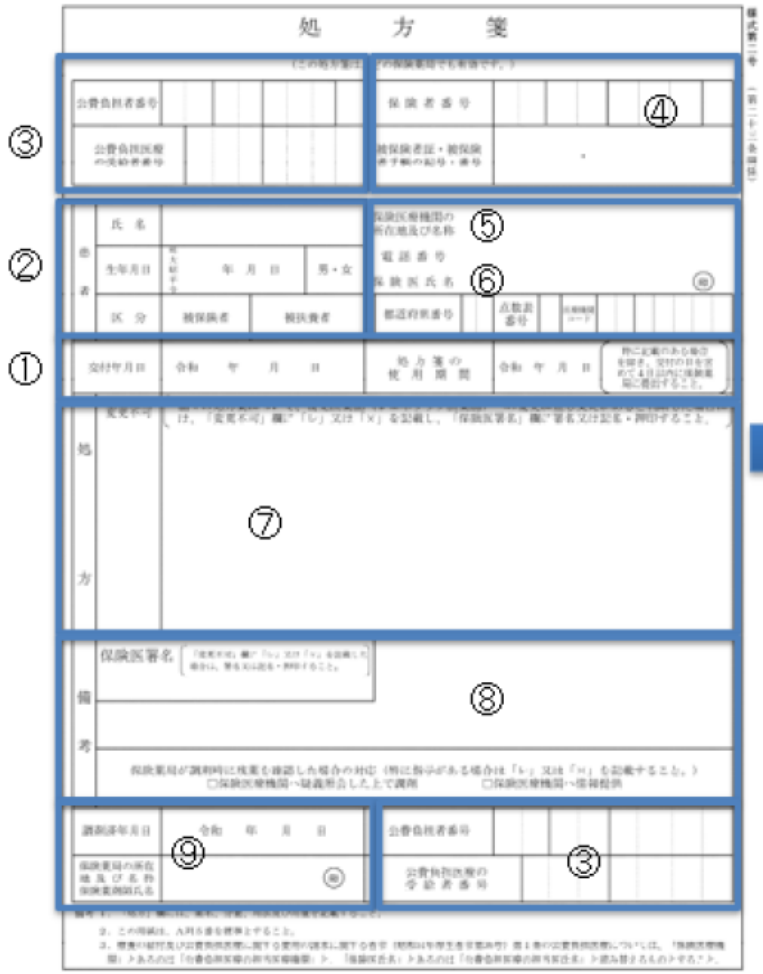
図 5 電子処方箋の全体構造の概要

電子処方箋 HL7 FHIR 記述仕様書案 2021-03-29A p.9より引用

紙の処方箋との対応

- 紙の処方箋を構成する情報は、それぞれのFHIRリソースと対応している
- 電子処方箋FHIR記述仕様では、**文書全体に電子署名**をすることにより改ざん等を検知できる仕組みになっている

電子処方箋仕様における FHIR主要リソースの構成



HL7 FHIR Bundleリソース（文書タイプ）

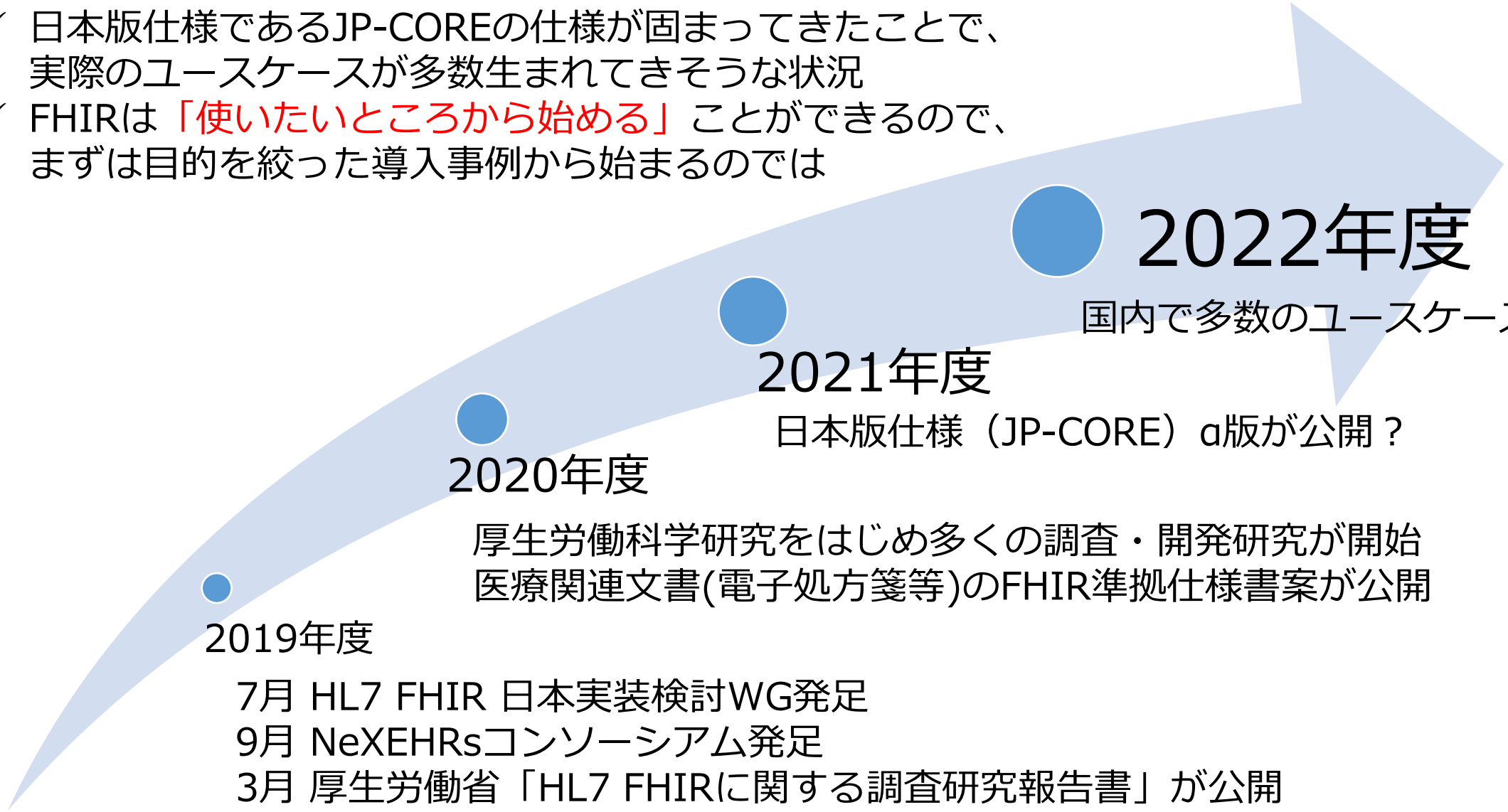
	リソース内容	FHIRリソース名
①	文書情報	Composition
②	患者情報	Patient
③	公費負担情報	Coverage
④	被保険者情報	Coverage
	保険者情報	Organization
⑤	処方医療機関情報	Organization
	診療科情報	Organization
⑥	処方医役割情報	PractitionerRole
	処方医情報報	Practitioner
⑦	医薬品処方情報	MedicationRequest
	:（繰り返し）	:
⑧	備考・薬局への伝達情報	Communication
⑨	調剤時記録情報	策定中
	全体のメッセージダイジェスト（ハッシュ値）	（Signature）

図3 紙の処方箋を構成する情報とFHIR リソースとの対応関係

電子処方箋 HL7 FHIR 記述仕様書案 2021-03-29A p.5より引用

現時点までの国内でのFHIR普及に向けた動向

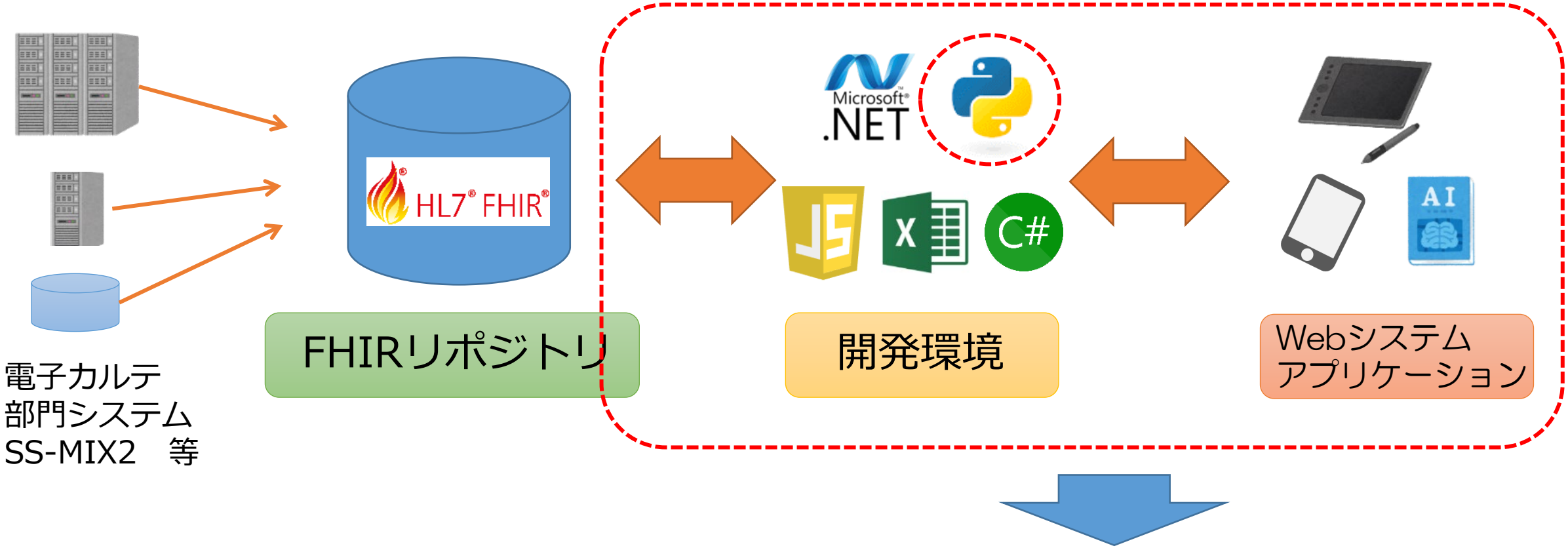
- ✓ 日本版仕様であるJP-COREの仕様が固まってきたことで、実際のユースケースが多数生まれてきそうな状況
- ✓ FHIRは「使いたいところから始める」ことができるので、まずは目的を絞った導入事例から始まるのでは



チュートリアル前半

本日の内容

- **FHIRを現場に導入するまでの道のり**



開発の敷居の低さを体感する

Pythonのインストール（未インストールの方のみ）

・Python 3.7以降（推奨は下記の3.9.6）をインストールして下さい

「Download」からOSを選択してクリックし、表示されたページを下の方にスクロール

https://www.python.org/

※Macの方はこちらを利用して下さい

- Python 3.9.6 - June 28, 2021
  - Download macOS 64-bit Intel installer
  - Download macOS 64-bit universal2 installer

上段が従来のインストーラです。M1チップ搭載機など Apple Siliconを利用している方は下段を利用

- Python 3.9.6 - June 28, 2021

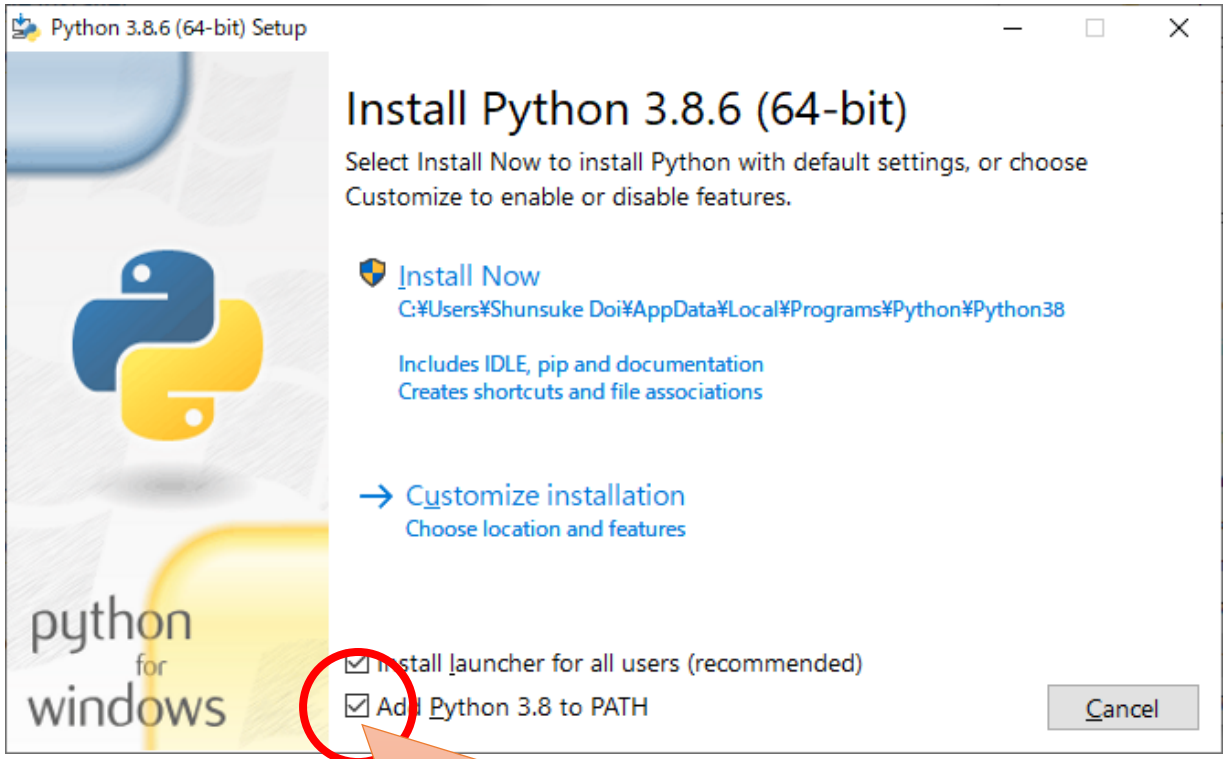
Note that Python 3.9.6 *cannot* be used on Windows 7 or earlier.

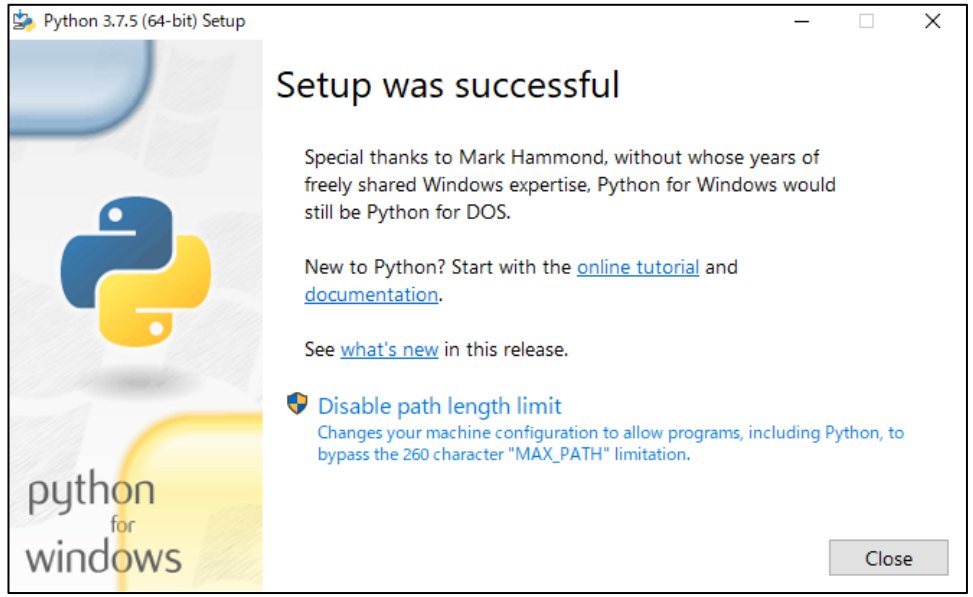
- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (64-bit)
- Download Windows help file
- Download Windows installer (32-bit)
- Download Windows installer (64-bit)

- Download Windows installer (32-bit)
- Download Windows installer (64-bit)

Windowsの方は上記のインストーラをダウンロードいただき、ダブルクリックで実行して下さい。（32bitか64bitかで違うため注意）

（Pythonのインストール（未インストールの方のみ）





※重要※
必ずチェックを入れておいて下さい。

これでインストールは完了です。

- **他のバージョンでも動作するものと思いますが、全環境への動作保証はしておりません。**
- **また、最新版の3.10.0も動作未検証ですのでその点ご了承下さい。**

ハンズオン準備（作業用フォルダと環境の準備）

① 協会提供のリンクから、今回使用するプログラムをダウンロードして下さい。

当日のご案内・環境要件について

1. pythonを利用するハンズオンのため、事前にpython 3.7以降（3.8.xまたは3.9.6までを推奨、3.10.0は未検証）をインストールしたパソコン・タブレット等をご用意下さい。機器の貸出等は行っておりません。
2. 本チュートリアルではインターネット接続が必要となります。会場のWi-Fiを利用いただくか、通信手段をご準備下さい。
3. 現地参加の場合は、デバイスをあらかじめ充電の上ご参加いただきますようご協力をお願いいたします。
4. **会場の都合上、机をご用意できません。**膝上で操作いただく形になりますので何卒ご了承下さい。

事前準備・ダウンロード

当日チュートリアルをスムーズに進行するため、当日利用するPythonの環境の準備やデータの事前ダウンロードにご協力をお願いいたします.

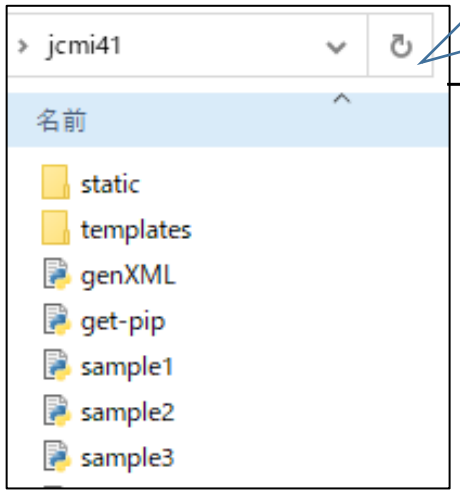
1. 事前準備用の説明資料
   こちらからダウンロードして下さい。
   Pythonのインストール方法から、Flaskのインストールまで一通りの説明を載せています。
2. Pythonのインストール
   本チュートリアルではpipを利用するため、Python 3.6以降をインストールして下さい。バージョンについては、最新版の3.10.0は動作未検証のため、3.8.xまたは3.9.6までを推奨します。
   Pythonのダウンロード
3. チュートリアル当日用の説明資料
   準備中（11/17(水)までにアップロード予定）
4. 配布プログラム（日本Mテクノロジー学会作成）：

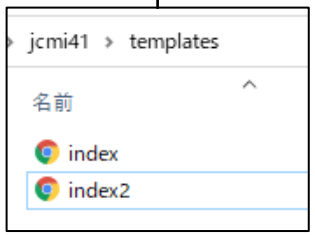
   準備中（11/17(水)までにアップロード予定）
   【※著作権について※】本プログラムの著作権は一般社団法人日本Mテクノロジー学会に帰属します。複製、再配布の際には当会の提供であることを明記いただき、改変使用される場合は当会までご連絡下さい。なお、本プログラムの使用により生じたいかなるトラブル、損失、損害等に対して、当会は一切責任を負いません。

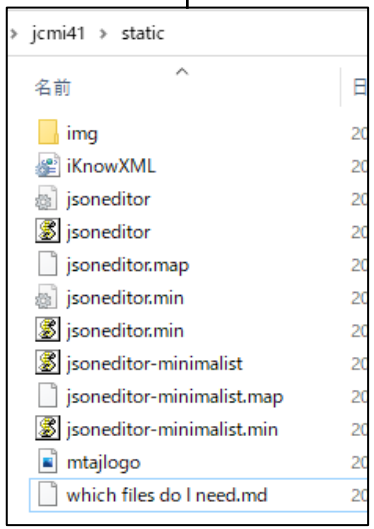
ご不明な点につきましては事務局（mta-office【あっとまーく】mta.gr.jp）までご連絡をお願いいたします。

② デスクトップに「hl7fhir」等の名前の作業用フォルダを準備し、事前ダウンロードしたファイルを格納します。

作業フォルダの中にpythonの実行ファイルと「templates」「status」の2つのフォルダがあります。



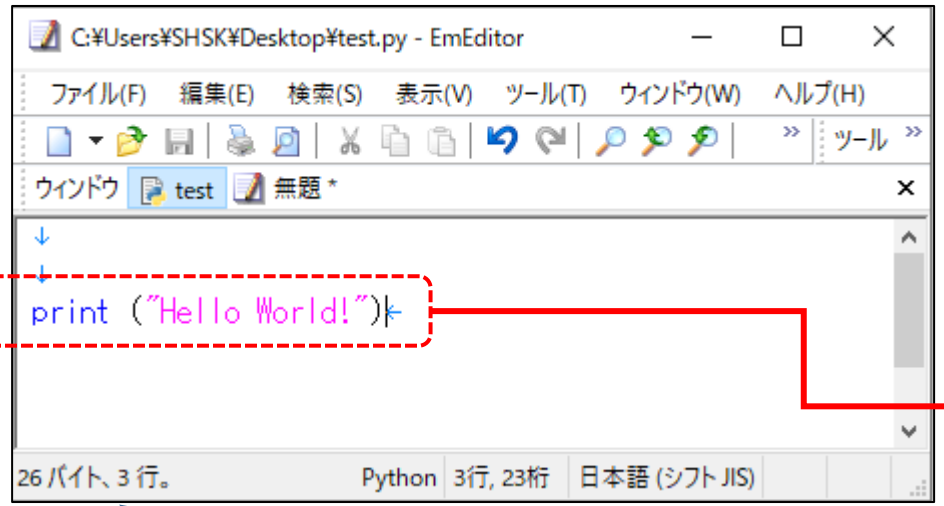


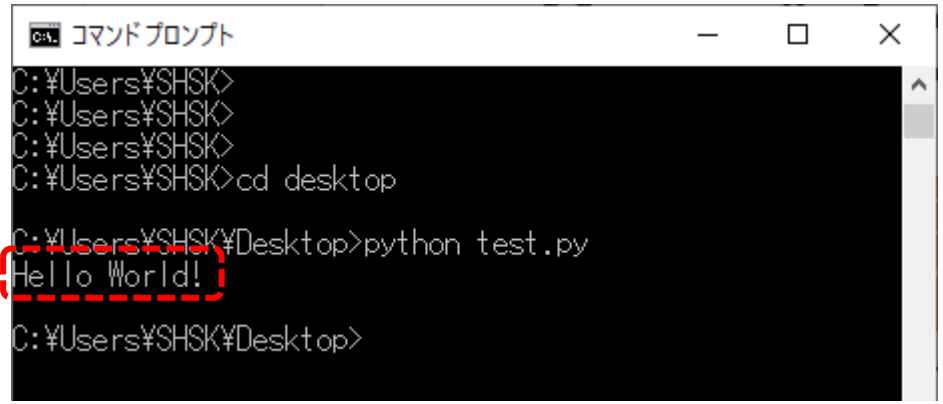


ハンズオン準備（Pythonの基本的な動作方法の説明）

① テキストエディタでコードを編集し、「○○.py」のファイル名で保存します

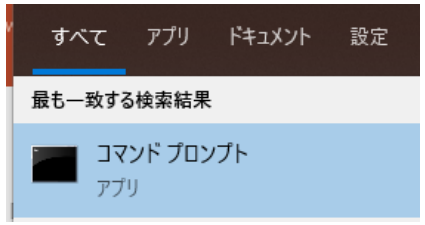
② 保存したPythonのファイルをコマンドから「python ○○.py」の形で実行します。
※MACの方、2.x系もインストールしている方は、「python -3 ○○.py」と入力して下さい。





エディタは各自の環境をご使用下さい。

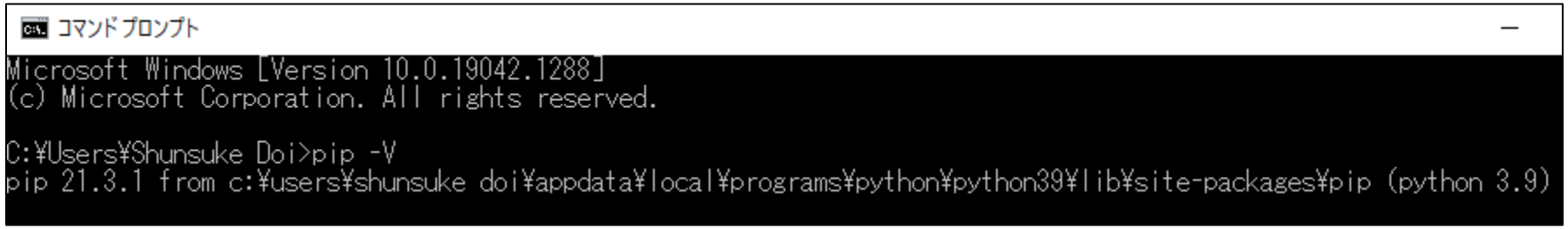
Windowsの方はTeraPadが標準でインストールされています。



Windowsの方はスタートボタンをクリック後に「cmd」と入力しEnterを押下するとコマンドプロンプトが起動します。
(Macではターミナルというアプリ)

ハンズオン準備（pipのインストール確認）

- **本チュートリアルでは、Pythonのライブラリを利用するためpipを利用します。**
- **pipがインストールされているかは「pip -V」のコマンドで確認できます。**

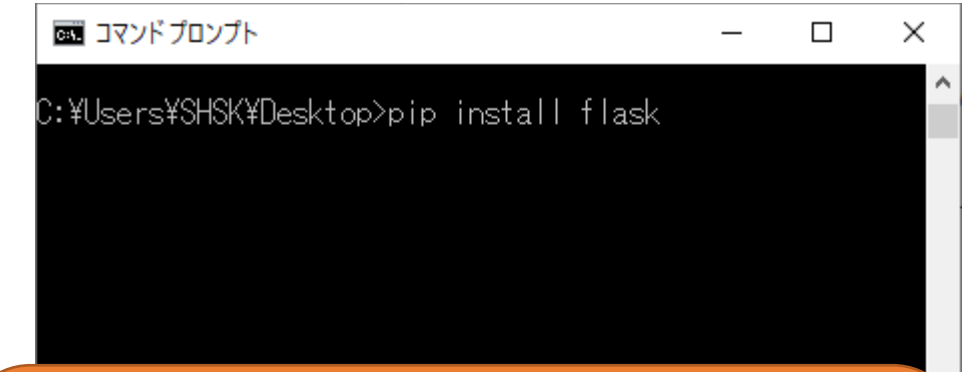


- **pipのバージョンが古い場合は、「python –m pip install --upgrade」でアップデートすることができます。**

※pipがインストールされていない方は、一度pythonをアンインストールし、3.9系のpythonをダウンロードしてインストールし直して下さい。（ver.3.4 以降は標準で内包されています）

ライブラリのインストール

- **本チュートリアルでは、PythonのWebフレームワークとして、「Flask」というライブラリを使用します。**
- **pipコマンドを使用することで簡単にインストール可能です。**
- **同様に「iknowpy」もインストールします。**



① コマンドプロンプトで「pip install flask」と入力し実行します。（ここはどのフォルダで実行してもOK）
※MACの方、2.x系もインストールしている方は、「pip3 install flask」と入力して実行して下さい。

② メッセージが表示されインストールが進行します。同様に「pip install iknowpy」も実行します。

ハンズオン準備（Flaskのインストールの確認）

- **作業フォルダ内にある「hello.py」をCMDで実行します。**

hello.py - TeraPad

ファイル(F) 編集(E) 検索(S) 表示(V) ウィンドウ(W)

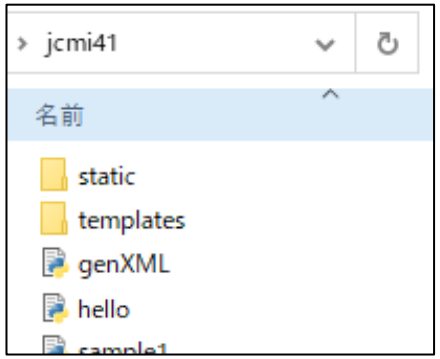
```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    name = "Hello World"
    return name
@app.route('/jcmi41')
def good():
    name = "今日は学会初日です！"
    return name
if __name__ == "__main__":
    app.run(debug=True)
[EOF]
```

A

B

hello.pyのコード

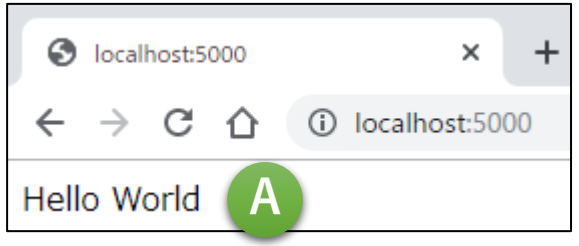
① 作業フォルダにあるhello.py を使用します。



② コマンドプロンプトで作業フォルダに移動し、hello.pyを実行

③ Webブラウザを開くとPythonの出力をブラウザ上に表示できます。

http://localhost:5000/



http://localhost:5000/jcmi41/

④ Ctrl ＋ C で終了します

エラーが出る方は

- **Macの方は、既定でインストールされているPython 2.xと競合することがあります。以下のコードをファイルの1行目に加えることで、解決することがあります。**
- **#!/usr/bin/env python3**

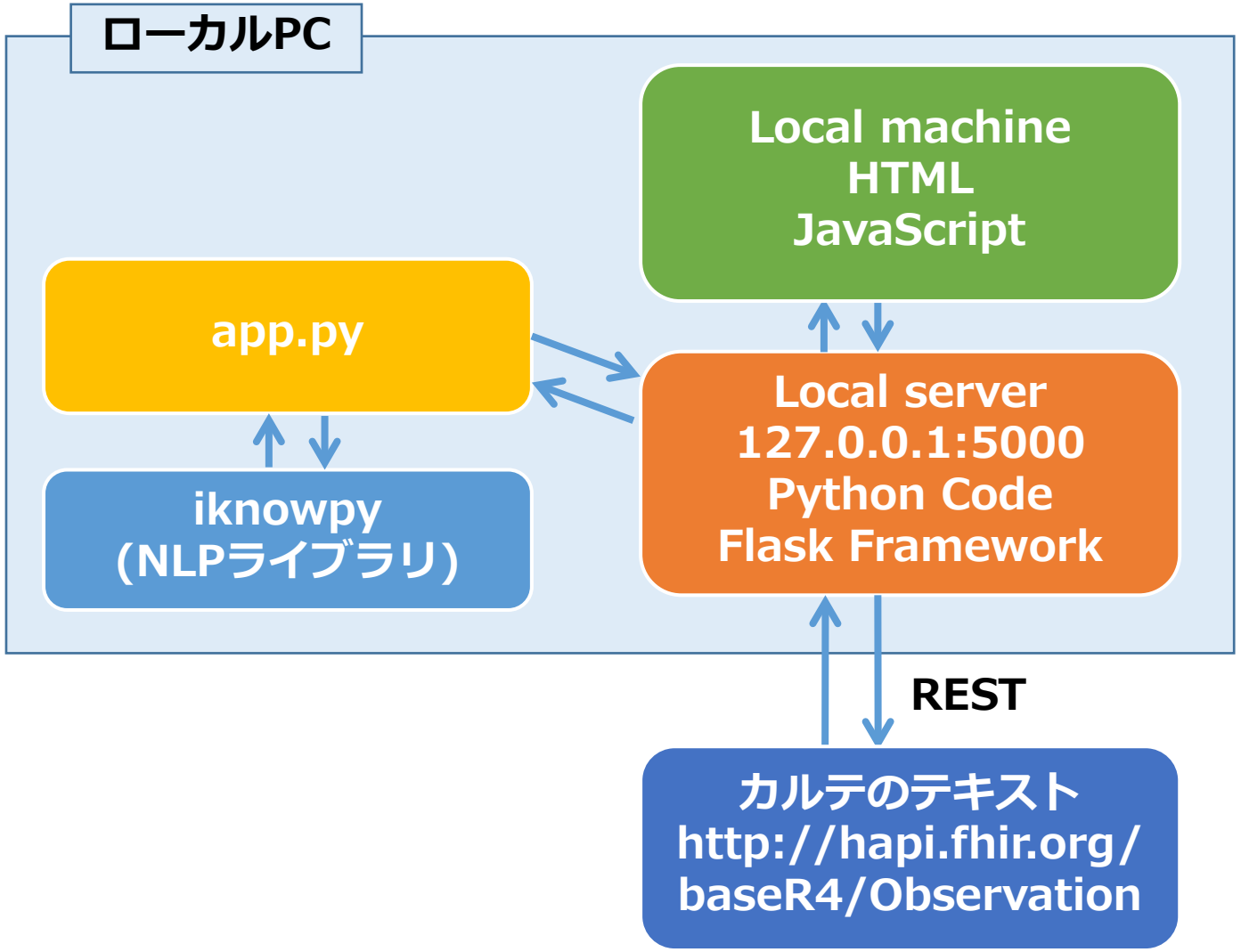
- **Pythonではディレクトリは「/」（スラッシュ）で記述して下さい。例：C:/Users/user/desktop/jcmi41**

- **それでも動かない場合、終了後にご質問ください**

本日の流れ

- **HL7 FHIRについて（20分）**
  - 医療情報ユーザーから見たFHIRアプリケーションの利点
  - FHIRとはどんな規格か、REST, JSONについて
- **ハンズオン ～Python＋自然言語処理エンジンで自作カルテ解析サーバを作ろう！～（90分）**
  - 環境準備・確認（Pythonバージョンの確認、各種ライブラリのインストール）
  - ハンズオン1：PythonでRESTによりFHIRの患者データを取得しよう！
  - ハンズオン2：iKnowPyを利用して解析結果をローカル環境に出力しよう！
- 質疑（10分）

完成イメージ



Python/Flaskが中心となり、

1）FHIRサーバとのREST通信
2）jsonデータの読み取り

Observationリソースの取得とテキスト解析、HTMLへの送信プログラムを作成
（表示系は一部JavaScriptを利用）

Pythonとは

スクリプト型言語の一種

事前のデータ型定義を必要としない

処理の階層、繰り返しの際のインデントなど、書式そのものの規定がある

機械学習についてのライブラリが充実していたことも人気を拡大した

優れたオープンソースライブラリが提供されている

PythonにおけるREST呼び出しの最も単純な形

サンプルコード
（app1.pyの一部）

```
#!/usr/bin/env python3
↓
# ライブラリのインポート
import urllib.request
import ssl
import json
import pprint
↓
ssl._create_default_https_context = ssl._create_unverified_context
↓
# RESTで取得するFHIRデータのURI
resturl = 'https://fhirtestjp.med.gunma-u.ac.jp/Patient/1'
↓
# 取得したデータを加工し表示
with urllib.request.urlopen(resturl) as response:
    fhir = response.read()
    json1 = json.loads(fhir.decode('utf-8')) #注意: json.loadはファイルから読む場合に使用する
↓
    #要素全てを取得
    print(json1['identifier'])
↓
    #リストの最初を読み出し
    print(json1['identifier'][0]['value'])
↓
    #リストの大きさを理解して読み出し
    print(json1['identifier'][len(json1['identifier'])-1]['value'])
↓
    #jsonデータ全てを表示
    pprint.pprint(json1)
```

RESTのrequest処理

JSON処理に必須

FHIRリソースのURL

JSONをdictionary型にして読み込む

特定の要素のみを出力

JSONデータをそのものをコンソール出力

ハンズオン1　PythonでRESTによりFHIRの患者データを取得

RESTで取得したデータを「コンソールに出力する」「ファイルに出力する」方法

```python
#!/usr/bin/env python3
↓
# ライブラリのインポート↓
import urllib.request↓
import ssl↓
import json↓
import pprint↓
↓
ssl._create_default_https_context = ssl._create_unverified_context↓
↓
# RESTで取得するFHIRデータのURI↓
resturl = 'https://fhirtestjp.med.gunma-u.ac.jp/Patient/1'↓
↓
# 取得したデータを加工し表示↓
with urllib.request.urlopen(resturl) as response:↓
    fhir = response.read()↓
    json1 = json.loads(fhir.decode('utf-8')) #注意: json.loadはファイルから読む場合に使用する
↓
    #要素全てを取得↓
    print(json1['identifier'])↓
↓
    #リストの最初を読み出し↓
    print(json1['identifier'][0]['value'])↓
↓
    #リストの大きさを理解して読み出し↓
    print(json1['identifier'][len(json1['identifier'])-1]['value'])↓
↓
    #jsonデータ全てを表示↓
    pprint.pprint(json1)↓
    ↓
    #jsonデータをテキストファイルに出力する↓
    path_w = './applout.txt'  # ※ここのパスは好きな場所に書き換え可↓
    with open(path_w, mode='w') as f:↓
        json2 = json.dumps(json1, indent=2, ensure_ascii=False)↓
        f.write(json2) # 問題なく出力されることを確認します。↓
```

エディタにapp1.pyを表示

ファイル出力部

①CMDで「python app1.py」を実行

```
¥pythonfiles¥jcmi41>
¥pythonfiles¥jcmi41>app1.py
```

No module named ***
のようなエラーが出た場合は、モジュールのインストールが済んでいないので、「pip3 install ***」のように入力してインストールします。

```
Traceback (most recent call last):
  File "app.py", line 5, in <module>
    import matplotlib.pyplot as plt
ModuleNotFoundError: No module named 'matplotlib'
```

ハンズオン1　RESTで取得したFHIRデータの確認

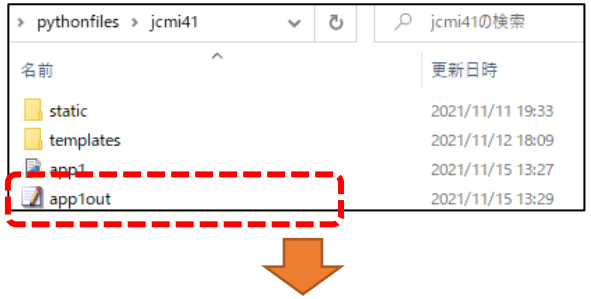
RESTで取得したデータを「ブラウザに表示する」「ファイルに出力する」方法

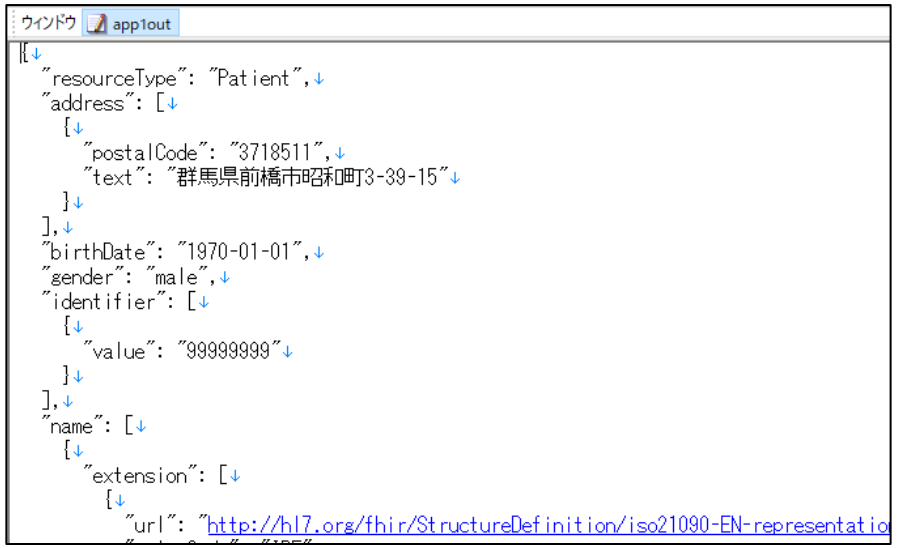
② コンソールに取得したFHIRデータが表示される

```
[['value': '99999999']]
99999999
99999999
['address': [['postalCode': '3718511', 'text': '群馬県前橋市昭和町3-39-15']],
 'birthDate': '1970-01-01',
 'gender': 'male',
 'id': '1',
 'identifier': [['value': '99999999']],
 'meta': {'lastUpdated': '2021-03-06T12:57:58Z', 'versionId': '1'},
 'name': [['extension': [['url': 'http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation',
                          'valueCode': 'IDE']],
           'family': '群馬',
           'given': ['太郎'],
           'text': '群馬 太郎',
           'use': 'official'],
          ['extension': [['url': 'http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation',
                          'valueCode': 'SYL']],
           'family': 'グンマ',
           'given': ['タロウ'],
           'text': 'グンマ タロウ',
           'use': 'official']],
 'resourceType': 'Patient',
 'telecom': [['value': '0272208773']]]
```

```
#要素全てを取得
print(json1['identifier'])          → [['value': '99999999']]

#リストの最初を読み出し
print(json1['identifier'][0]['value'])          → 99999999

#リストの大きさを理解して読み出し
print(json1['identifier'][len(json1['identifier'])-1]['value'])          → 99999999

#jsonデータ全てを表示
pprint.pprint(json1)          → ['address': [['postalCode':
                                 'birthDate': '1970-01-01',
```

③作業フォルダにできた「app1out.txt」を開くと、②のjsonと同じ内容が表示されます。





ハンズオン1　FHIRデータの欲しい部分を取得するには

Patientリソースの構造（HL7 FHIR公式）

8.1.2 Resource Content

Structure | UML | XML | JSON | Turtle | R3 Diff | All

Structure

Name	Flags	Card.	Type	Description & Constraints
Patient	N		DomainResource	Information about an individual or animal rece Elements defined in Ancestors: id, meta, impl
identifier	Σ	0..*	Identifier	An identifier for this patient
active	?! Σ	0..1	boolean	Whether this patient's record is in active use
name	Σ	0..*	HumanName	A name associated with the patient
telecom	Σ	0..*	ContactPoint	A contact detail for the individual
gender	Σ	0..1	code	male | female | other | unknown AdministrativeGender (Required)
birthDate	Σ	0..1	date	The date of birth for the individual
deceased[x]	?! Σ	0..1		Indicates if the individual is deceased or not
deceasedBoolean			boolean	
deceasedDateTime			dateTime	
address	Σ	0..*	Address	An address for the individual
maritalStatus		0..1	CodeableConcept	Marital (civil) status of a patient MaritalStatus (Extensible)
multipleBirth[x]		0..1		Whether patient is part of a multiple birth
multipleBirthBoolean			boolean	
multipleBirthInteger			integer	
photo		0..*	Attachment	Image of the patient
contact	I	0..*	BackboneElement	A contact party (e.g. guardian, partner, friend

データ構造　多重度　データ型

FHIRリソースの設計を理解する

① データ構造（階層）の場所を確認する

リソースのプロファイルを確認し、Pythonの辞書型データのどこを指定するかを把握します。

② データの多重度(Cardinality)を確認する

(必須格納数)..(可能格納数)という表記。多重度によってプログラムを少し工夫する必要があります。

③ データ型(Type)を確認する

データ型によってはさらに深い階層にデータが格納されていることが（よく）あります。

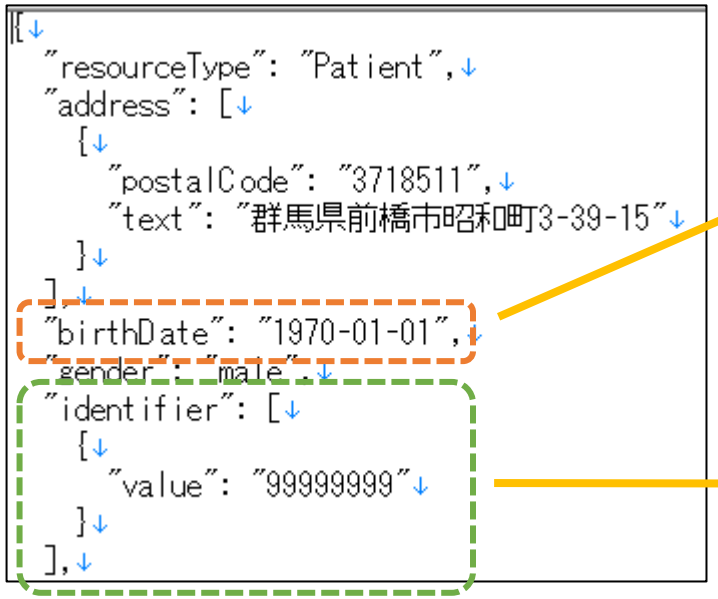
Structure

Name	Flags	Card.	Type
HumanName	Σ N		Element
use	?! Σ	0..1	code
text	Σ	0..1	string
family	Σ	0..1	string
given	Σ	0..*	string
prefix	Σ	0..*	string
suffix	Σ	0..*	string
period	Σ	0..1	Period

例えばHumanName型のデータは、左のような階層構造を持っており、**Patient.name.text**のように表現されます。

ハンズオン1　RESTで取得したFHIRデータの確認

改めて出力されたjsonを確認すると



“birthdate”は第1階層なので、

json1[‘birthdate’]

で取り出すことができます。

“identifier”の”value”は第2階層なので、

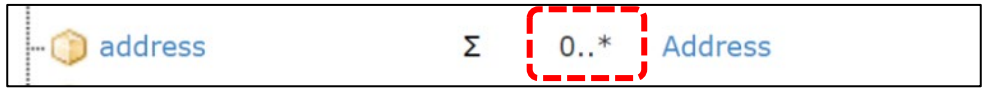
json1[‘identifier’][0][‘value’] となります。

この[0]は何でしょう？？

ハンズオン1　多重度が0...*または1...*の場合の対応方法

- **JSONではインスタンス数が2以上になる場合、[]←中カッコで囲われるので、python側でも読み取り方の工夫が必要になる**

【FHIRリソース】



Name	Flags	Card.	Type
Address	Σ N		Element
use	?! Σ	0..1	code
type	Σ	0..1	code
text	Σ	0..1	string
line	Σ	0..*	string
city	Σ	0..1	string
district	Σ	0..1	string
state	Σ	0..1	string
postalCode	Σ	0..1	string
country	Σ	0..1	string
period	Σ	0..1	Period

【jsonでの表現】

中カッコ[ ]が目印です

```
"address":[
  {
    "text": "神奈川県横浜市港区１－２－３",
    "postalCode": "123-4567",
    "country": "JP"
  }
]
```

【pythonのdictionary型での表現】

```
address_text = json1['address'][0]['text']
```

例え1つしか値が入っていない場合でも、pythonでは明示的に1つ目のデータを表す**[0]**が必要になります。
もしデータ数が不明の場合はインスタンス数のカウントや、存在しない場合のエラー処理が必要になる

ハンズオン1　名前のデータを取得して出力してみましょう

app1.pyに漢字氏名を出力するプログラムを書く

```
#要素全てを取得
print(json1['identifier'])

#リストの最初を読み出し
print(json1['identifier'][0]['value'])

#リストの大きさを理解して読み出し
print(json1['identifier'][len(json1['identifier'])-1]['value'])

#【ハンズオン1】ここに取り出しすデータを記入してみましょう


#jsonデータ全てを表示
pprint.pprint(json1)
```

```
"name": [
  {
    "extension": [
      {
        "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation",
        "valueCode": "IDE"
      }
    ],
    "use": "official",
    "text": "群馬 太郎",
    "family": "群馬",
    "given": [
      "太郎"
    ]
  },
  {
    "extension": [
      {
        "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation",
        "valueCode": "SYL"
      }
    ],
    "use": "official",
    "text": "グンマ タロウ",
    "family": "グンマ",
    "given": [
      "タロウ"
    ]
  }
],
```

ヒント

すぐ上の**identifier**のコードをコピーすると便利です

print(json1　　)

この部分に指定するコードを書きます。

ハンズオン1　名前のデータを取得して出力してみましょう

（解答）app1.pyに漢字氏名を出力するプログラムを書いてみましょう。

```
#【ハンズオン1】ここに取り出しすデータを記入してみましょう
↓
print(json1['name'][0]['text'])
```

app1.pyを実行します

```
C:¥Users¥            ¥pythonfiles¥jcmi41>app1.py
[{'value': '99999999'}]
99999999
99999999
群馬 太郎
```

漢字氏名が表示されれば成功です


```
"name":[
  {
    "extension": [
      {
        "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation",
        "valueCode": "IDE"
      }
    ],
    "use": "official",
    "text": "群馬 太郎",
    "family": "群馬",
    "given": [
      "太郎"
    ]
  },
  {
    "extension": [
      {
        "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation",
        "valueCode": "SYL"
      }
    ],
    "use": "official",
    "text": "グンマ タロウ",
    "family": "グンマ",
    "given": [
      "タロウ"
    ]
  }
],
```


この中カッコはtextにはかかっていないので注意

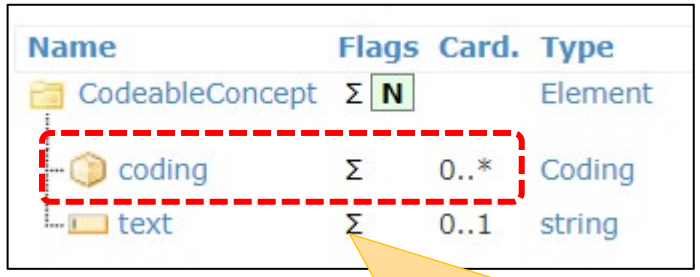
Tips　カナを取得したい場合は？？

JP-COREでは、漢字氏名の他にカナ氏名を入れる仕様になっています。漢字の方が先に格納されるので[0]を指定しましたが、実際のシステムでは”valueCode”のコードを一緒に読み込み、カナと漢字を判断します。

(参考)FHIRプロファイル（記述仕様）による仕様の違いに注意

- **電子処方箋FHIR記述仕様をもとにコーディングする場合に気をつけておきたいこと…電子処方箋独自の多重度に注意**

【FHIRの標準仕様】



CodeableConcept.codingの多重度は0…*

【電子処方箋仕様書案】

16	substitution				0..1	BackboneElement		後発医薬品への変更可否情報。詳細は「5.1 後発品変更可否」参照。
16.1		allowedCodeableConcept			1..1	CodeableConcept		
16.1.1			coding		1..1	Coding		後発品変更不可コード。
16.1.1.1				system	1..1	uri	"urn:oid:1.2.392.100495.20.2.41"	後発品変更不可コードを識別する URI。固定値。
16.1.1.2				code	1..1	code	“1”	後発品変更不可コード。値は例示。
16.1.1.3				display	0..1	string	“変更不可”	値は例示。

Substitution.allowedCodeableConcept.codingの多重度は1…1になっている

```
"substitution": [↓
  "allowedCodeableConcept": [↓
    "coding": [↓
      {↓
        "system": "urn:oid:1.2.392.100495.20.2.41",↓
        "code": "1",↓
        "display": "変更不可"↓
      }↓
    ]↓
```

ただしjsonは標準の0…*に合わせた[]中カッコが出現する仕様になっている

多重度でうまくいかない場合は、実際のjsonデータを見て確認するように

```
If jsonObj("entry")(i).Exists("substitution") = True Then
  Sheet2.Cells(row + 9 + j, 10) = jsonObj("entry")(i)("substitution")("allowedCodeableConcept")("coding")(1)("display")
  Sheet2.Cells(row + 9 + j, 11) = jsonObj("entry")(i)("substitution")("reason")("text")
End If
```

チュートリアル後半

本日の流れ

- **HL7 FHIRについて（20分）**
  - 医療情報ユーザーから見たFHIRアプリケーションの利点
  - FHIRとはどんな規格か、REST, JSONについて
- **ハンズオン ～Python＋自然言語処理エンジンで自作カルテ解析サーバを作ろう！～（90分）**
  - 環境準備・確認（Pythonバージョンの確認、各種ライブラリのインストール）
  - ハンズオン1：PythonでRESTによりFHIRの患者データを取得しよう！
  - ハンズオン3：Pythonの変数をWebブラウザ上に表示してみよう！
- 質疑（10分）

（復習）ライブラリ「Flask」の仕組み

- **Pythonの変数をHTMLコードに動的に組み込みWebサービスを構築**

hello.py - TeraPad

ファイル(F) 編集(E) 検索(S) 表示(V) ウィンドウ(W)

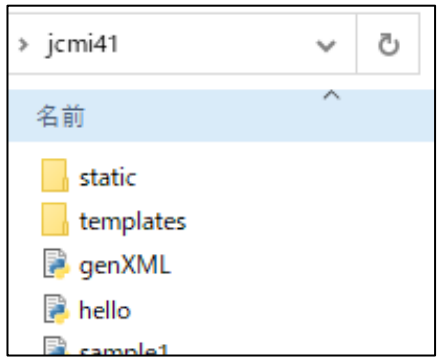
```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    name = "Hello World"
    return name
@app.route('/jcmi41')
def good():
    name = "今日は学会初日です！"
    return name
if __name__ == "__main__":
    app.run(debug=True)
[EOF]
```

A

B

hello.pyのコード

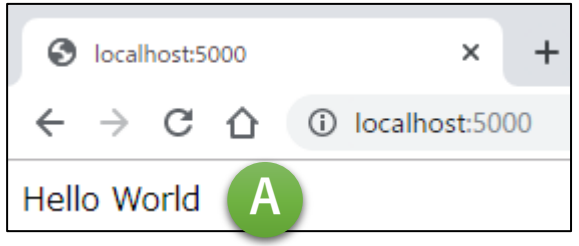
① 作業フォルダにあるhello.pyを使用します。



② コマンドプロンプトで作業フォルダに移動し、hello.pyを実行

③ Webブラウザを開くとPythonの出力をブラウザ上に表示できます。

http://localhost:5000/



http://localhost:5000/jcmi41/

④ Ctrl ＋ C で終了します

FlaskフレームからHTMLへのデータの動的渡し

- **ただ出力を表示するだけでなく、テンプレートになるHTMLファイルを用意しておき、その中に変数を組み込むこともできます**

app3.py

```
#FHIR取得したリソースを「text」という変数に入れhtmlに送る
html = render_template('index.html', text = jdata)
return html
```

文字列変数を割り当ててindex.htmlに送信

/templates/index.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <title>HELLO</title>
</head>
<body>
  <form class="" action="/" method="post">
    <input type="text" name="single" value="text" placeholder="テキスト">
    <input type="submit" name="" value="送信">
  </form>
  <p>From Server</p>
  <p>{{ text }}</p>
</body>
</html>
```

{{}}で囲んだところがマクロとして置換される

text 送信

From Server

{ "resourceType": "Patient", "address": [{ "postalCode": "3718511", "text": "群馬県前橋市昭和町3-39-15" }], "birthDate": "1970-01-01", "gender": "male", "identifier": [{ "value": "99999999" }], "name": [{ "extension": [{ "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation", "valueCode": "IDE" }], "use": "official", "text": "群馬 太郎", "family": "群馬", "given": ["太郎"] }, { "extension": [{ "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation", "valueCode": "SYL" }], "use": "official", "text": "グンマ タロウ", "family": "グンマ", "given": ["タロウ"] }], "telecom": [{ "value": "0272208773" }], "id": "1", "meta": { "lastUpdated": "2021-03-06T12:57:58Z", "versionId": "1" } }

ハンズオン２　PythonでDICOMデータを取得しHTMLに表示

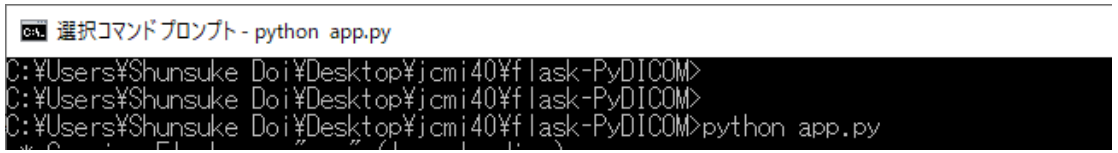
RESTで取得したデータを「ブラウザに表示する」「ファイルに出力する」方法

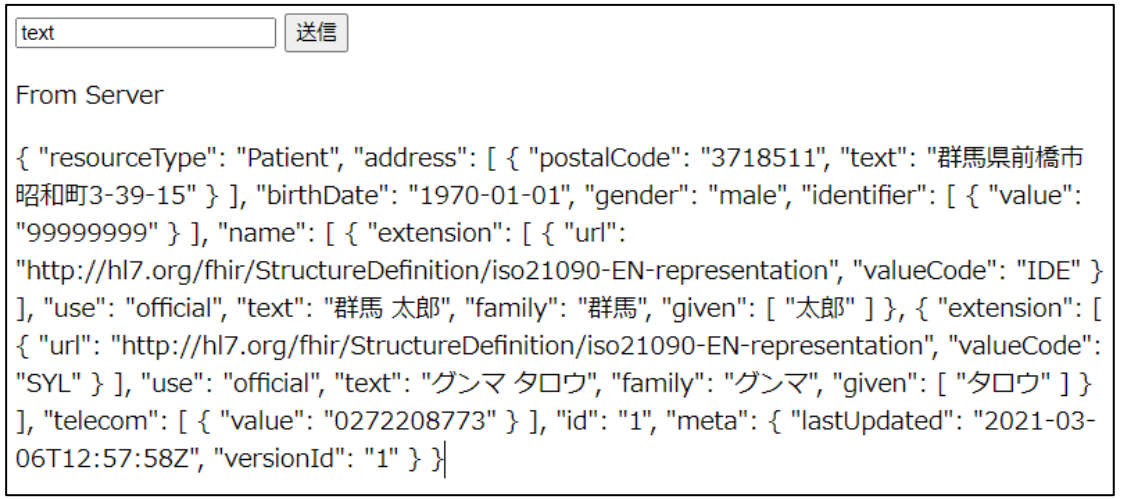
エディタにapp3.pyを表示

```
#!/usr/bin/env python3
↓
import requests
import json
from flask import Flask, render_template
import pprint
import os
import ssl
↓
↓
app = Flask(__name__)
↓
@app.route('/')
def fhir_rest():
↓
  ssl._create_default_https_context = ssl._create_unverified_context
↓
# RESTで取得するFHIRデータのURI
  resturl = 'https://fhirtestjp.med.gunma-u.ac.jp/Patient/1'
↓
  #FHIR serverからRESTでリソースを取得する
  headers = {"content-type": "application/json"}
  response = requests.get(resturl, headers=headers, verify=False)
  ↓
  jread = response.json()
  jdata = json.dumps(jread, indent=2, ensure_ascii=False)
↓
  path_w = './app3out.txt' #テキスト
  with open(path_w, mode='w') as f:
    f.write(jdata) #
↓
  #FHIR取得したリソースを「text」という変数に入れhtmlに送る
  html = render_template('index.html', text = jdata)
  return html
    ↓
if __name__ == "__main__":
        app.run()
```

index.htmlにjsonデータを表示する

②CMDで「python app3.py」を実行





ハンズオン3　PythonでDICOMデータを取得しHTMLに表示

RESTで取得したデータを「ブラウザに表示する」「ファイルに出力する」方法

③ http://localhost:5000/
にアクセスすると、JSONデータが表示される

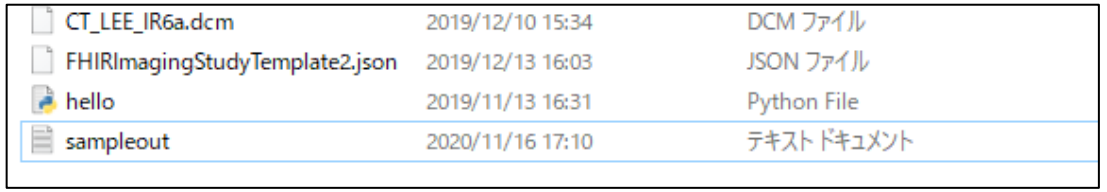
text 送信

From Server

{ "resourceType": "Patient", "address": [{ "postalCode": "3718511", "text": "群馬県前橋市昭和町3-39-15" }], "birthDate": "1970-01-01", "gender": "male", "identifier": [{ "value": "99999999" }], "name": [{ "extension": [{ "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation", "valueCode": "IDE" }], "use": "official", "text": "群馬 太郎", "family": "群馬", "given": ["太郎"] }, { "extension": [{ "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation", "valueCode": "SYL" }], "use": "official", "text": "グンマ タロウ", "family": "グンマ", "given": ["タロウ"] }], "telecom": [{ "value": "0272208773" }], "id": "1", "meta": { "lastUpdated": "2021-03-06T12:57:58Z", "versionId": "1" } }

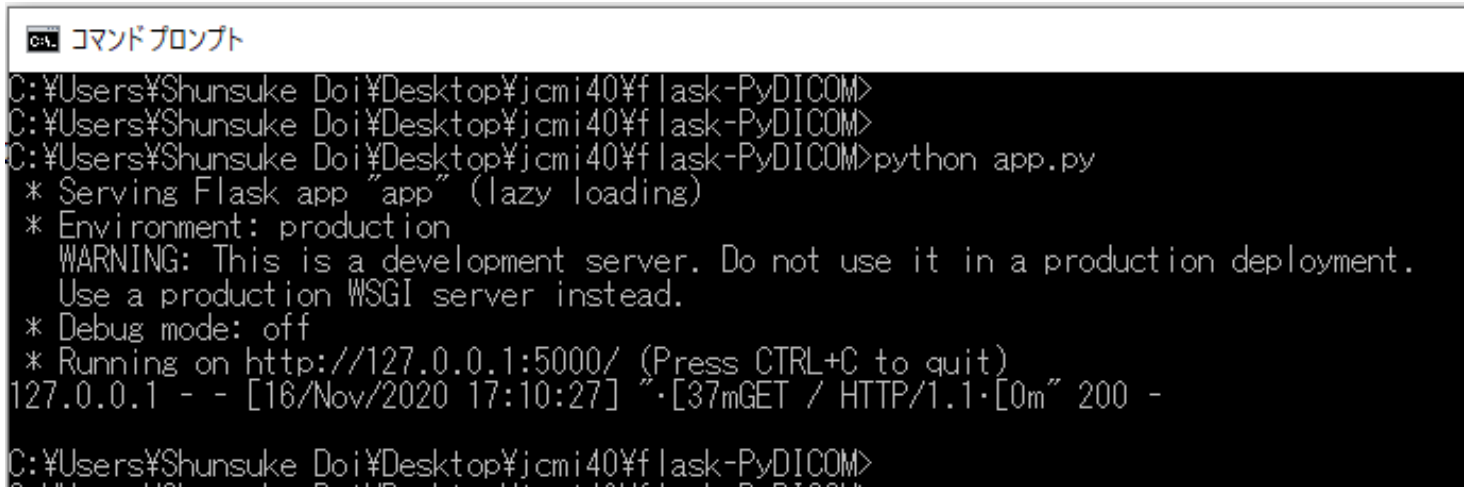
index.htmlにjsonデータを送信し、表示された結果

④作業フォルダにできた「app3out.txt」を開くと、①と同じ内容が表示されます。



ハンズオン3　PythonでDICOMデータを取得しHTMLに表示

確認が終わったら、「Ctrl + C」を押下することで終了できます。



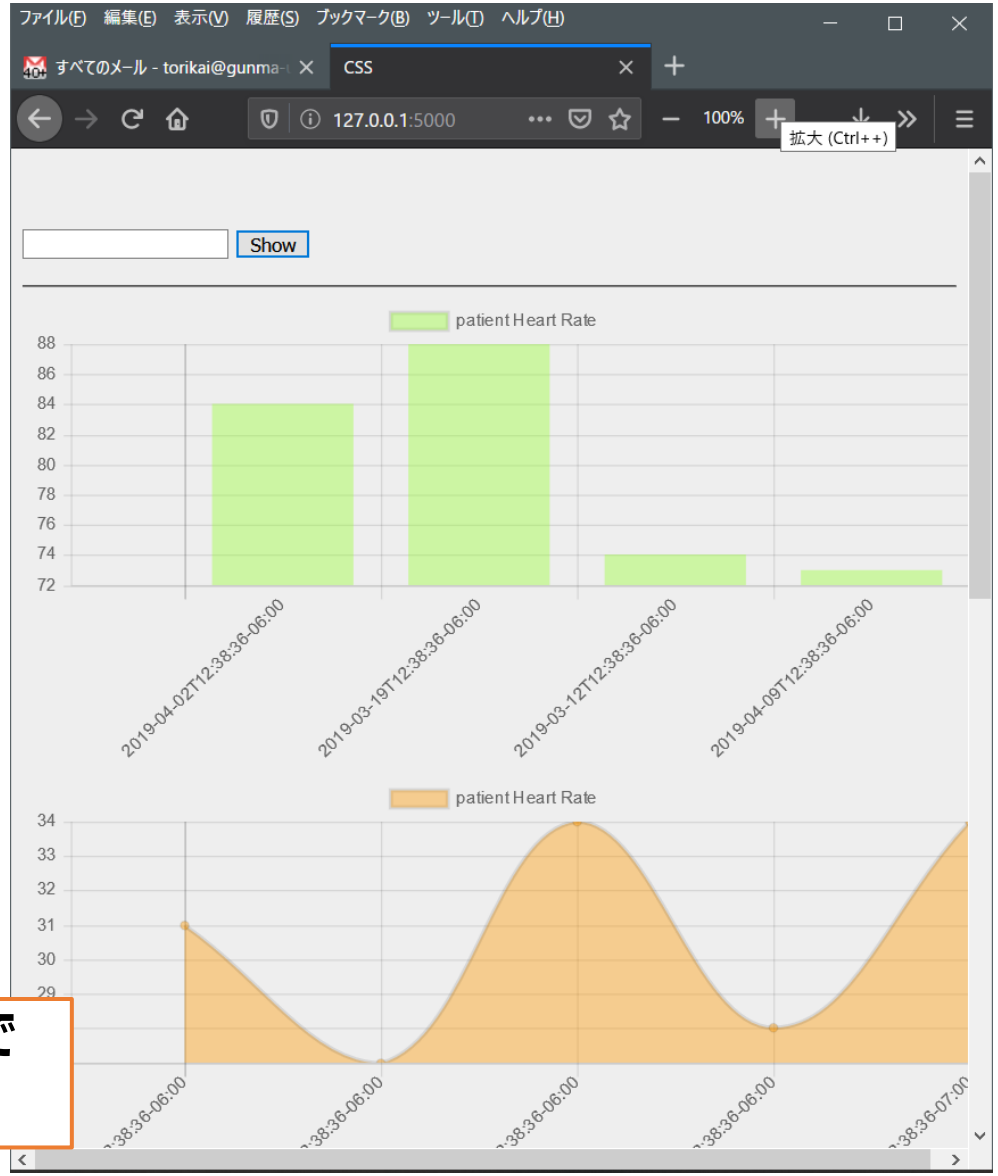
テキストファイルにFHIRデータをファイルに落とすことさえできれば、
Pythonの豊富なライブラリを活用することで様々な処理が可能になる

活用例：様々な帳票やダッシュボードの作成

データをFHIRリソースで表現できれば、Web標準技術で可能な様々なアプリや、様式なども自由に作成できる

開発の敷居が低いので、個別に必要なものをDIYで作成するのに適している

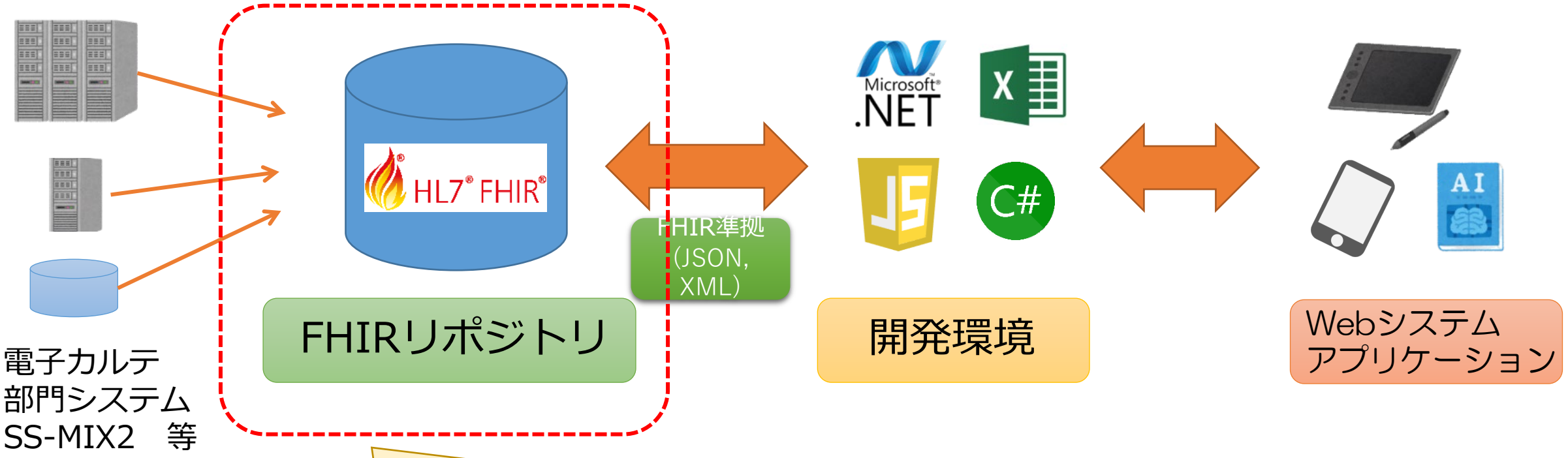
javascriptなどのWeb開発、pythonを利用した統計解析など、既存のライブラリをフル活用できるところもよい



以前のチュートリアルで作成したWEBで表示できる簡易検査結果モニタアプリ

ただし、良いことばかりではなく・・・

・サービスを現場に導入するまで（FHIRを利用）



サーバ側の構築はやっぱり大変
- 既存のデータベースとのマッピング（HL7規格であれば相互運用可能）
- オープン環境での構築も可能だが、時間と知識が求められる

FHIRは（当分の間）「完全な」規格ではない

- 発展途上であること
  - 現在のR4版からは正式版となっているが、リソースが「Normative」（＝確定版）となったものは**Patient（患者基本情報）とObservation（検査）**のみ
- 「自由度の高さ」は管理されなければ「自由奔放」になりかねない
  - **80%のシステムで実際に使われるであろう要素を収載（「80%ルール」と呼ばれる）**
  - 利用者が自由に構築できる以上、それ以外の要素が自由に拡張されかねない

日本医療情報学会では、昨年度「HL7 FHIR実装検討WG（NeXEHRS研究会）」「FHIR研究会」など複数の課題研究会が立ち上がり、日本におけるリソースのあり方や実装のユースケースなどの検討の場となることが期待されている

- Web標準技術であるが故に
  - **認証の管理やセキュリティ対策**はもちろん開発者に責任が生じる
  - 医療分野以外のベンダからの参入が期待できるとはいえ、医療情報の安全管理に関するガイドラインは最低限遵守されるよう、発注者側も管理しなければならない

FHIRの活用についてもっと知りたい方へ

シンポジウム5

11月20日（土）9:10～11:10　H会場

加速するFHIR活用とその課題

オーガナイザー：中山　雅晴（東北大学）
座長：土井　俊祐（東京大学大学院医学系研究科）
　　塩川　康成（キヤノンメディカルシステムズ株式会社）

3-H-1-01　HL7 FHIRの電子カルテシステム側への実装と課題
鳥飼　幸太・松山　龍之介（群馬大学医学部附属病院）

3-H-1-02　HL7 FHIR導入の院内実装および地域連携における課題
田中　良一（岩手医科大学）

3-H-1-03　PHRの実装における課題
木村　映善（愛媛大学）

3-H-1-04　医療情報銀行を用いた医療機関から患者への情報返却
武田　理宏（大阪大学大学院医学系研究科医療情報学）

3-H-1-05　SS-MIX2データを活用するためのFHIR®ベースPHRの開発
中山　雅晴（東北大学大学院医学系研究科医学情報学）

総合討論

チュートリアルA-2

開催日時	2021年11月21日（日）９時10分～11時10分　※受付開始８時40分
開催会場	D会場（名古屋国際会議場２号館１階・展示室212）
オーガナイザー	大江　和彦
主催団体（オーガナイザー所属）	FRUCtoS Japan／東京大学
テーマ	FHIRサーバ「OpenFRUCtoS」の概要と使い方
座長	大江　和彦（東京大学大学院医学研究科）
演者	大江　和彦（東京大学大学院医学研究科） 小西　由貴範（株式会社ケーアイエス） 土井　俊祐（東京大学医学部附属病院）
参加申込方法	事前申込制です。現地参加、Web参加、どちらの場合でも次のページに掲載のWeb参加申し込みをしてください。直前まで申し込み可能です。
事前参加申込URL	https://nexehrs.jp/jcmi41-tutorial-a2/

FHIRのユースケースを検討する課題研究会「FHIR研究会」の主催するシンポジウムです

オープンソースのFHIRサーバ「OpenFRUCtoS」を紹介します

次世代健康医療記録システム共通プラットフォーム課題研究会

日本医療情報学会（JAMI）

・　発展目覚ましい新しい技術を柔軟に活用できる新たな健康医療記録のありかたを検討する。
・　これまでの標準化基盤をベースにして、これらの技術にも対応していく新しい電子カルテシステムの共通プラットフォームを設計する。
・　来たるAI/IoT時代の次世代標準健康医療記録システムの基本コンセプト、共通プラットフォームのあり方、医療制度と法制度の課題も含めて議論する。

代表幹事　大江和彦 東京大学大学院医学系研究科医療情報学分野・JAMI常任幹事・前JAMI代表理事
幹事　黒田知宏　京都大学医学部附属病院医療情報企画部・JAMI理事
幹事　澤 智博　帝京大学医療情報システム研究センター・JAMI理事
幹事　中島直樹　九州大学病院メディカル・インフォメーションセンター長・JAMI代表（学会長）
幹事　松村泰志　大阪大学大学院医学系研究科医療情報学・JAMI評議員

次世代健康医療記録システムNeXEHRSのコンセプト

■　**３つの基本コンセプト**　Patient-centered, Sharable, Co-welfare

1. **本人主体管理**：個人に基づく健康医療情報は医療提供機関単位ではなく、本人（患者等）単位で１記録とし、そのバックアップコピーを恒常的に預かる組織が運用されることを前提とする。
2. **本人・医療提供者間での情報共用**：本人と医療提供者は、医療時に医療情報を共用する（明示的に拒否する場合を除く）

イベントカレンダー

2019年 7月

日	月	火	水	木	金	土
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	**19**	20
21	22	23	24	25	26	27
28	29	30	31			

祝日
イベント

新着情報

- 7月8日シンポジウムページに厚労省資料を追加しました。
- 設立趣意書・規約案
- コンソーシアム設立説明会 2019.7.16 資料掲載済み

FHIR研究会は医療情報学会内でFHIRの実装、活用を主として検討、提案する研究会です。

FHIR®研究会

Since 2019

トップページ | 活動内容 | 新着情報・FAQ | お問い合わせ | アクセス | 団体情報 | その他情報

トップページ

相互運用性を確保しつつ、医療現場のニーズに真に対応できるHL7 FHIRの実装、活用を中心とした研究ならびに提案を行っています。

トピックス

- FHIR®研究会は日本HL7協会、NeXEHRS研究会と協力してHL7 FHIR®の普及推進に努めてまいります。
- FHIR®研究会は日本医療情報学会の課題研究会として組織されました。

ニュース

2019年11月12日 2019/11/11HL7 FHIRに関する有識者会議に出席しました

ブログ

活動ブログを開設しました

2019/9/18FHIR研究会（予定）

2019/10/27岩手医療情報研究会と共催でFHIR研究会を開催しました

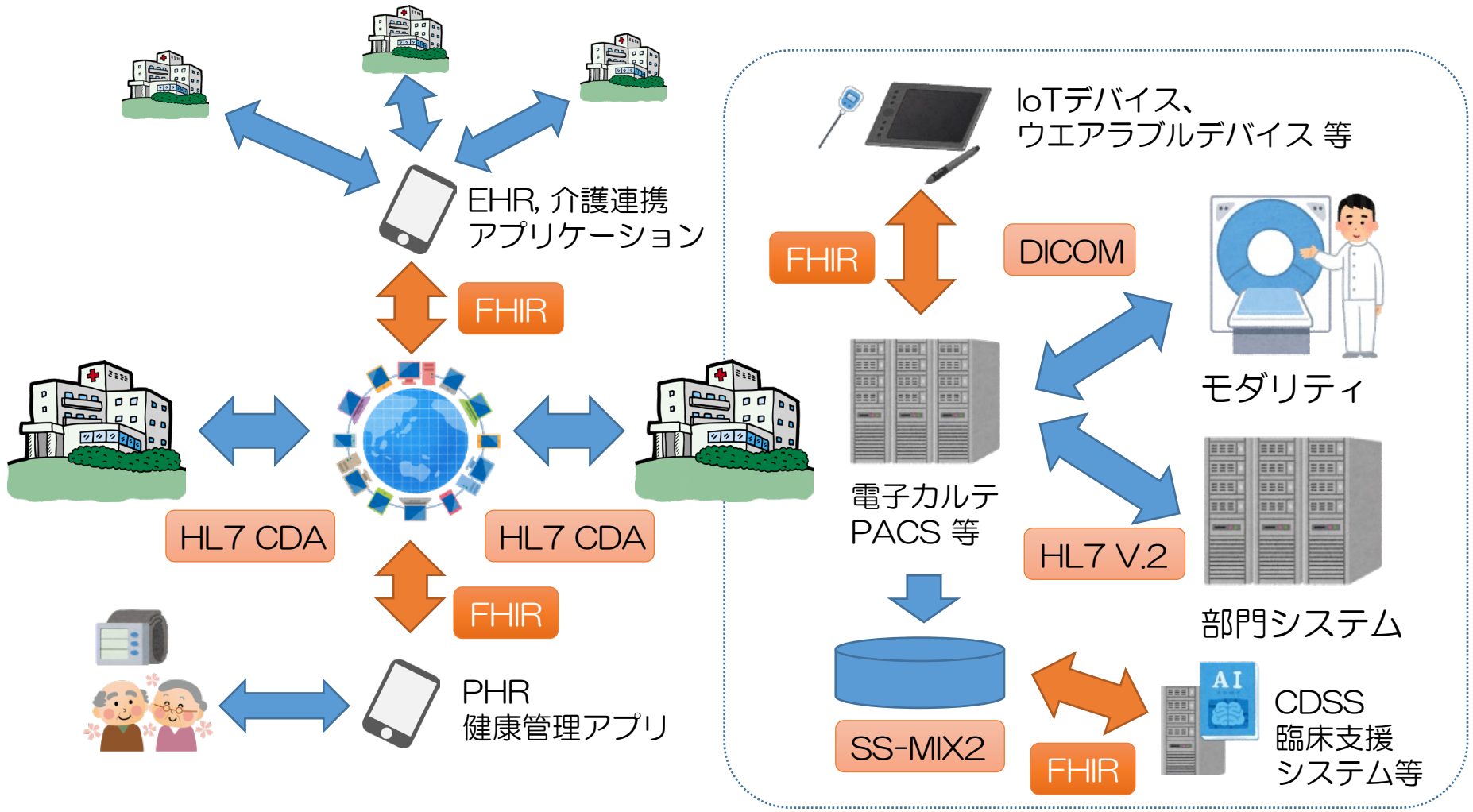
一般社団法人　日本医療情報学会(監修)

大江　和彦(監修 | 翻訳)

岡田　美保子(監修 | 翻訳)

澤 智博(監修 | 翻訳)

今後のシステム構築の形



- **現在の標準規格を利用しながら、これまで連携が難しかったデバイスや利用者との接続を補完し、さらにより広いベンダの参入を可能に**

日本Mテクノロジー学会について

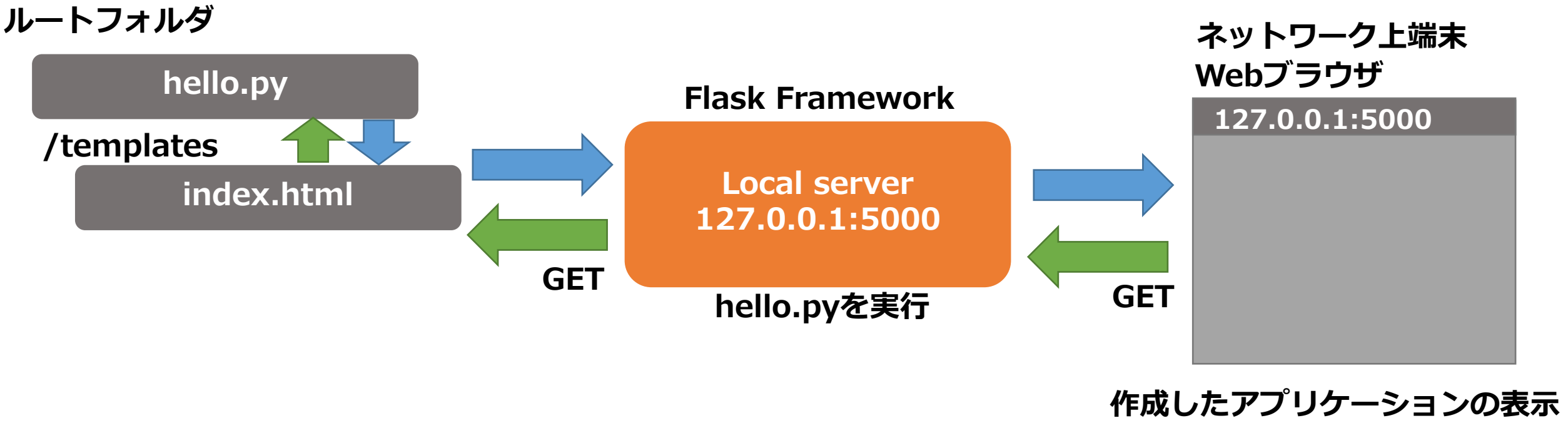
- (主に)医療データベース、プログラミング等に関連する領域の利用、応用、改良、及び普及を行うことを目的とした団体です。
  - 現代のMテクノロジーは関数型のプログラム、ツリー型とテーブル型の両方式のDBを統合できるオブジェクト指向型の開発環境です。
  - より良い医療システムアーキテクチャを探究しています。

- プログラミングやデータベースの技能・知識を持った情報部門担当者の育成を目的としたチュートリアルを年3回実施します
  - 学術部会（主に大学・病院関係）と技術部会（主にベンダ関係）が中心
  - 年次大会ではユーザとベンダのそれぞれの立場からの学術発表、技術討論、チュートリアル等を行い、会員のレベルアップを図っています。

ご清聴ありがとうございました

参考資料

掲示板プログラムの構造



掲示板は1分毎に更新するように動作する
＊ユーザーが操作しなくても、情報が機器側から示され、更新されることがポイント

pipによるFlaskのインストール

pip install flask
でインストールされる

コードの説明

run.py

```
1  from flask import Flask
2  app = Flask(__name__)
3
4  @app.route('/')
5  def hello_world():
6      return '<html><body><h1>sample</h1></body></html>'
7
8  if __name__ == '__main__':
9      app.run()
```

アドレス位置の指定

表示文字列の指定

Pythonによるファイル出力

Pythonでファイル出力を行う方法

app5.py - TeraPad

ファイル(F) 編集(E) 検索(S) 表示(V) ウィンドウ(W) ツール(T) ヘルプ(H)

```
app = Flask(__name__)

@app.route('/')
def hello_world():

    headers = {"content-type": "application/json"}
    response = requests.get(

        'http://hapi.fhir.org/baseDstu3/Observation',

        params={'patient': '1723865'},
        headers=headers)

    jread = response.json()
    jdata = json.dumps(jread, indent=2)

    path_w = 'c:/python/sampleout.txt'
    with open(path_w, mode='w') as f:
     f.write(jdata) # 問題なく出力される

    html = render_template('index4.html', a = jdata)
    return html

if __name__ == "__main__":
        app.run()
[EOF]
```

出力先を指定

jdataを出力

PythonにおけるREST呼び出しの最も単純な形

サンプルコード

hapifhirtest0.py - TeraPad

ファイル(F) 編集(E) 検索(S) 表示(V) ウィンドウ(W) ツール(T) ヘルプ(H)

```
import pprint
import requests
import matplotlib.pyplot as plt
import math
import numpy as np
import json

def main():

    response = requests.get(

        'http://hapi.fhir.org/baseDstu3/Observation',

        params={'patient': '1723865'})


    jread = response.json()

    jdata = json.loads(json.dumps(jread))

    pprint.pprint(json.dumps(jread))

if __name__== '__main__':

    main()
[EOF]
```

RESTのrequest処理

JSON処理に必須

FHIRを返すサイト

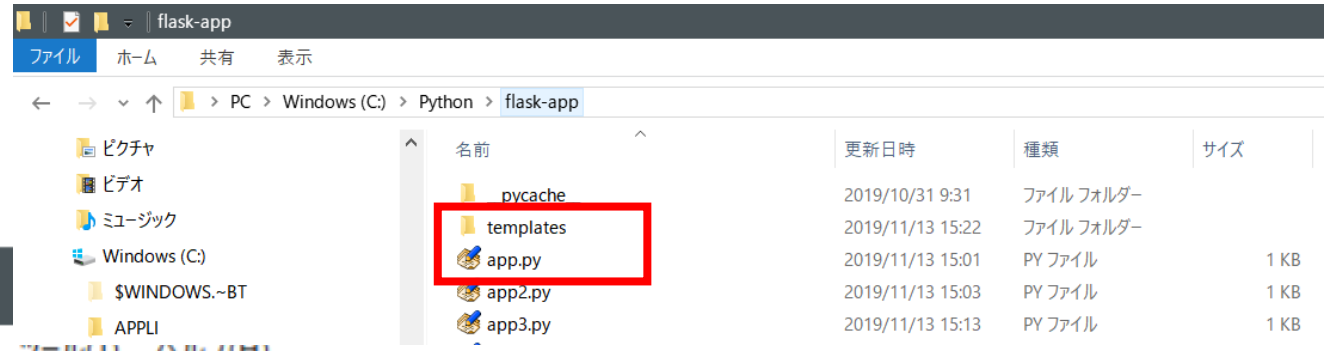
RESTのパラメータ入力方法

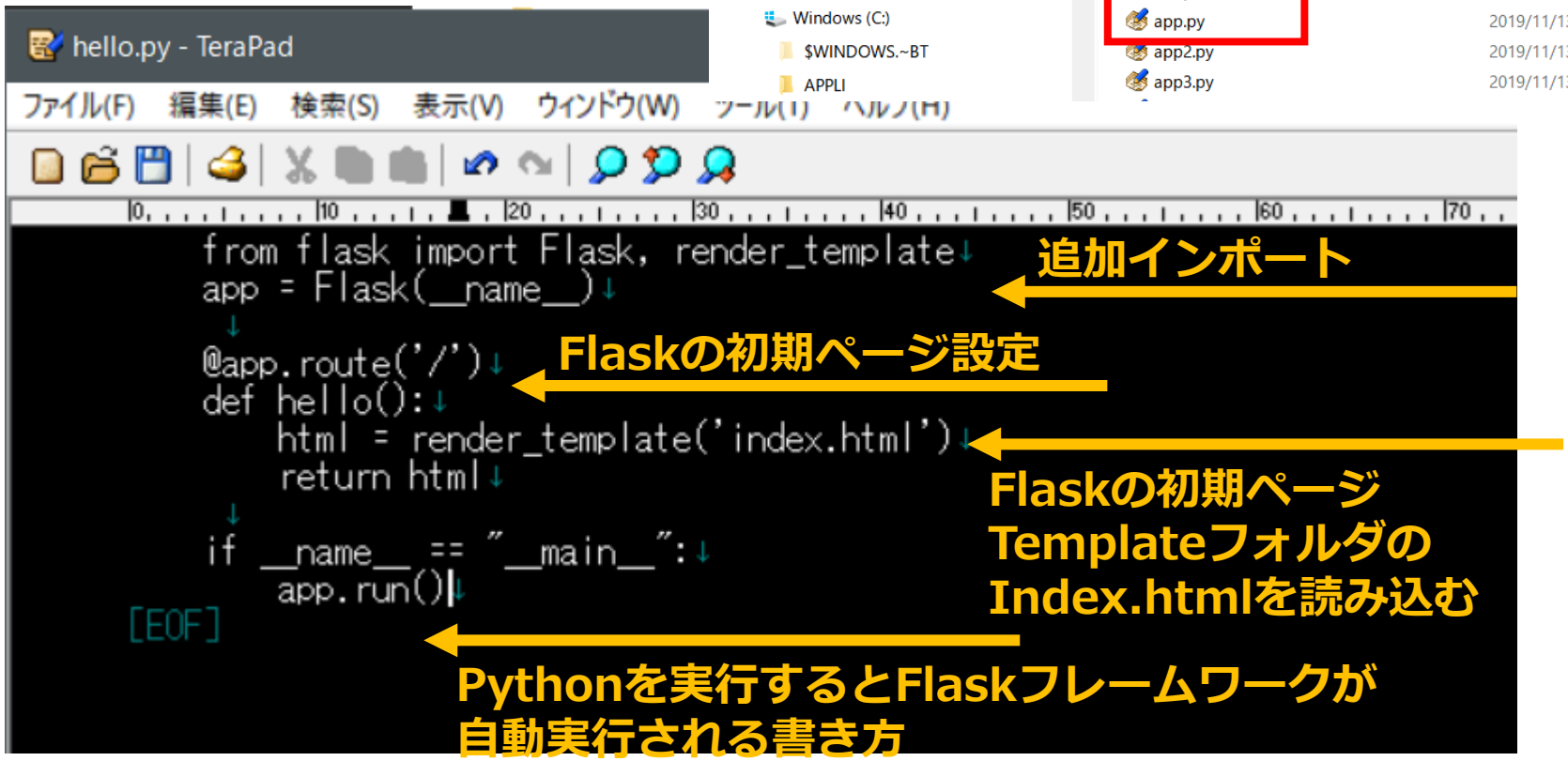
JSON文字列に変換

JSON文字列としてコンソール出力

pyコードによるFlaskのデフォルトページの指定

templatesフォルダ内の”index.html”を参照させる
※名前は必ず*.py直下の”templates”にすること





コマンドプロンプトから python hello.pyで実行

FlaskフレームからHTMLへのデータの動的渡し

```python
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/')
def hello():
    html = render_template('index.html', a = '変数なう')
    return html

if __name__ == "__main__":
    app.run()
```

```python
a = '変数なう'
```

文字列変数を割り当てる

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>HELLO</title>
</head>
<body>
    <p>Sayhello</p>
    <p>{{a}}</p>
</body>
</html>
```

```html
<p>{{a}}</p>
```

{{}}で囲んだところが
マクロとして置換される

PythonにおけるJSON出力の取り扱い

Pythonでは「辞書型」というデータ型が存在する

そのまま書き込む場合

```
dic = {"A" : 1, "B" : 2}
print(dic)
# 出力
# {"A" : 1, "B" : 2}
```

空の辞書に書き込んでいく場合

```
dic = {}
dic["key"] = "value"
print(dic)
# 出力
# {"key" : "value"}
```

valueにはリストも取れます

```
dic = {}
dic["key"] = ["value1", "value2"]
print(dic)
# 出力
# {"key" : ["value1", "value2"]}
```

```
dic["key"] = "value"
```

JSONならdic.keyとするところ

https://qiita.com/E_Schaf/items/9ef26c0c514325d69e80

Flask-HTMLにおけるJSON渡しの注意点

index4.html - TeraPad

ファイル(F) 編集(E) 検索(S) 表示(V) ウィンドウ(W) ツール(T) ヘルプ(H)

```
                } 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]);
                        }
                }
        }

        function onButtonClick() {
          target = document.getElementById("output");

          let insertLabels = '';
          let insertDatas = '';
          let insertLabels2 = '';
          let insertDatas2 = '';

          var predata = {{ a | safe }};
          var data = predata; //JSON.parse(predata);//もともとJSONの書式で値を記述するので、パースは不要
          //要素の求め方 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 + "',";
                                insertDatas += "'" +  data.entry[i].resource.valueQuantity.value + ",";
                                if(i < (data['entry'].length - 1)){
                                        insertLabels += ",";
                                        insertDatas += ",";
                                }
                        }
                        if (data.entry[i].resource.code.text == 'Respiratory Rate'){
                                insertLabels2 += "'" + data.entry[i].resource.effectiveDateTime + "',";
```

```
var predata = {{ a | safe }};
```

| safeがないと、" が %39;になってJSONとして認識されない

PythonにおけるWebアプリとしてのグラフ出力(1)

VBAでJSON!のHTMLを援用

```
<head>
<script type="text/javascript" language="javascript">
  function onButtonClick() {
      var data = {{ a | safe }};
          for (  var i in data.entry  ) {
              if (data.entry[i].resource.code.text == 'Heart Rate'){
                  insertLabels += """ +    data.entry[i].resource.effectiveDateTime + """;
                  insertDatas += """ +  data.entry[i].resource.valueQuantity.value + """;
                  if(i < (data['entry'].length - 1)){
                      insertLabels += ",";
                      insertDatas += ",";
                  }
              }
 var outtext =  '<br> patient id =' + data.entry[0].resource.subject.reference + '<br> elements.No = ' +
data['entry'].length + '<br>' +extractdata  + '<br>' + predata;
      target.innerHTML = outtext;
onShowChart2(insertLabel, insertData);
```

PythonにおけるWebアプリとしてのグラフ出力(2)

VBAでJSON!のHTMLを援用

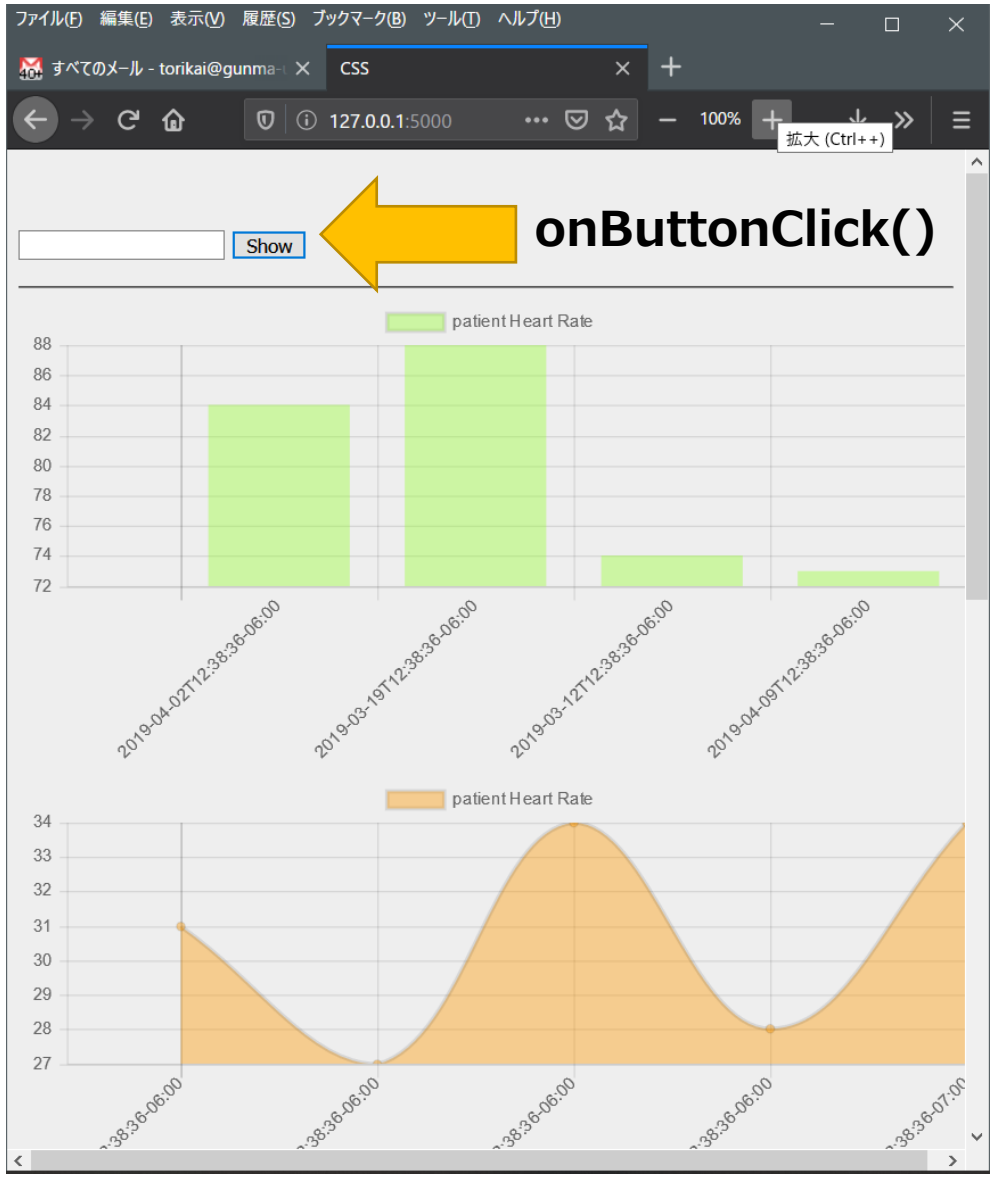
```
function onShowChart2(insertLabel, insertData){

  var chartheader = document.createElement("script");
  chartheader.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(chartheader);
}
```

PythonにおけるWebアプリとしてのグラフ出力(3)

VBAでJSON!のHTMLを援用

```
<body>
 <form name="form1" id="id_form1" action="">
  <input name="textBox1" id="id_textBox1" type="text" value="" />
  <input type="button" value="Show" onclick="onButtonClick();" />
 </form>
 <hr/>


</body>

 <body>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.1.4/Chart.min.js"></script>
  <canvas id="myChart"></canvas>
</body>
```

PythonにおけるWebアプリとしてのグラフ出力(4)

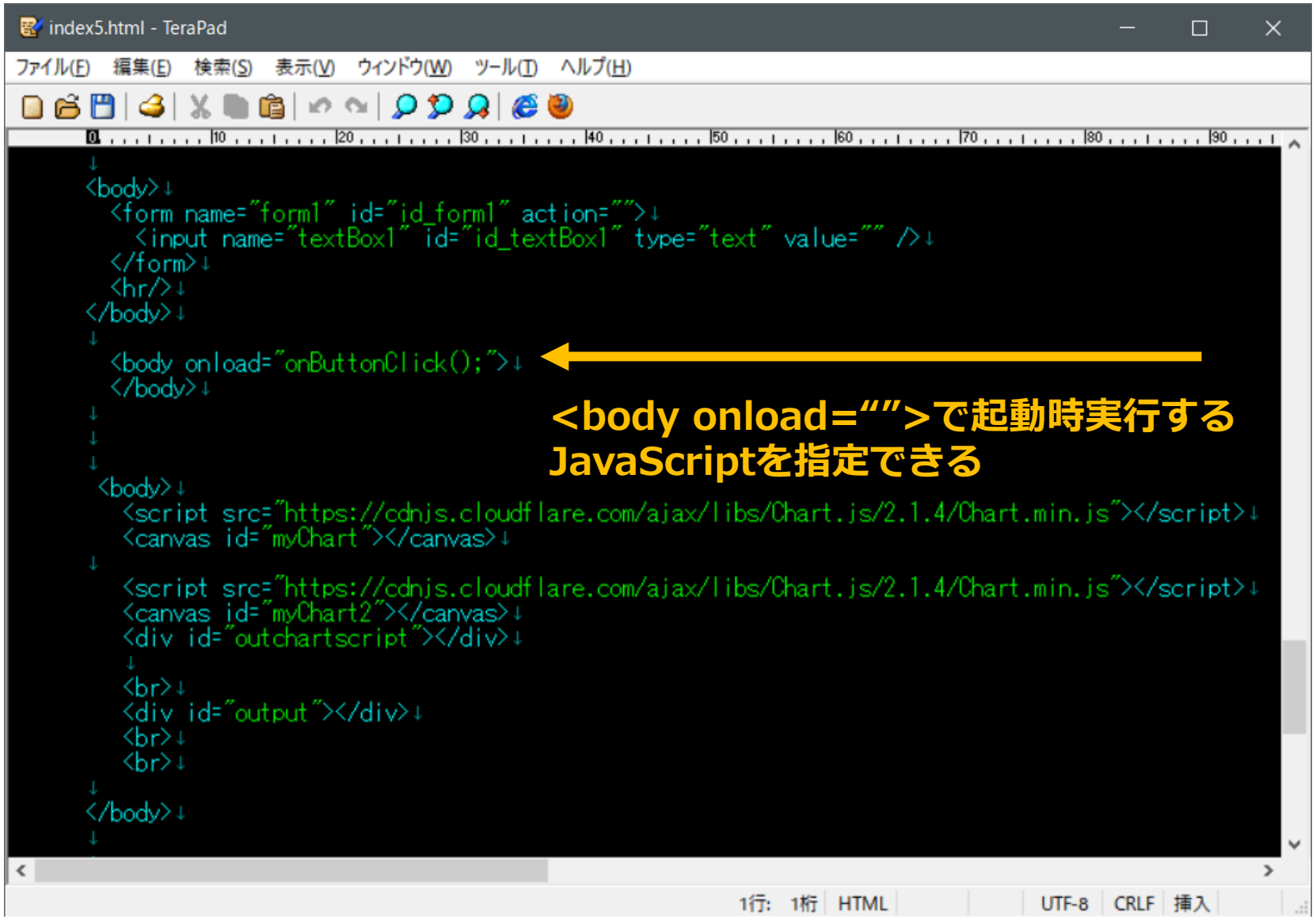


JSONから
insertLabel, insertDataを生成

```
function onShowChart2(insertLabel, insertData){
document.getElementById('myChart').getContext('2d');"
target = document.getElementById("outchartscript");
target.appendChild(chartheader);
```

```
<canvas id="myChart">
<div id="outchartscript">
```

ボタンを押さないルーチンへの変更



活用例：Flaskのルーティング

app41.py - TeraPad

```
import requests
import json
from flask import Flask, render_template
import urllib.parse

app = Flask(__name__)


@app.route('/')
def hello_world():

    headers = {"content-type": "application/json"}
    response = requests.get(

        'http://hapi.fhir.org/baseDstu3/Observation',

        params={'patient': '1723865'},
        headers=headers)

    #result.encoding = 'UTF-8'

    jread = response.json()
    jdata = json.dumps(jread, indent=2)

    path_w = 'c:/python/sampleout.txt'
    with open(path_w, mode='w') as f:
     f.write(jdata) # 問題なく出力される

    #jdata2 = urllib.parse.unquote(jdata)
    html = render_template('index4.html', a = jdata)
    return html


@app.route('/hello')
def hello():
    return 'Hello World'

@app.template_filter('cr')
def cr(arg):
    return arg.replace('&#34;', '"')

if __name__ == "__main__":
        app.run()
[EOF]
```


```
@app.route('/hello')
def hello():
    return 'Hello World'

@app.template_filter('cr')
```

”の中にルーティングを設定できる

活用例３：スマートフォンによる表示

Flask公式サイトより

外部から見ることができるサーバー:

サーバーを走らせた時、そのサーバーが外部ネットワークからではなくあなたのコンピュータだけで利用できていることを確認しましょう。外部から利用できないのはデフォルトの設定になっています。デバッグモードでそのアプリケーションのユーザーは誰でもあなたのコンピュータ上でPythonコードを恣意的に実行できてしまうからです。もし *debug* モードを使用不可にするか、ネットワーク上の全ユーザを信頼するなら、サーバーを公共利用にすることも可能です。

`run()` メソッドの宣言を以下のように変更してください。

```
app.run(host='0.0.0.0')
```


```
app.run(host='0.0.0.0')
```


これはオペレーティングシステムがパブリックIPを参照する指定です。

発展的活用：QRコードによるスマートフォンでの利用

QRコードを掲示して、スマートフォンからの読み込みを行う

https://qr.quel.jp/form_bsc_url.php