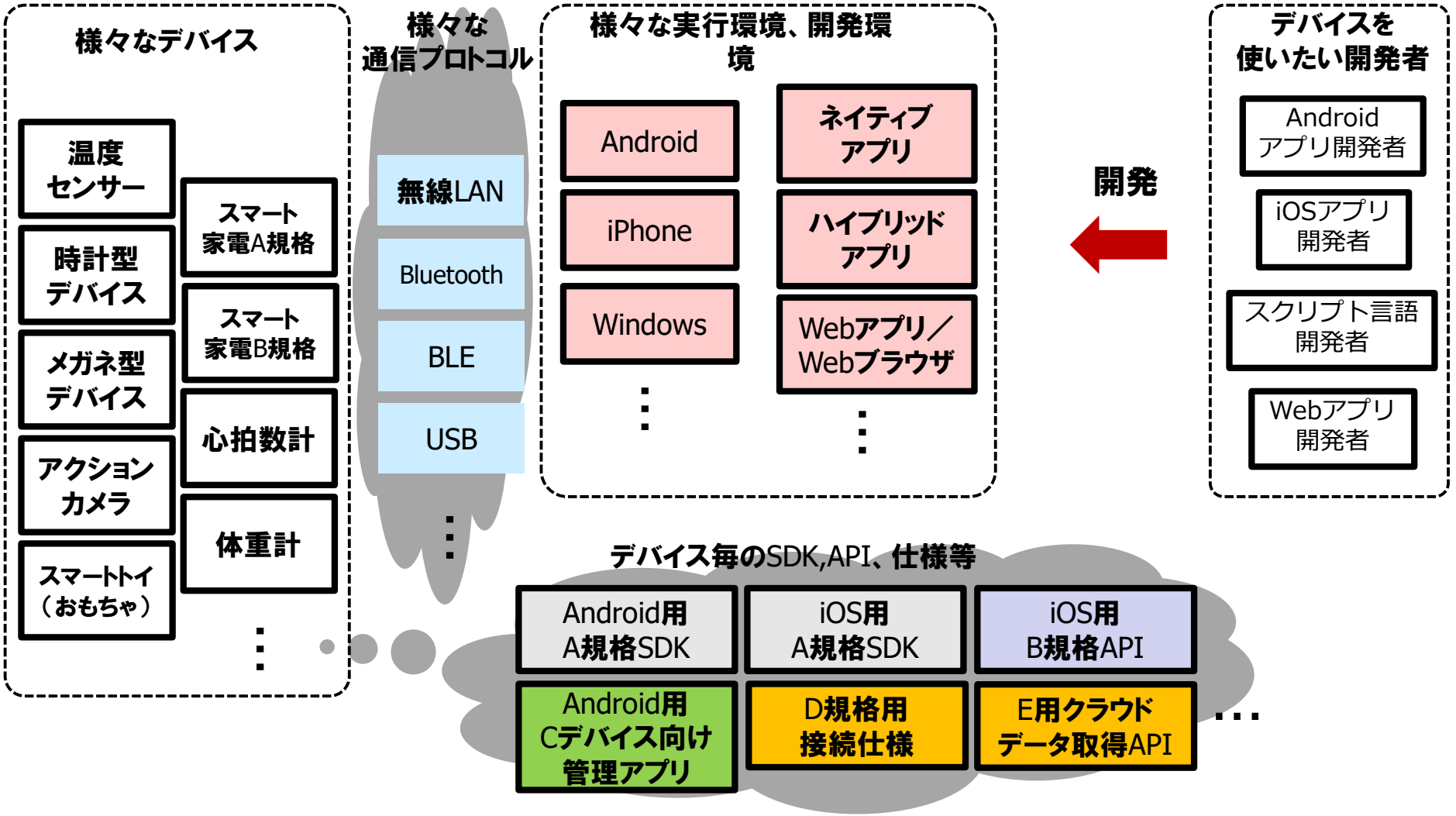
デバイスWebAPI設計の進め方

2017/7/14
株式会社NTTドコモ

デバイスWebAPI技術のおさらい

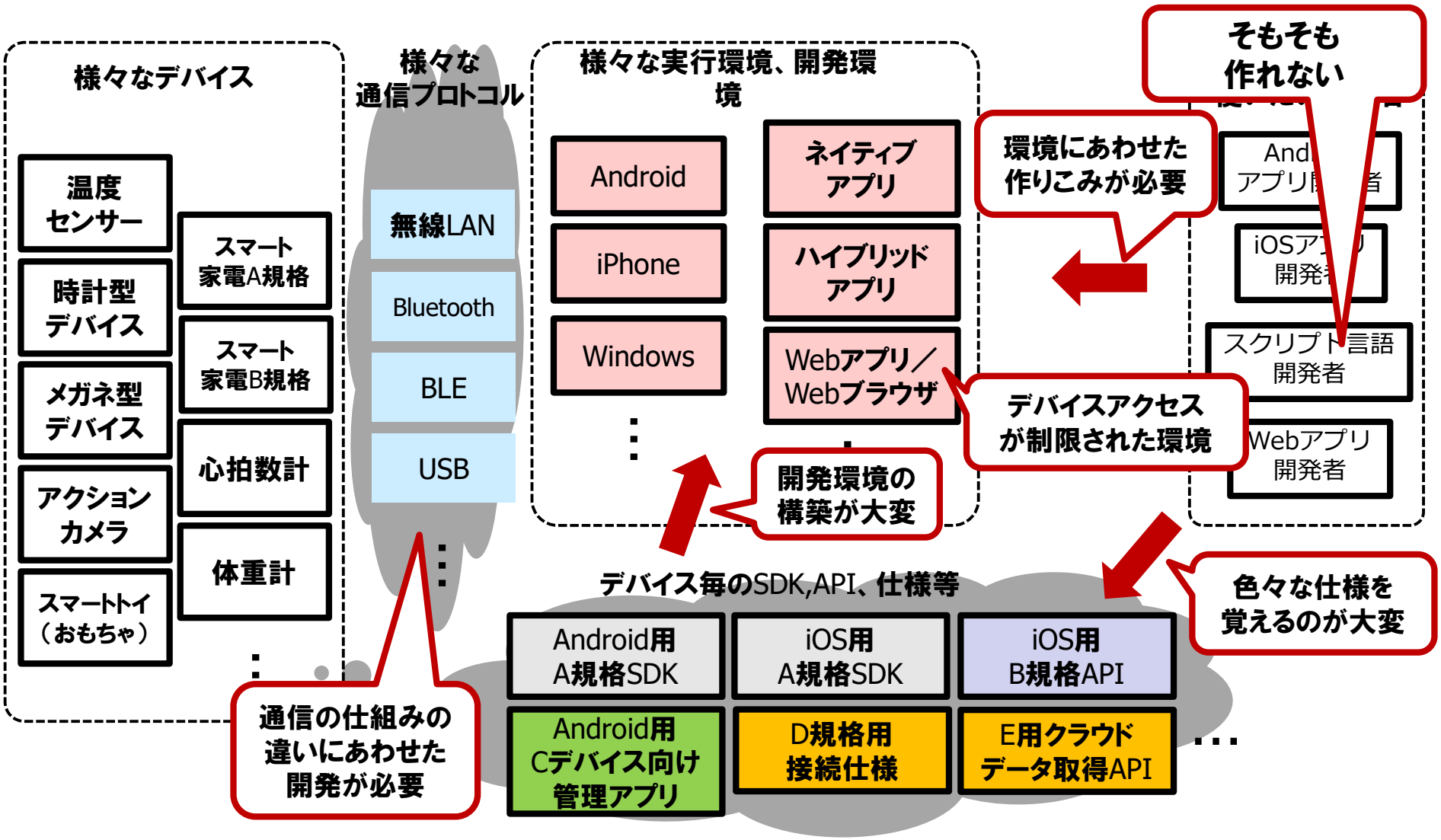
スマートフォンにつながる様々なデバイスの現状

- ひとつひとつのデバイスや規格にあわせ、それぞれの環境での開発が必要



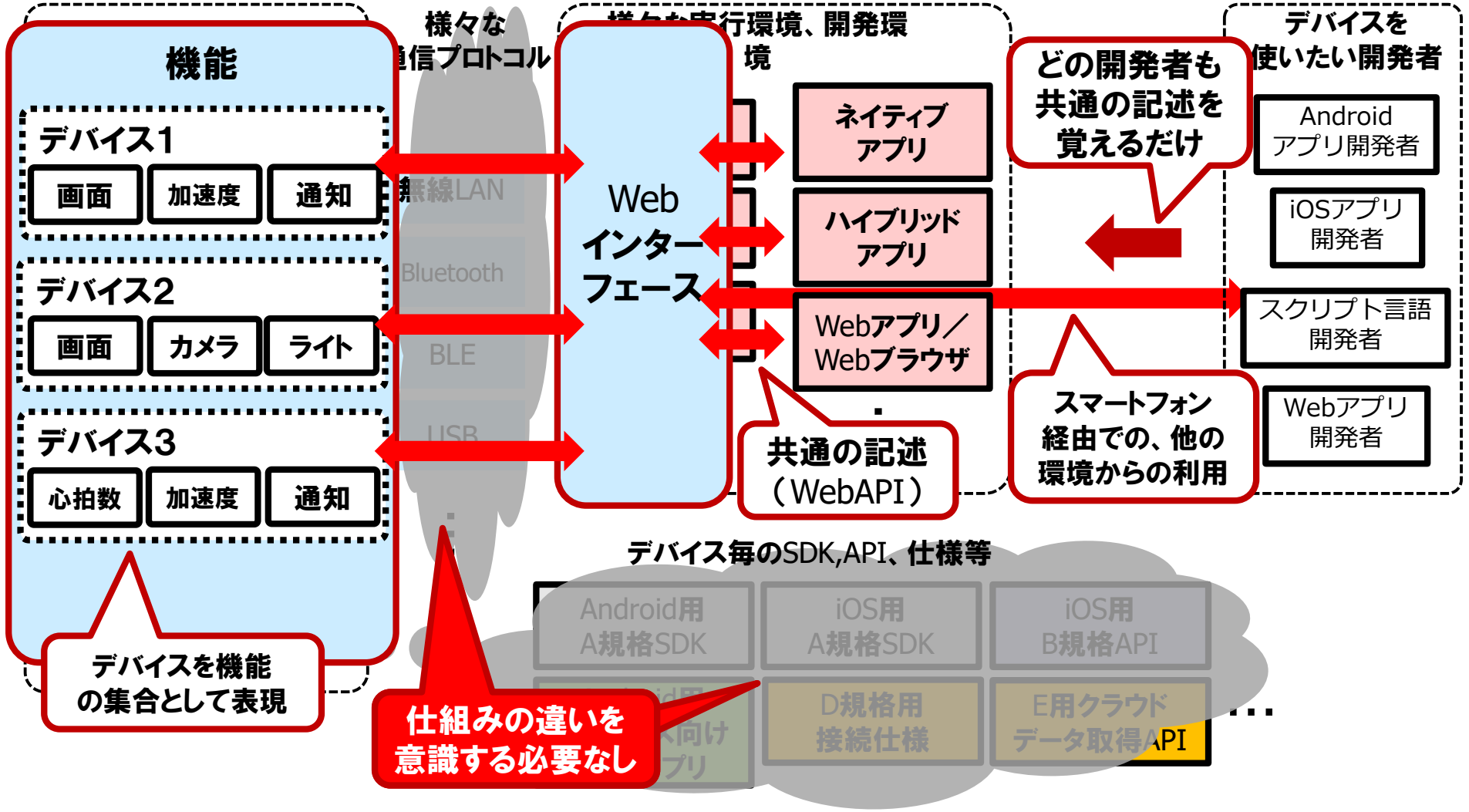
スマートフォンにつながる様々なデバイスの現状

- ひとつひとつのデバイスや規格にあわせ、それぞれの環境での開発が必要



デバイスＷｅｂＡＰＩとは？

- デバイスの持つ機能に、共通の記述（WebAPI）でアクセスする仕組みで現状の課題を解決



参考：そもそもＷｅｂＡＰＩとは？

Web API【 Web Application Programming Interface 】ウェブAPI

Web APIとは、コンピュータプログラムの提供する機能を外部の別のプログラムから呼び出して利用するための手順・規約（API：Application Programming Interface）の類型の一つで、HTTPなどWebの技術を用いて構築されたもののこと。

IT用語辞典e-words（http://e-words.jp/w/Web_API.html）より

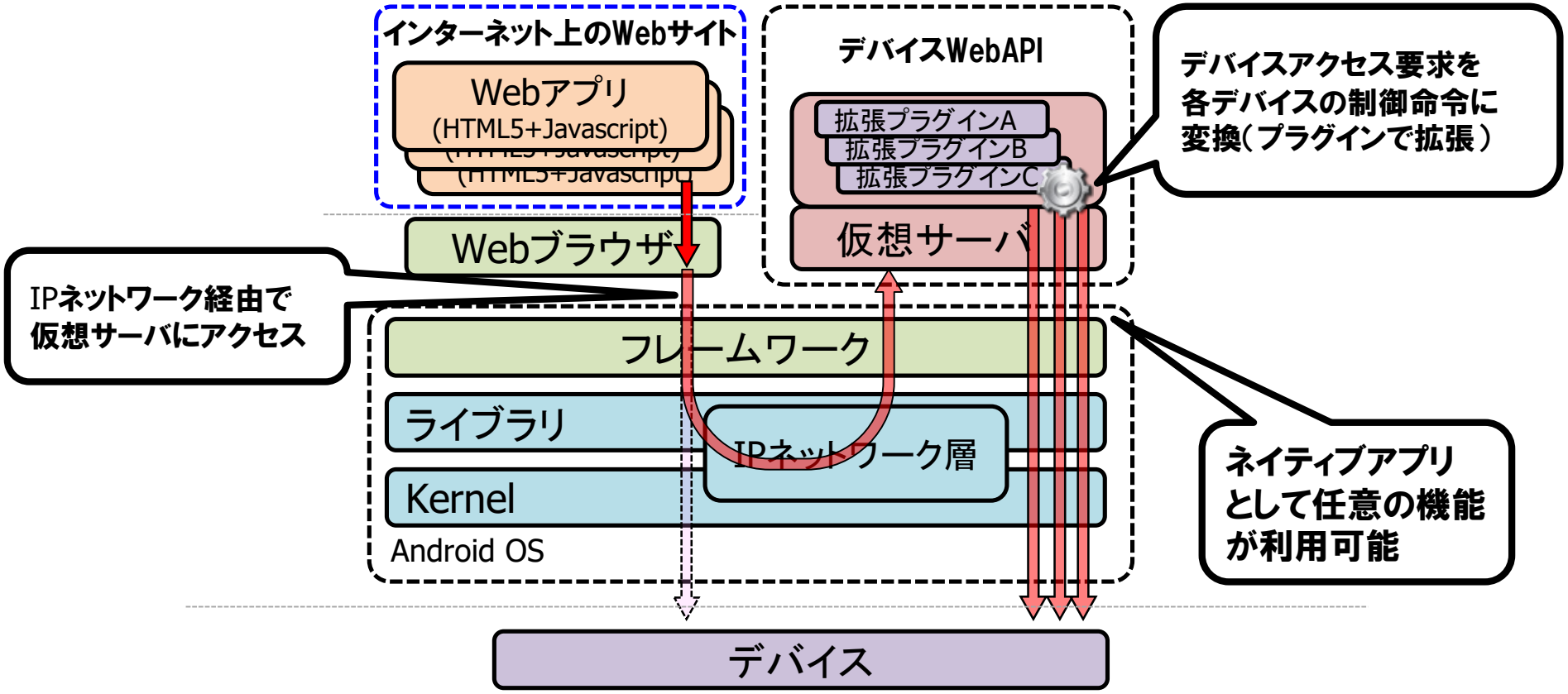
WebAPI（HTTPリクエスト）記述のお作法
- RESTfulという設計思想に従うのが一般的だが、細かいところでは諸説あり
- UpperCamelCase、lowerCamelCase、snake_case
- サブドメインとして表すか、パスとして表すか
- バージョン表記（整数、小数、つけるかどうか）
- APIの粒度（パスとして表すか、リクエストで送る中身（JSON等）に書くか）

HTTP GET
POST
PUT
DELETE
：

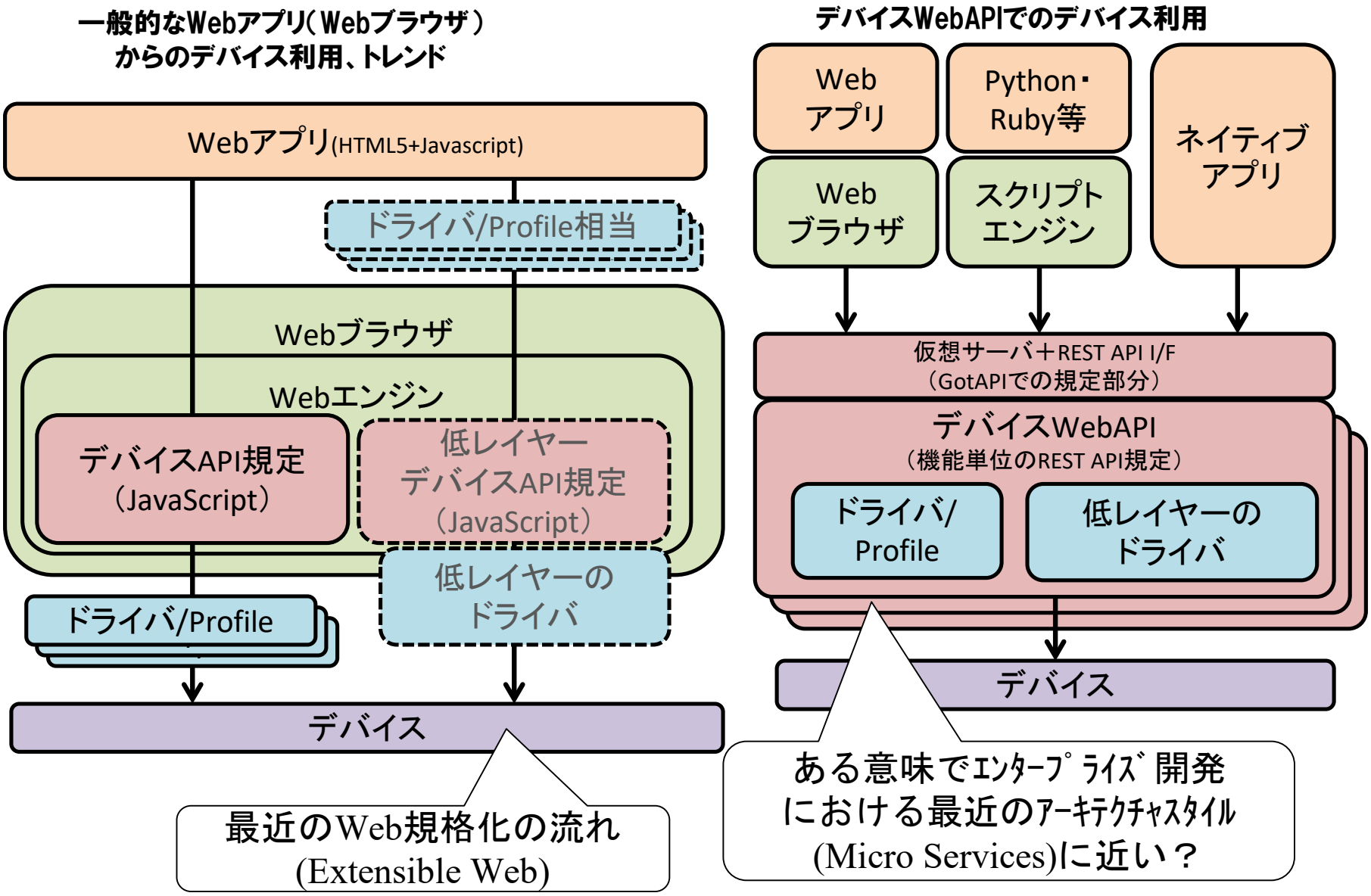
JSON、XML、HTML、
ﾘｿｰｽ(画像データ等)、
WebSocket等

どうやって実現しているか

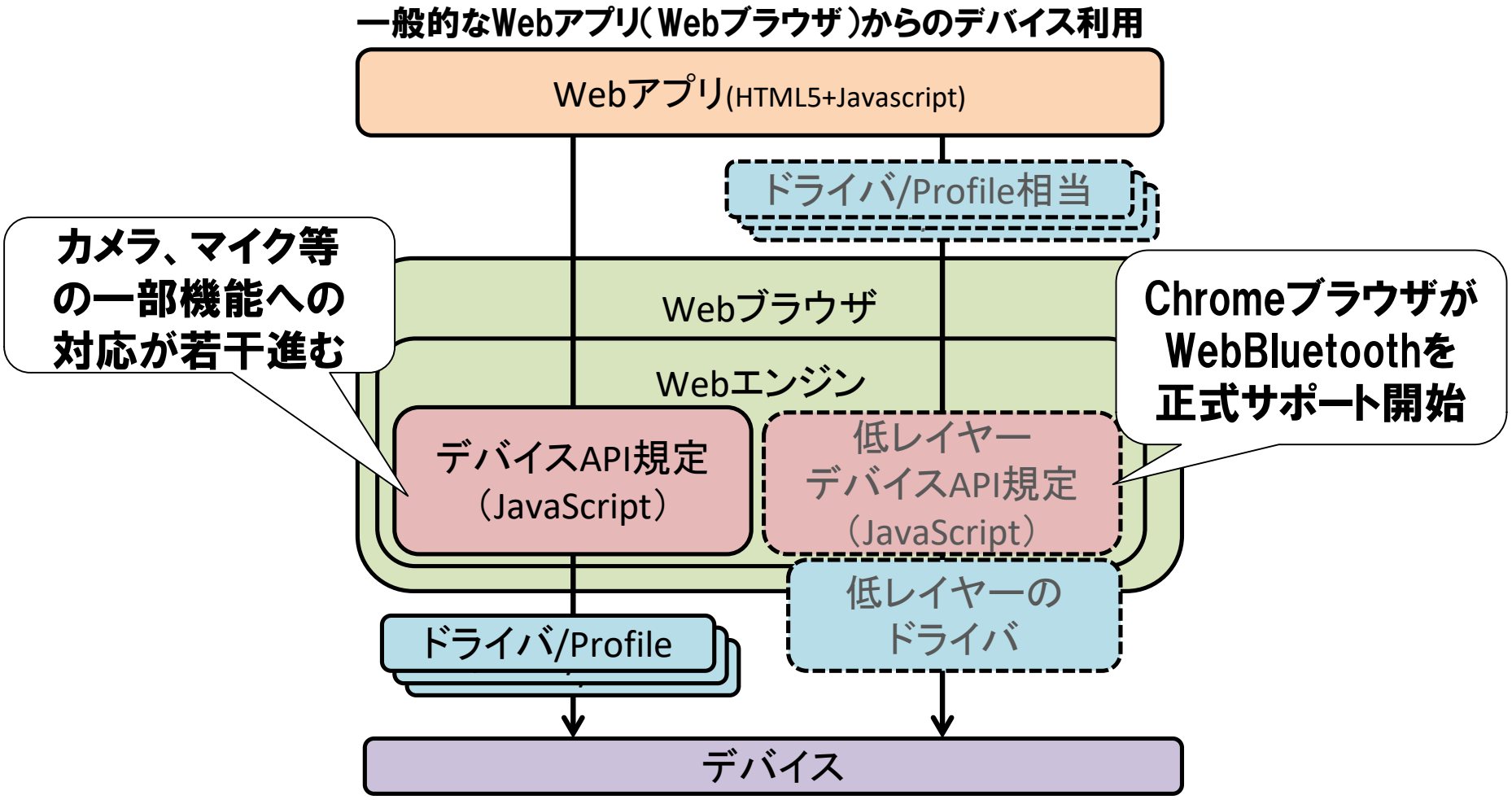
- スマートフォン上で動作する仮想サーバに、スマートフォン内部のIPネットワーク層を経由することで、Webブラウザからでも高度な機能アクセスを実現
- ネイティブアプリからも同様に利用可能



Webのトレンドで見た場合※2年前の資料

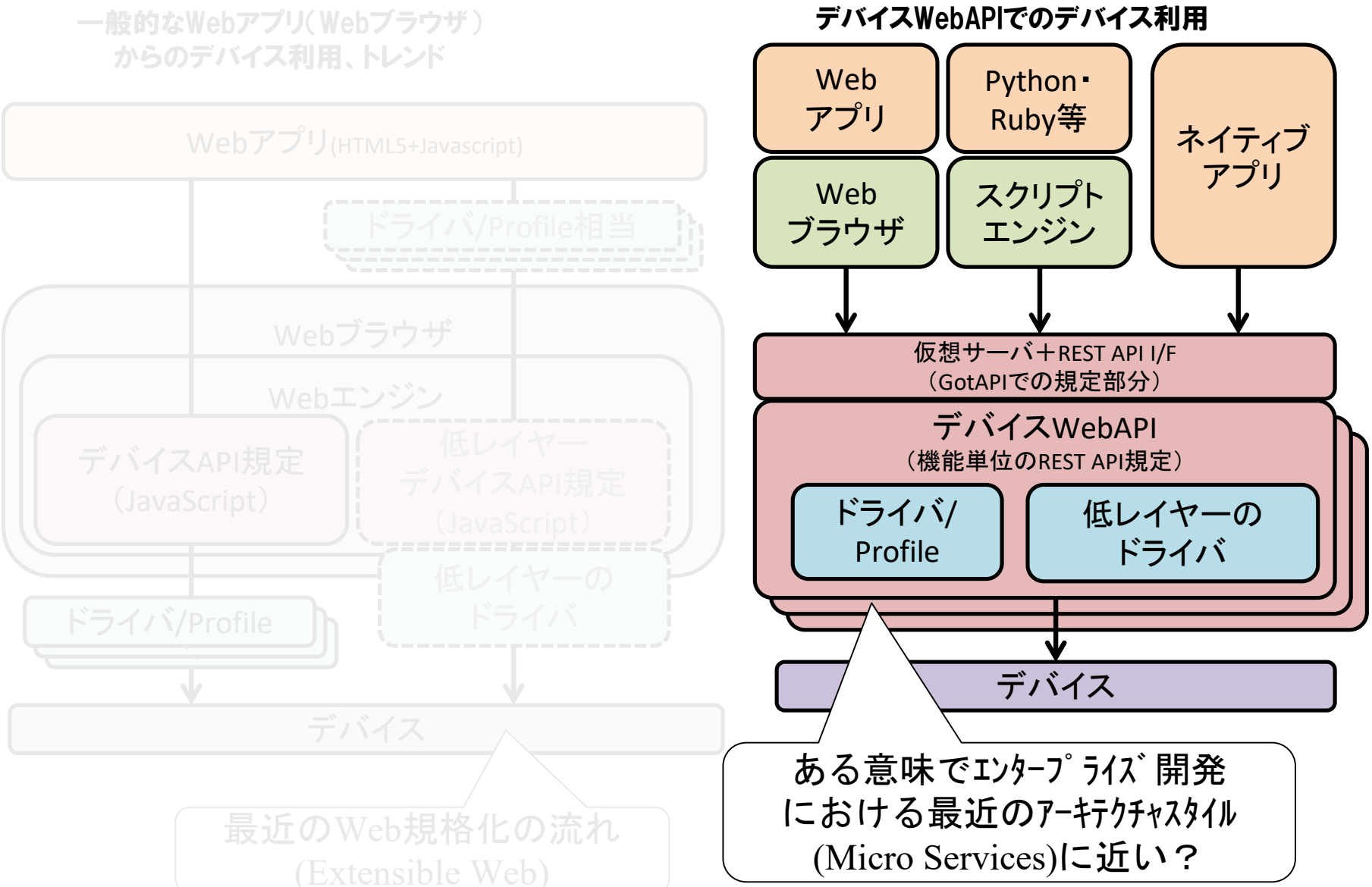


ブラウザでのデバイスAPIの現状



Webブラウザも限定的にデバイス機能アクセスの範囲が拡大中だが前述の課題は残ったまま

Webのトレンドから見てデバイスＷｅｂＡＰＩはどうなったか？



ＷｅｂＡＰＩのトレンド

・サーバレスアーキテクチャ
自前サーバではなく、AWS(Amazon Web Serivce)等のクラウド基盤を活用したサービス設計

・マイクロサービス
一枚板なサービス基盤を作るのではなく、OSといった実行環境や開発環境の違う小さなサービスを組み合わせて一つのサービスを提供する設計。サービス間の環境の違いを吸収してつなげるために、WebAPIが用いられる。

・ＷｅｂＡＰＩ設計の国際標準化
AWSのAPI Gateway等で使われているAPI記述仕様のSwaggerをベースとした標準化が、ﾏｲｸﾛｿﾌﾄ, Google等が立ち上げたOpen API Initiativeで進行中。

WebAPIのトレンド(API記述の標準化)を
デバイスWebAPIでも活用

デバイスWebAPIの現状

・デバイスWebAPIでのSwagger仕様の活用
・ローカル環境からクラウド環境への拡張
・ドキュメント整備

Swaggerとは

・RESTful APIのドキュメントや、サーバ、クライアントコード、エディタ、またそれらを扱うための仕様（YAML/JSON）などを提供するフレームワーク

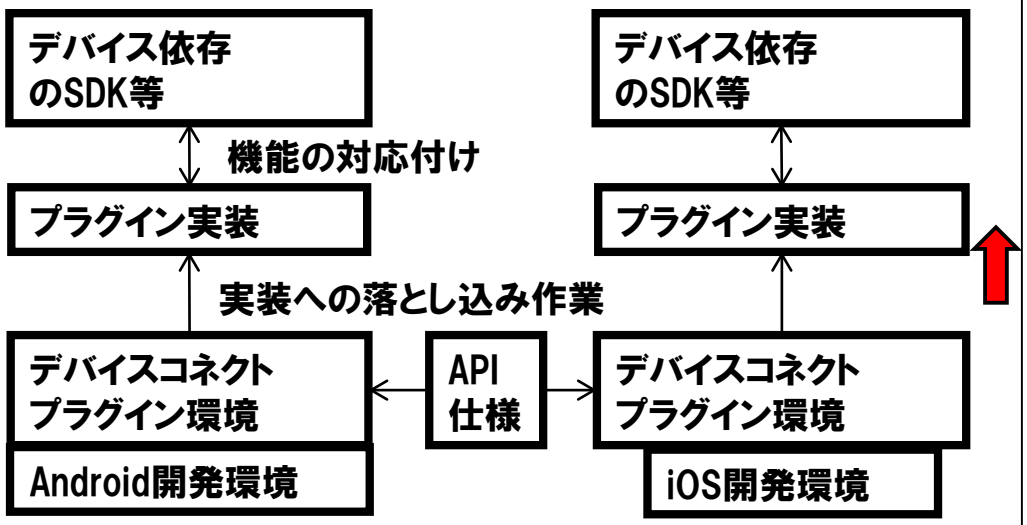
・Google、IBM、Microsoft等がOpen API InitiativeというSwagger仕様に基づくWebAPI標準化団体を設立
（仕様の名前もSwaggerからOAS：Open API Specificationに）

・Amazonは上記団体に入っていないが、AWS API Gatewayでも利用されている

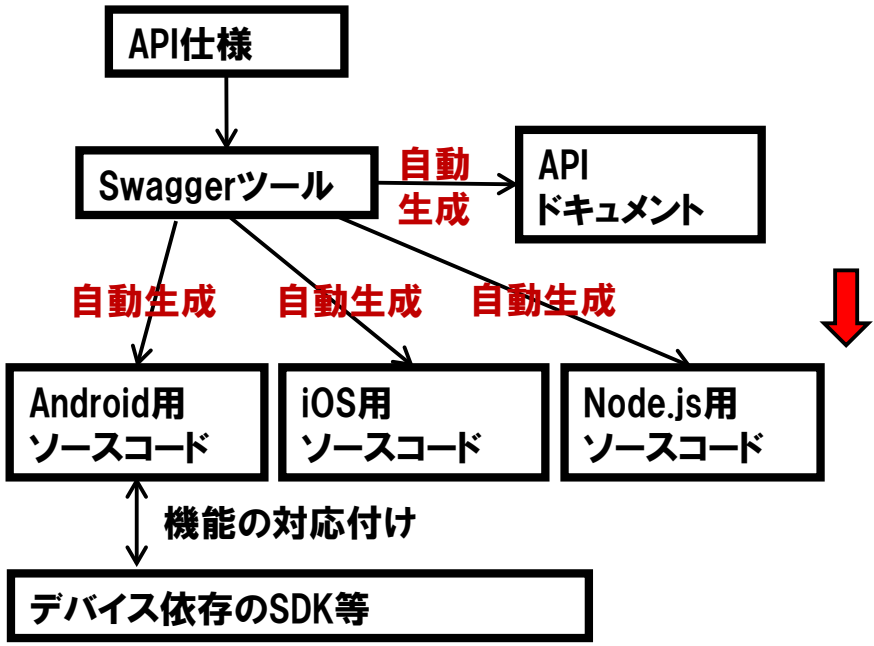
Swaggerベースでのモデルファースト開発

デバイスWebAPIでもSwaggerベースでのモデルファーストのフレームワーク提供
⇒同一のAPI設計で、様々な環境での機能提供を実現

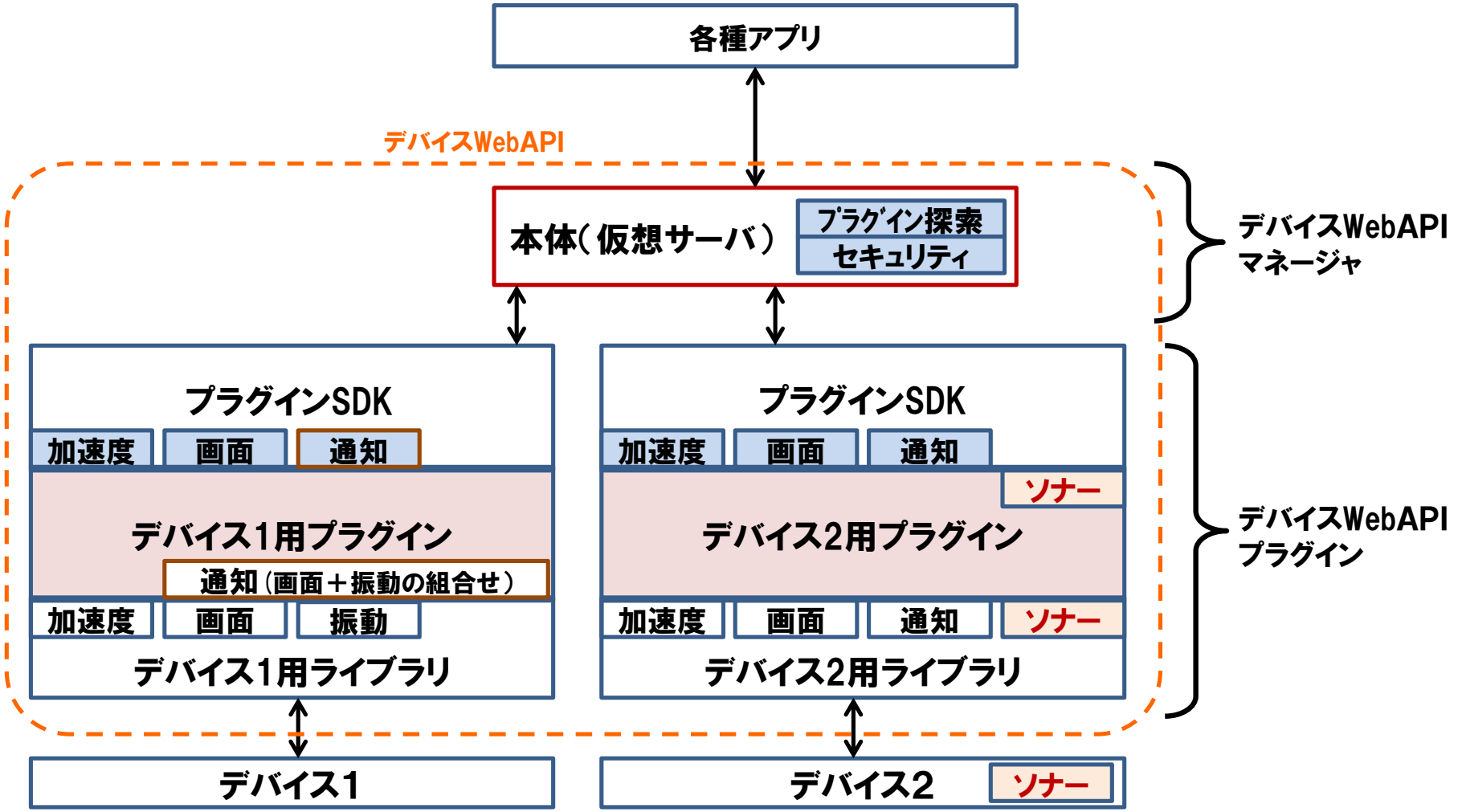
以前
一般的なアプリ開発のやり方



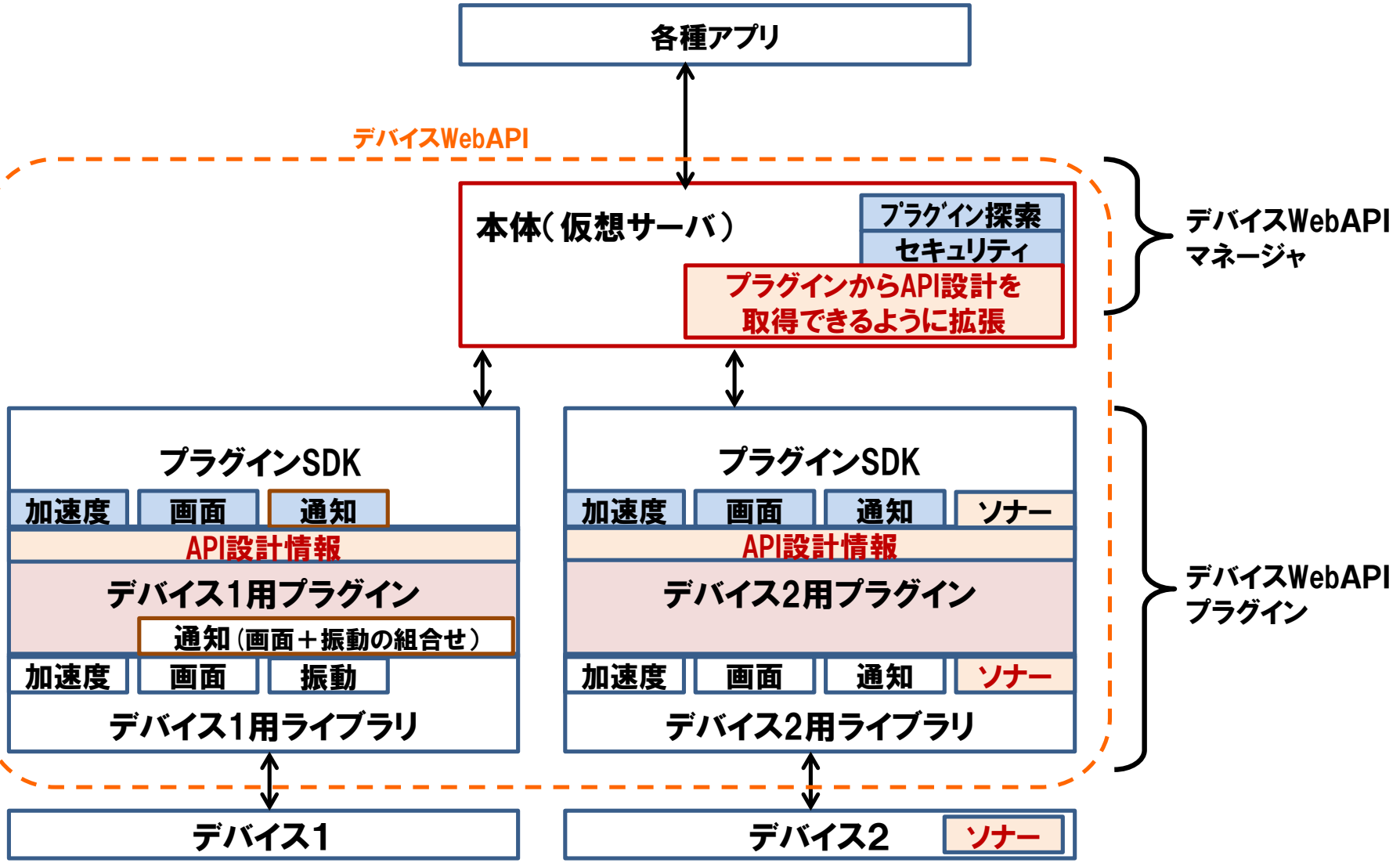
現在
API仕様から各環境のコードを生成



以前のデバイスＷｅｂＡＰＩの構造



現在のデバイスＷｅｂＡＰＩの構造



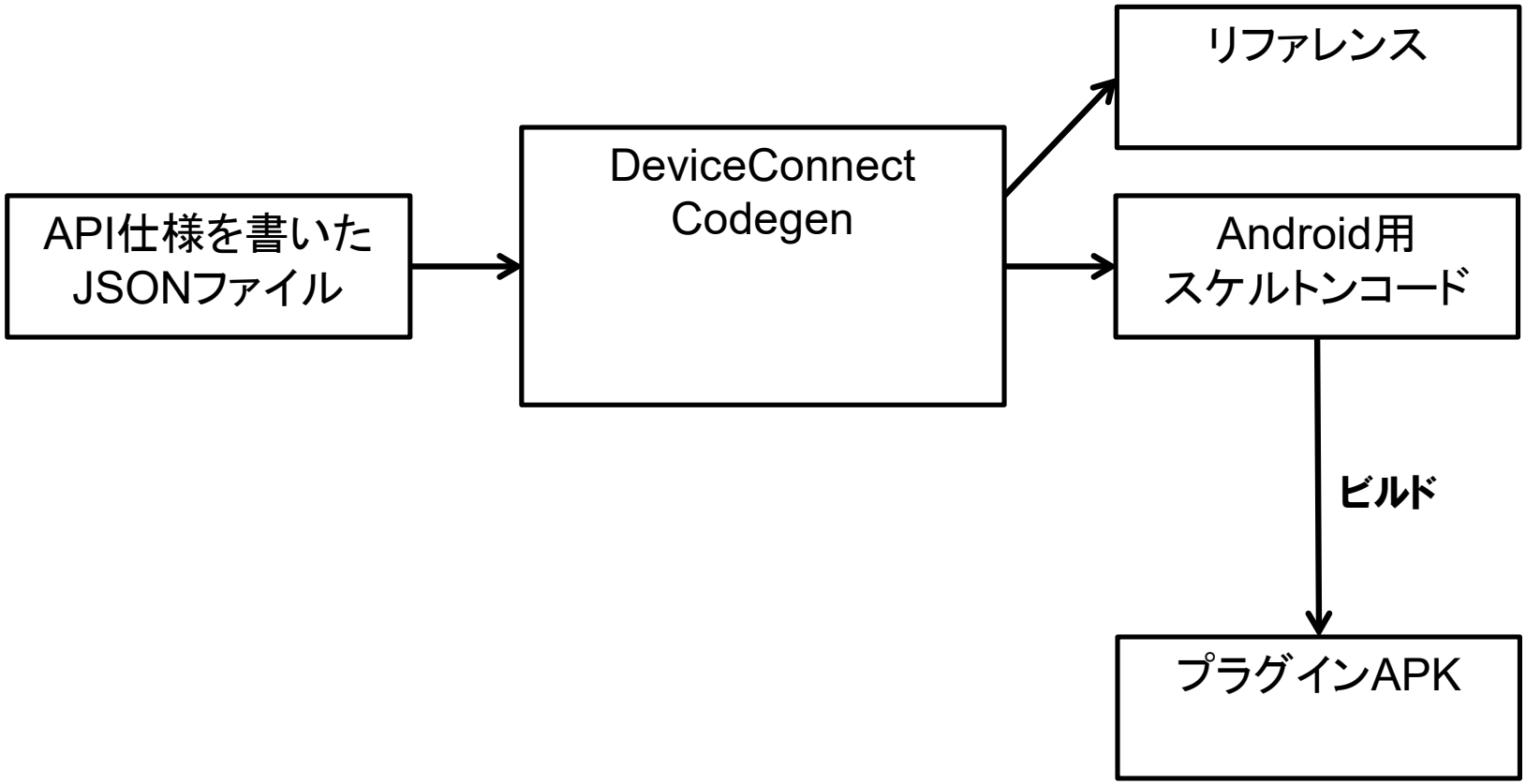
Swaggerでのスケルトンコード生成

・DeviceConnect-Specレポジトリに API仕様を
Swagger 2.0形式で取りまとめて公開

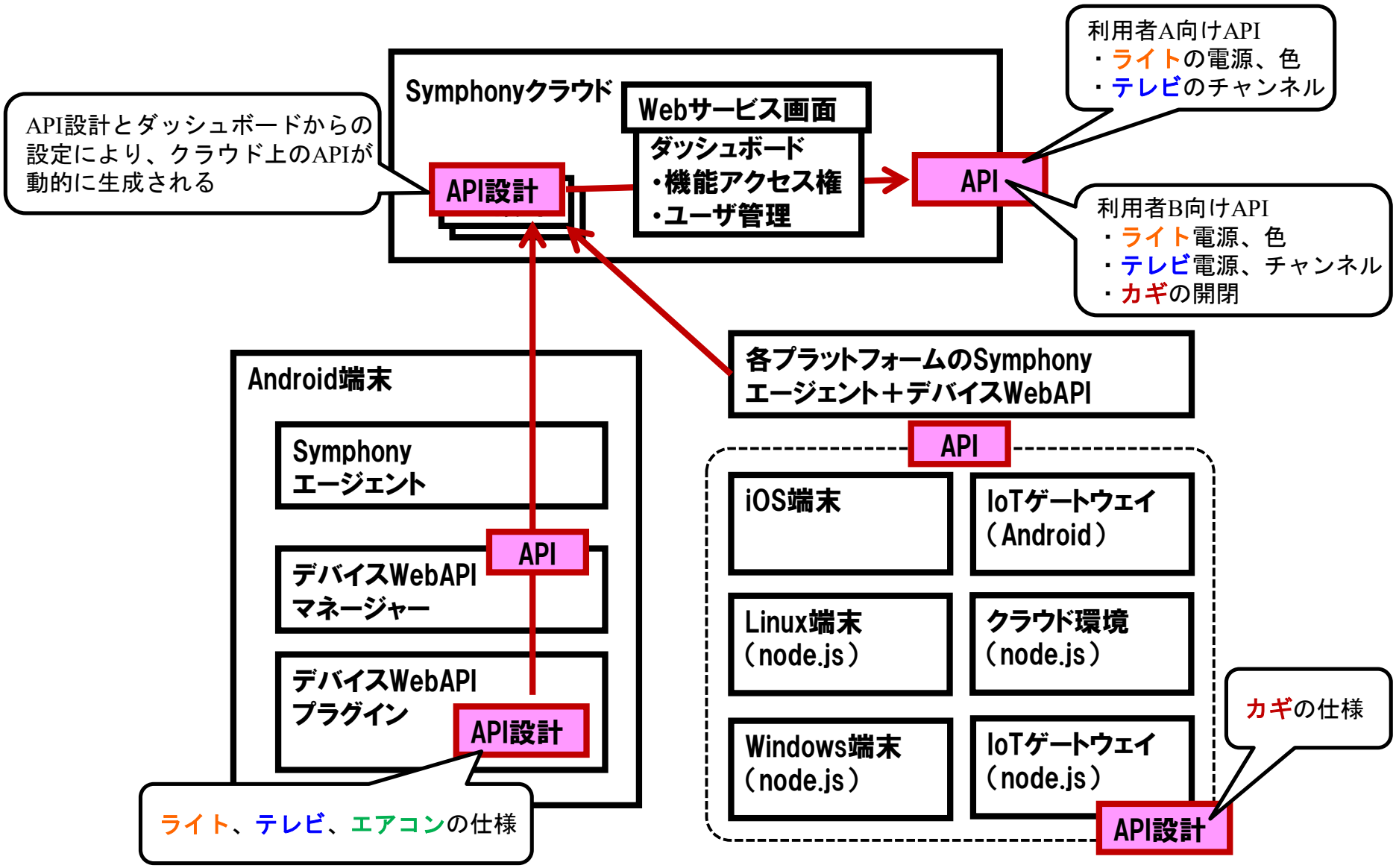
https://github.com/DeviceConnect/DeviceConnect-Spec

こちらのAPI記述を元にして、
API設計の議論をしていきたい

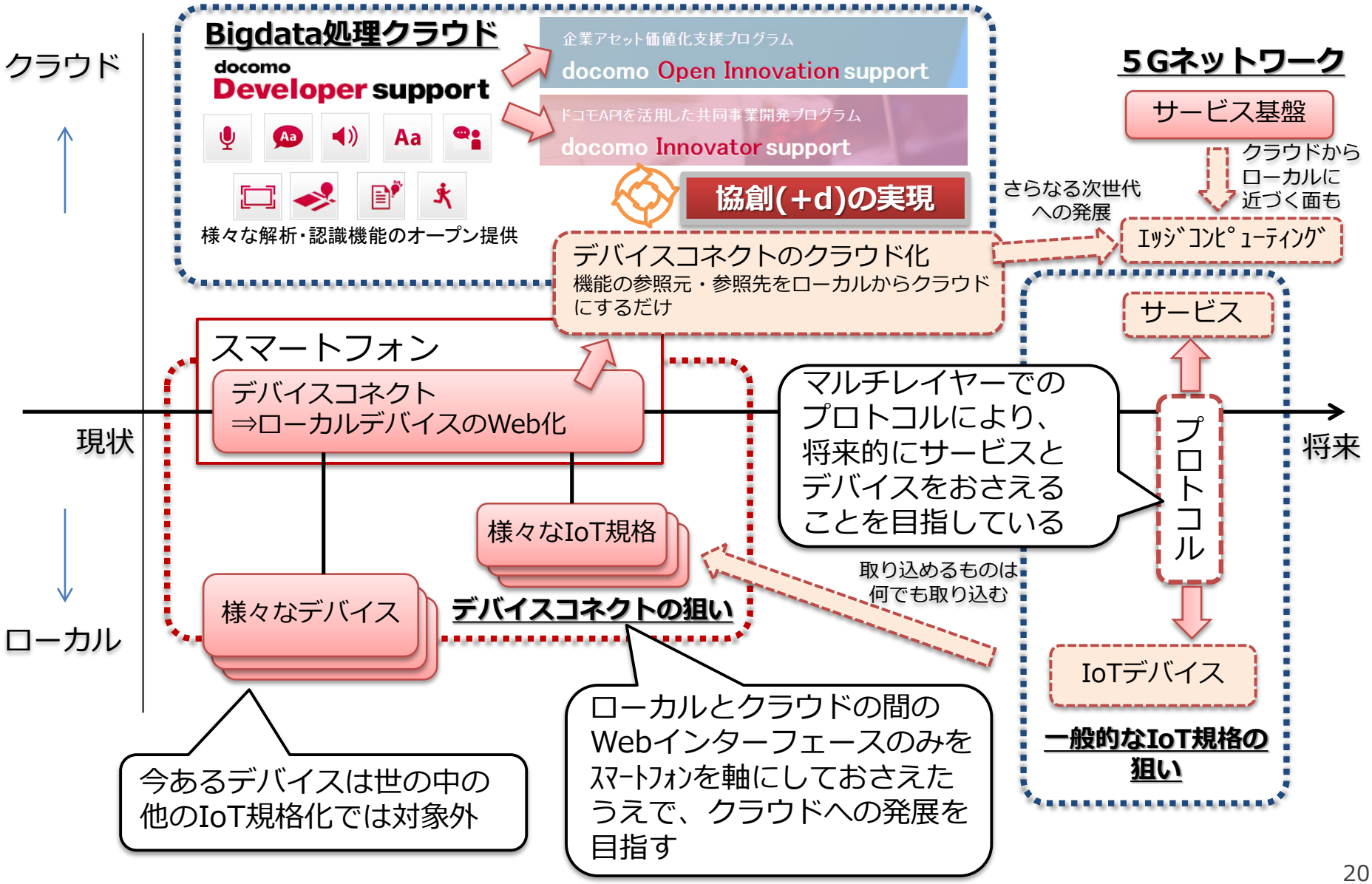
Swaggerでのスケルトンコード生成



Symphonyの構造



DeviceConnectの目指すところ



ドキュメント整備

・デザインパターン：用途・目的に応じたAPIの設計指針

・API作成ガイドライン：API記述の共通ルール

デバイスＷｅｂＡＰＩのデザインパターン

・One shot data ： HTTP GET/POSTでの単純アクセス
　例：アクセスした瞬間の加速度センサーの値を取得（繰り返し値が欲しい場合はポーリング）

・Event driven data ： PUT/DELETE、WebSocketでのイベント処理
　例：加速度センサーの値に変化があった瞬間の値を連続的に自動取得
　※ただし、あくまでイベントとしての処理であり、大容量データは対象としない

・Streaming data ： URIの直接参照
　例：大容量の加速度センサーログの取得、低遅延・高サンプリングレートでのリアルタイムの加速度センサー値の取得等

シンプルで手軽なHTTPアクセスと、効率的なWebSocketによるイベント処理を両立

API作成ガイドライン

・APIの階層構造　/gotapi/profile[/interface] [/attribute]

・命名規則

例: PUT /gotapi/mediaPlayer/play　※lowerCamelCase

誤: PUT /gotapi/meida_payer/play　※snake_case
PUT /gotapi/meida-payer/play
PUT /gotapi/MeidaPayer/Play　※UpperCamelCase
PUT /gotapi/meidapayer/play

※ただし、大文字・小文字の違いについては互換性のために内部的には許容している

・レスポンス定義、エラーコード

等、API記述の共通ルールを整備

https://github.com/DeviceConnect/DeviceConnect-Docs/wiki/Specification-Api-Guidelines

意見交換の場について

https://github.com/DeviceConnectUsers/Community/

GitHubのDeviceConnectはドコモで立ち上げたプロジェクト

DeviceConnectUsersはドコモ管理外のGithub.ioでのコミュニティサイト
（立ち上げ：MOONGIFTさん）