



**G's ACADEMY**  
**TOKYO**

# WebAPI/Ajax



# Web API 沢山あります

私達の作ったアプリとWeb上に公開されているWebAPIをRequest/Responseでデータのやり取りをし、必要なデータ・情報を利用することが可能

# ぐるなび API



新規アカウント発行

[> 利用規約](#)  
[> 設定変更](#)

[ぐるなび Web サービス](#) [はじめての方へ](#) [API 仕様](#) [API テストツール](#) [API 利用ルール](#) [FAQ](#)

## 新しいアイデア、発信しませんか？

ぐるなびWebサービスは、ぐるなびが所有する豊富な飲食店情報を提供する、無料のAPIです。  
ぐるなびWebサービスは、新たなコンテンツを制作する開発者をサポートします。

はじめての方へ



# リアルタイム データベース API

chatのようなリアルタイム通信やMessage通知が可能

Firebase 特長 顧客 料金 ドキュメント サポート

検索

コンソールへ移動

ログイン



Firebase はモバイルアプリの改善とビジネスの成長に貢献します。

スタートガイド

動画をみる

アプリをすばやく作成、インフラストラクチャの管理

Google のインフラが支える、多数の人気アプリが信頼

連携する全プロダクトを1つのコンソールで管理

# VideoChat API

chatから音声電話、ビデオ電話を簡単に使用可能



ホーム 開発者 料金 パートナー サポート

新規登録 ログイン English



## リアルタイムコミュニケーションで イノベーションを起こそう

アプリやWebサイト、IoTデバイスにビデオ通話とP2P通信を追加できるAPI



# Microsoft AI API

## Cognitive Services

自然なコミュニケーション手段を通して、見たり、聞いたり、話したり、理解したり、ユーザーのニーズを解釈したりできるインテリジェントなアルゴリズムを、アプリや Web サイト、ボットに取り入れましょう。AI で貴社のビジネスを今すぐ変革しましょう。

Cognitive Services を無料で試す >

Cognitive Services についてさらに詳しく: [ディレクトリ](#) [価格](#) [ドキュメント](#)

## AI を使用してビジネスの問題を解決しましょう



### 視覚

イメージ処理アルゴリズムで、写真をスマートに識別したり、キャプションを追加したり、モデレートしたりできます。



### 音声

音声をテキストに変換したり、確認に音声を使用したり、アプリに話者の認識機能を追加したりできます。



### 知識

インテリジェントなお勧め機能やセマンティック検索といったタスクを解決するために、複雑な情報やデータをマッピングします。



### Search

Bing Search APIs をアプリに追加して、1 つの API 呼び出しで 何十億という Web ページやイメージ、ビデオ、ニュースをくまなく調べる機能を実装しましょう。



### 言語

アプリが、事前に構築済みのスクリプトで自然言語を処理したり、センチメントを評価したり、ユーザーが望んでいることの認識方法を学習したりできます。

# 画像認識 API

画像から背景、人、場所など様々な情報を取得

Home

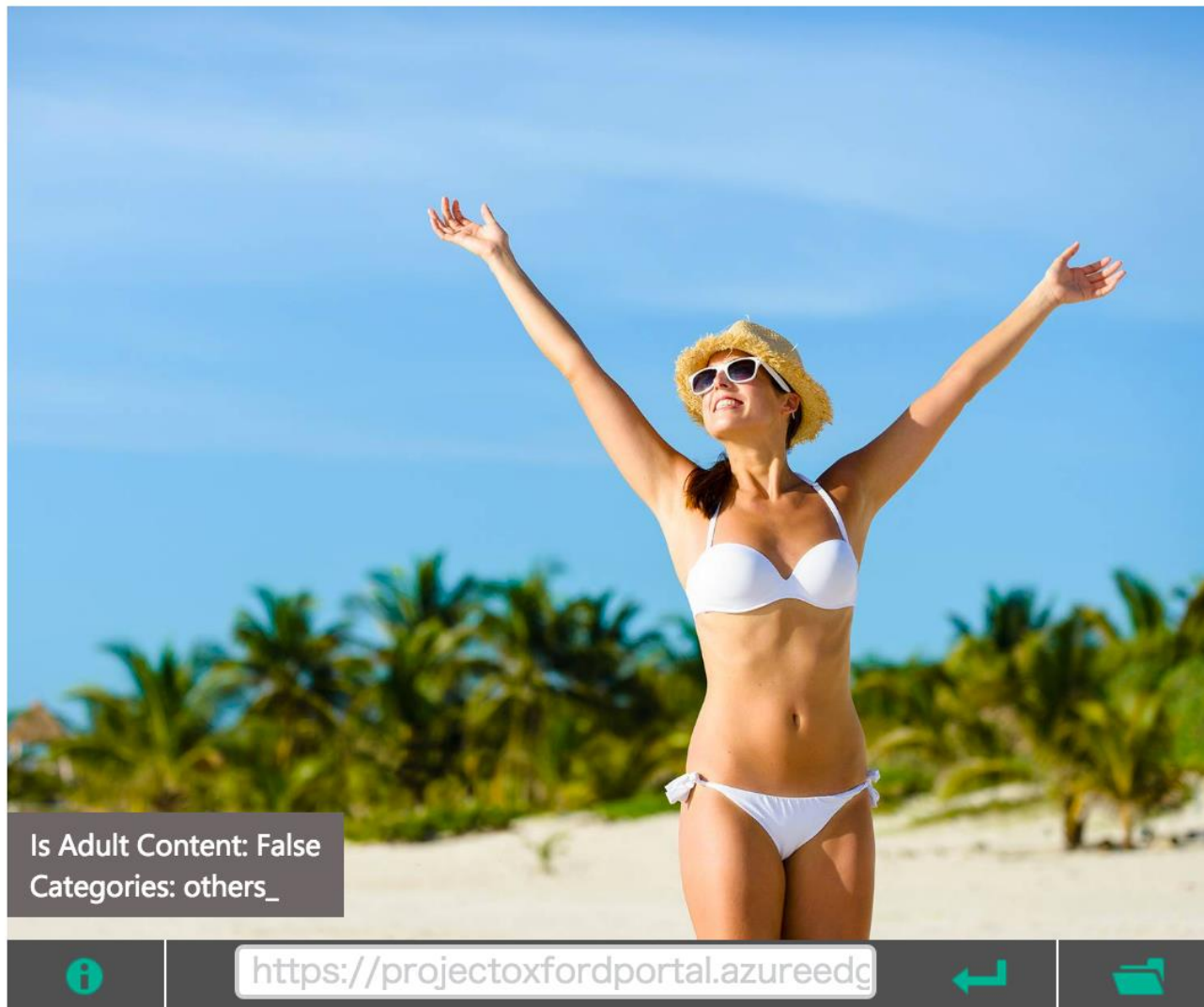
APIs ▼

Applications

Developers ▼

Pricing

My account



## Features:

Feature Name	Value
Description	{ "type": 0, "captions": [ { "text": "a beautiful woman standing on a beach", "confidence": 0.679803189466983 } ] }
Tags	[ { "name": "sky", "confidence": 0.999806821346283 }, { "name": "outdoor", "confidence": 0.9973123073577881 }, { "name": "woman", "confidence": 0.9627310633659363 }, { "name": "person", "confidence": 0.9388522505760193 }, { "name": "beach", "confidence": 0.9086331725120544 }, { "name": "swimsuit", "confidence": 0.5581952929496765 }, { "name": "beautiful", "confidence": 0.5157498717308044 } ]
Image Format	jpg
Image Dimensions	1500 x 1155
Clip Art Type	0 Non-clipart
Line Drawing Type	0 Non-LineDrawing
Black & White Image	Unknown



# 感情認識 API

画像から顔の場所を判定し、感情を数値化する

Home

APIs ▼

Applications

Developers ▼

Pricing

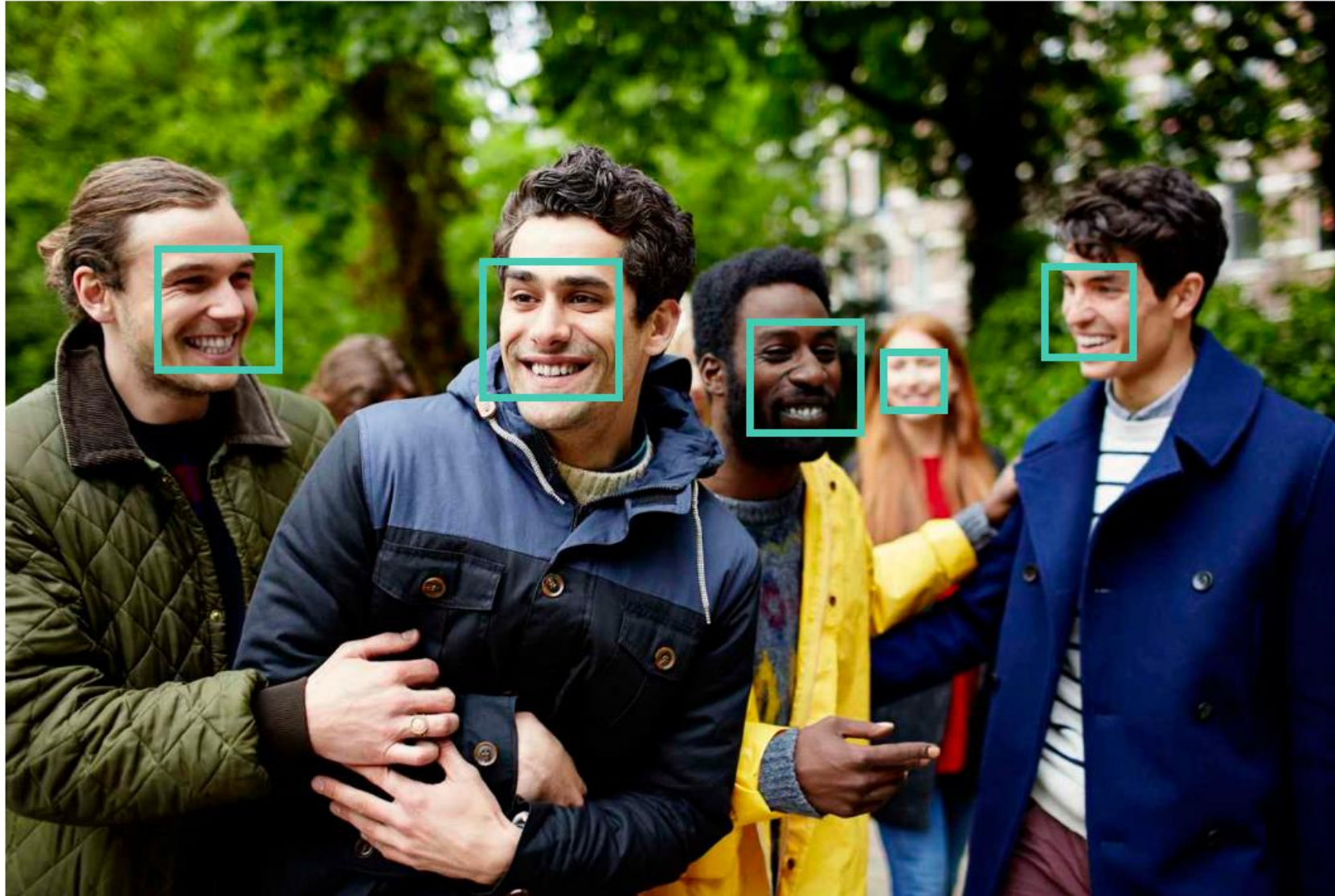


Image URL



Detection Result:

5 faces detected

JSON:

```
[
  {
    "faceRectangle": {
      "left": 488,
      "top": 263,
      "width": 148,
      "height": 148
    },
    "scores": {
      "anger": 9.075572e-13,
      "contempt": 7.048959e-9,
      "disgust": 1.02152783e-11,
      "fear": 1.778957e-14,
      "happiness": 0.9999999,
      "neutral": 1.31694478e-7,
      "sadness": 6.04054263e-12,
      "surprise": 3.92249462e-11
    }
  },
  {
    "faceRectangle": {
      "left": 238,
      "top": 421,
      "width": 148,
      "height": 148
    },
    "scores": {
      "anger": 9.075572e-13,
      "contempt": 7.048959e-9,
      "disgust": 1.02152783e-11,
      "fear": 1.778957e-14,
      "happiness": 0.9999999,
      "neutral": 1.31694478e-7,
      "sadness": 6.04054263e-12,
      "surprise": 3.92249462e-11
    }
  },
  {
    "faceRectangle": {
      "left": 358,
      "top": 458,
      "width": 148,
      "height": 148
    },
    "scores": {
      "anger": 9.075572e-13,
      "contempt": 7.048959e-9,
      "disgust": 1.02152783e-11,
      "fear": 1.778957e-14,
      "happiness": 0.9999999,
      "neutral": 1.31694478e-7,
      "sadness": 6.04054263e-12,
      "surprise": 3.92249462e-11
    }
  },
  {
    "faceRectangle": {
      "left": 421,
      "top": 478,
      "width": 148,
      "height": 148
    },
    "scores": {
      "anger": 9.075572e-13,
      "contempt": 7.048959e-9,
      "disgust": 1.02152783e-11,
      "fear": 1.778957e-14,
      "happiness": 0.9999999,
      "neutral": 1.31694478e-7,
      "sadness": 6.04054263e-12,
      "surprise": 3.92249462e-11
    }
  },
  {
    "faceRectangle": {
      "left": 493,
      "top": 421,
      "width": 148,
      "height": 148
    },
    "scores": {
      "anger": 9.075572e-13,
      "contempt": 7.048959e-9,
      "disgust": 1.02152783e-11,
      "fear": 1.778957e-14,
      "happiness": 0.9999999,
      "neutral": 1.31694478e-7,
      "sadness": 6.04054263e-12,
      "surprise": 3.92249462e-11
    }
  }
]
```



# 顔認識 API

画像から顔、目、鼻、口など複数のポイントあらゆる情報を取得可能

Home APIs ▼ Applications Developers ▼ Pricing

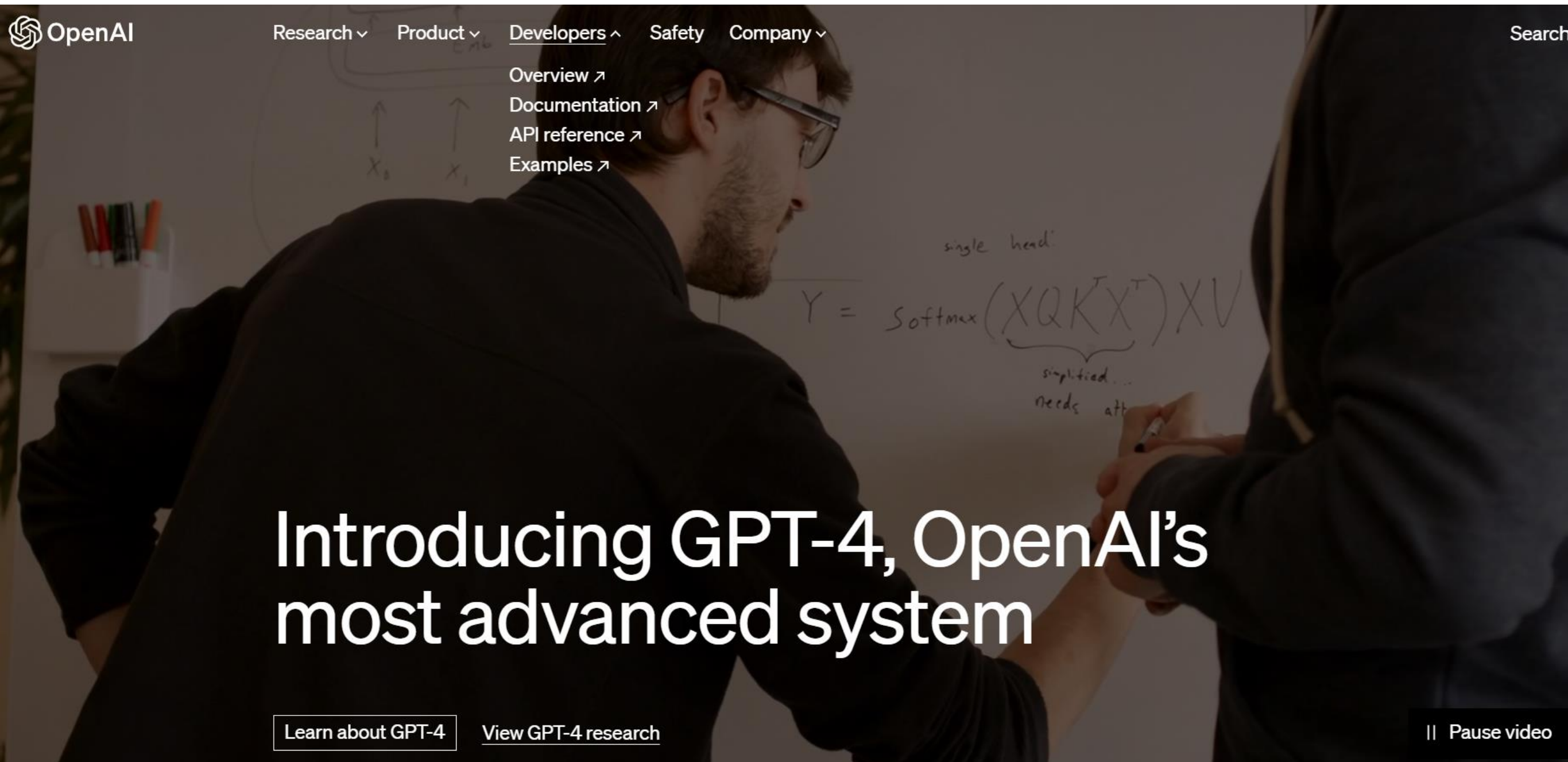


```
Detection Result:
JSON:
[
  {
    "faceId": "80ec7366-da66-4413-b463-5631e73b813d",
    "faceRectangle": {
      "width": 68,
      "height": 68,
      "left": 134,
      "top": 47
    },
    "faceLandmarks": {
      "pupilLeft": {
        "x": 148.5,
        "y": 73
      },
      "pupilRight": {
        "x": 178.7,
        "y": 62.1
      },
      "noseTip": {
        "x": 169.2,
        "y": 83
      }
    }
  }
]
```



# OPEN AI API

自然な会話で検索エンジンのように情報を収集したりデータ創作したりできる



[API Reference - OpenAI API](#)



# APIを使用するメリット

◇自分たちで全てを作らない。

◇外部サービスを利用することで素早く開発。

プロトタイプを作成するには必須の知識です。

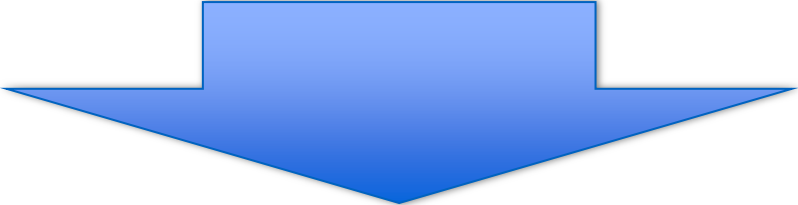
# Ajax



# Ajaxとは

- AjaxのAである「Asynchronous（非同期）」は、非同期でのクライアント・サーバ間の通信を指します
- JavaScriptによりクライアント上での動作が主で必要なデータのみ受信することで通信量の負担を軽減
- 特殊なサーバの設定等は必要としない

# Ajaxのメリット

- Webページのリンクをクリックした時のレスポンス待ち時間の体感時間が少ない。
- 必要な部分の情報のみを取得変更し、必要なときに更新可能のため高速に動作する。
- 例)  
5画面 → 5HTMLファイル (通常の方法)  
  
5画面 → 1HTMLファイル (Ajaxだとかうできる！)



# Ajaxのデメリット

- SEO対策には不向き
- スクリプトの知識が必要
- 作りが複雑になりがち
- 更新履歴が残りません、ブラウザの[戻る]ボタンでは1つ前の状態には戻りません。
- 更新後でも再表示すると初期状態に戻ってしまいます。

# Ajaxを使う準備

◇ CDN : HTML内の「 axios 」を使う前に記述しましょう！！

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

## 【重要】

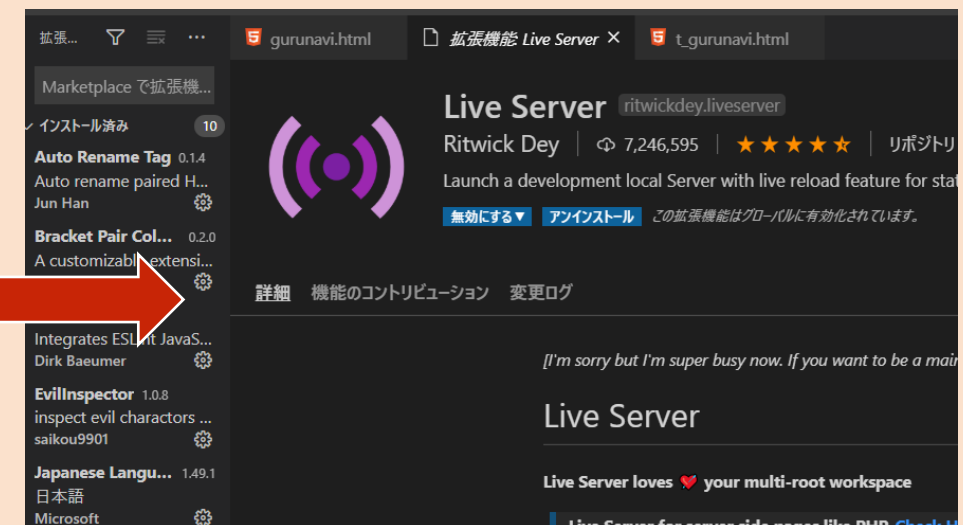
Ajaxは「 HTTP 」通信で動作します！！

開いてるブラウザの**URLの最初が「 http://... 」 or 「 https://... 」**

どちらかで始まっている必要があります！！

※URLの最初が「 file:///.... 」はNGです！！

拡張機能「Live Server」があればOK！！



# axios基本1[get]

ajax.html

構文は以下がベースになります

```
axios.get('ajax.php').then(function ( response ) {  
    console.log( response.data ); //通信OK  
}).catch(function (error) {  
    console.log(error); //通信Error  
}).then(function () {  
    console.log("Last");//通信OK/Error後に処理を必ずさせたい場合  
});
```



# axios基本2[get]

ajax\_get.html

## GETでデータをサーバー側に送信するケース

```
//送信データを用意
```

```
const data = {  
  id:'value1', mode:"value2", type:"value3"  
};
```

```
//Ajax（非同期通信）
```

```
axios.get('ajax_get.php' , {  
  params: data  
}).then(function (response) {  
  console.log(response.data); //通信OK  
  document.querySelector("#status").innerHTML="ajax_get.php/通信OK";  
}).catch(function (error) {  
  console.log(error);          //通信Error  
}).then(function () {  
  console.log("Last");         //通信OK/Error後に処理を必ずさせたい場合  
});
```

# GET送信 されてる?? <確認する方法>

ajax\_get.php/  
通信OK

The screenshot shows the Chrome DevTools Network tab. The address bar displays `localhost/php03/ajax/ajax_get.html`. The Network tab is selected, and the filter is set to 'XHR'. A single request is listed with the name `ajax_get.php?id=value1&mode=value2&type=value3`. The request is highlighted, and the 'Headers' sub-tab is active. The headers section shows the following information:

- Sec-Fetch-Mode:** cors
- Sec-Fetch-Site:** same-origin
- User-Agent:** Mozilla/5.0 (Windows NT 10.0; bKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.37.36)

Below the headers, the 'Query String Parameters' section is expanded, showing the following parameters:

- id:** value1
- mode:** value2
- type:** value3

Red dashed boxes and arrows highlight the 'Network' tab, the request name, the 'Headers' sub-tab, and the 'Query String Parameters' section.

# axios.get

Ajax通信でJSON文字列を簡単に取得  
google\_test1.html



# 演習 : GoogleBooks (実際に取得)

サンプル  
json01.html

GoogleB

console.dirで表示  
[右クリック → 検  
証 → consoleで確  
認]

Elements Console Sources

<top frame> ☐ Preserve

start

▼ Object *i*

▼ items: Array[10]

- ▶ 0: Object
- ▶ 1: Object
- ▶ 2: Object
- ▶ 3: Object
- ▶ 4: Object
- ▶ 5: Object
- ▶ 6: Object
- ▶ 7: Object
- ▶ 8: Object
- ▶ 9: Object

length: 10

▶ \_\_proto\_\_: Array[0]

kind: "books#volumes"

totalItems: 614

▶ \_\_proto\_\_: Object

itemsに全て入っている

オブジェクトの  
中身を参照！

# 演習 : GoogleBooks (実際に取得)

サンプル  
json01.html

start

▼ Object *i*

- ▼ items: Array[10]
  - ▼ 0: Object
    - ▶ accessInfo: Object
      - etag: "l0UBaATMQ84"
      - id: "nhjV0RFfmogC"
      - kind: "books#volume"
    - ▶ saleInfo: Object
    - ▶ searchInfo: Object
    - selfLink: "https://www.googleapis.com/books/v1/volumes/nhjV0RFfmogC"
    - ▶ volumeInfo: Object
    - ▶ \_\_proto\_\_: Object
  - ▶ 1: Object
  - ▶ 2: Object
  - ▶ 3: Object

volumeInfo.title  
本のタイトルが記載されてる。  
※他にどんな情報があるか確認！

# 演習

## google\_test2.html

わからないとき！手を動かしてやってみる！



## ◇ 他データも表示（3つ：書籍名、著者、解説）

```
axios.get('https://www.googleapis.com/books/v1/volumes?q=jquery')
.then(function (res) {
  const val = res.data;
  console.log(val); //確認 : v.items[0].volumeInfo.title
```

//レコードの数だけ繰り返す

```
let html="";
```

```
for(let i=0; i<val.items.length; i++){
  const elm = val.items[i].volumeInfo;
```

```
  console.log(elm.authors);
```

```
  console.log(elm.description);
```

```
  html += '<tr>';
```

```
  html += '<td>書籍名 : ' + elm.title + '</td>';
```

```
  if( !typeChk(elm.authors, "undefined") ){
```

```
    html += '<td>著者 : ' + elm.authors[0] + '</td>';
```

```
  }
```

```
  if( !typeChk(elm.description, "undefined") ){
```

```
    html += '<td>解説 : ' + elm.description + '</td>';
```

```
  }
```

```
  html += '</tr>';
```

```
};
```

```
$("#book_list").append(html);
```

```
});
```

① console.log でデータ構造を確認

② TRタグ内 TDタグを追加。  
「書籍名、著者、解説」を表示！

# 課題

わからないとき！手を動かしてやってみる！

課題：以下内容を来週授業までにやっておきましょう！

## GoogleBooks検索

jquery

データ読み込み

② 検索結果を表示

① 入力して検索

題名：jQuery逆引きマニュアル	出版社株式会社インプレスジャパン	<a href="#">リンク</a>
題名：jQueryによるWebサービス 活用ワザ実践サンプル集	出版社秀和システム	<a href="#">リンク</a>
題名：実践!Ajaxフレームワーク jQuery	出版社マイナビ出版	<a href="#">リンク</a>

ヒント："?q=jquery"を変数で変更できるように！



# OpenAI

openai.html

[API Reference - OpenAI API](#)

# JSON

JSON文字列

# JSONとは

オブジェクト変  
数

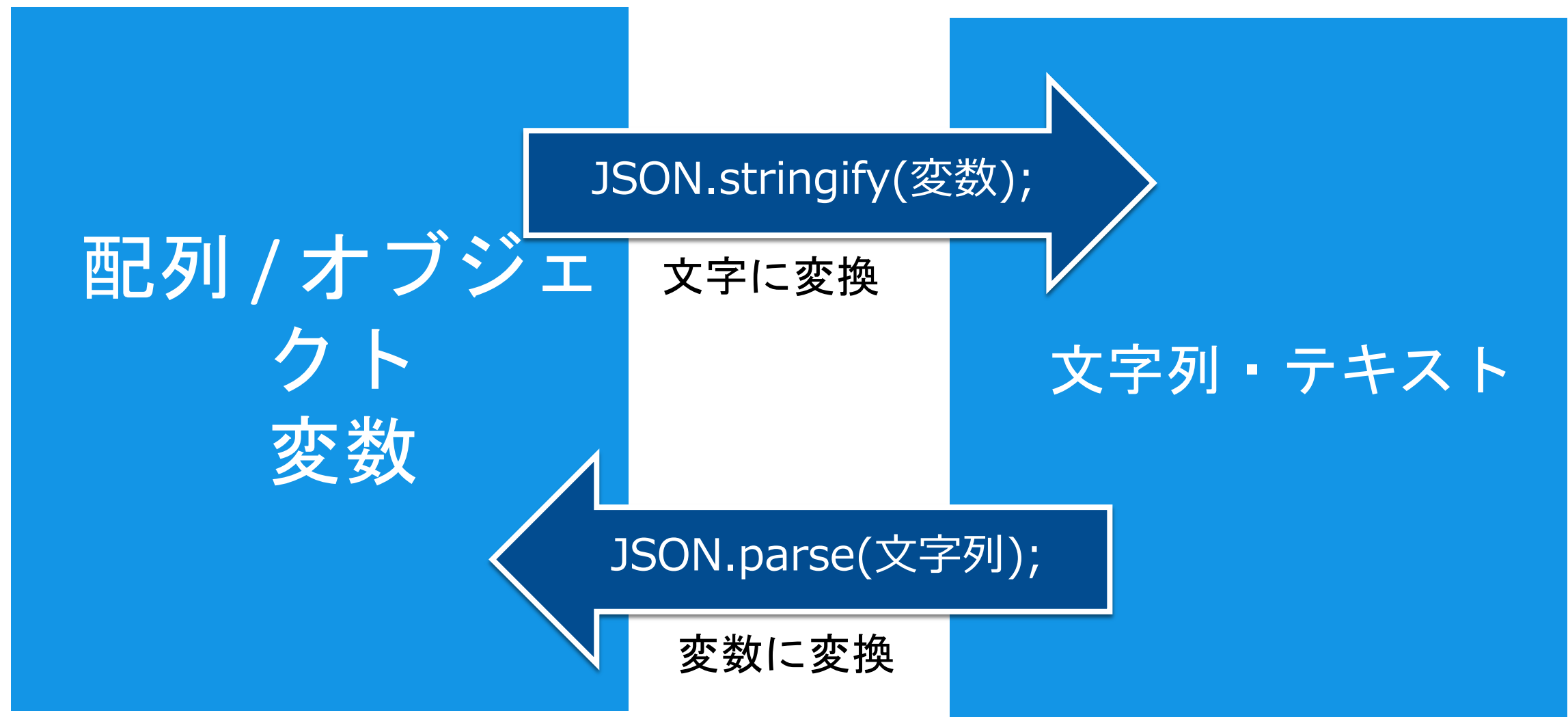
バーチャル値



JSON文字列

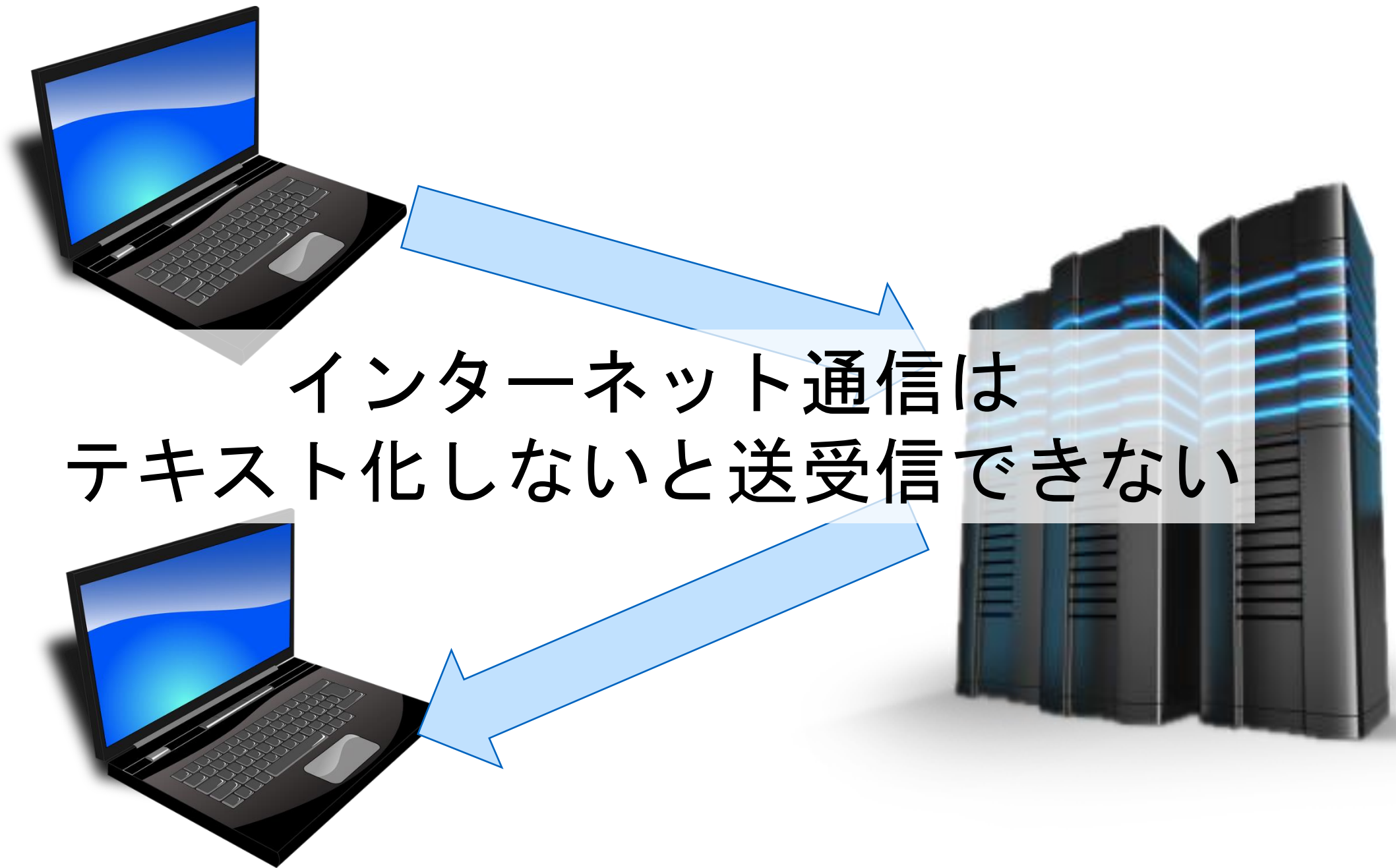
リアル値

# JSONとは





# JSON文字列になぜ変換するのか？



# JSON とは何か？ どう使うのか？

～配列/オブジェクトは生データ→文字変換～

## パソコン:ブラウザ上

①変数

```
let data =
```

```
[  
  {"fname": "Tarou", "lname": "Yamazaki"},  
  {"fname": "Anna", "lname": "Smith"},  
  {"fname": "Peter", "lname": "Jones"}  
]
```

②JSON形式の文字列に変換

```
let json_text = JSON.stringify(data);
```

④データ保存



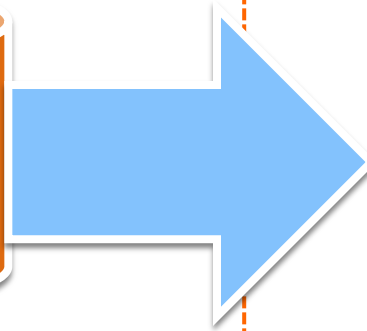
③json\_text (JSON形式の文字列)

```
[  
  {"fname": "Tarou", "lname": "Yamazaki"},  
  {"fname": "Anna", "lname": "Smith"},  
  {"fname": "Peter", "lname": "Jones"}  
]
```

※変数では他言語にデータを渡せない→JSON形式の“文字列”に変換することで可能になる！

## 【 JSONデータ（文字列からオブジェクトに変換）】

### ⑤データ取得



```
let json_text = [  
  {"fname": "Tarou", "lname": "Yamazaki"},  
  {"fname": "Anna", "lname": "Smith"},  
  {"fname": "Peter", "lname": "Jones"}  
]
```

JSON形式の文字列

### 【 JSON文字列 → 配列/オブジェクト 変数に変換】

```
let data = JSON.parse(json_text); //JSONからオブジェクトに変換  
console.log(data);
```

### 【 オブジェクトへのデータ参照方法 】

data

data[0].fname

Yamazaki

data[1].fname

Anna

data[2].fname

Peter