


# フォームと Ajax



# フォーム HTMLのおさらい

# フォーム HTML おさらい



フォーム専用のタグ

form, input, checkbox, submit など

```
<input type="" name="" value="">
```

nameとvalueは連想配列(keyとvalue)の関係

[https://developer.mozilla.org/ja/docs/Learn/Forms/The\\_native\\_form\\_widgets](https://developer.mozilla.org/ja/docs/Learn/Forms/The_native_form_widgets)





# Vue.jsでのフォーム

# v-model と v-bind & v-on

	v-model	V-bind(:) と v-on(@)
特徴	シンプルにつくれる 修飾子がつけれる (number, lazy, trim)	複雑な内容も設定できる
書き方	v-model="test"	:value = test @input="test = \$event.target.value"
応用	computedと組み合わせる事も (get/set)	\$event.target.value以外を扱ったり

v-model	v-bind(:)	v-on(@)
input, textarea	value	Input
checkbox, radio	checked	change
select	value	change

# 双方向データバインディング

---

## Two-way Data Binding

`<input :value="test"`  
          `@input="test = $event.target.value">`

data内 test: “

The diagram illustrates two-way data binding. A green arrow points from the text 'data内 test: “' up to the `:value="test"` attribute of the `<input>` tag. An orange arrow points from the `@input="test = $event.target.value">` attribute down to the same text 'data内 test: “'.

# フォームのイベント/オプション

タイミング	送信ボタンクリック時	リアルタイム フォーカス外れた時	リアルタイム (即時)
イベント	@click, @submit .prevent	@change, @blur v-model.lazy	@input v-model (inputタグ)
オプション API	methods	computed (get/set) ○ watch △	computed (get/set) ○ watch △


# computed (get/set)

---

```
computed:{  
  yourName:{ //監視したいプロパティ名  
    get(){ return }, // 必ずreturnが必要  
    set(value){ // 引数に新しい値が入る  
    }  
  }  
}
```




# フォームバリデーションの補足



設定多数・漏れ防止のため  
ライブラリを使う事も多い

VeeValidate など ->別のセクションで



# サンプル4 ToDoリスト (フィルターつき)

# 配列の検索(filterとindexOf)

---

array.filterで配列に1つずつ条件をつける  
(今回は文字列があるかどうか)

```
filteredList(){  
  let that = this  
  return this.todos.filter( todo =>{  
    return todo.item.indexOf(that.query) !== 1 })  
}
```



An abstract background on the left side of the slide, composed of numerous overlapping, semi-transparent triangles in various shades of green and yellow, creating a mosaic-like effect.

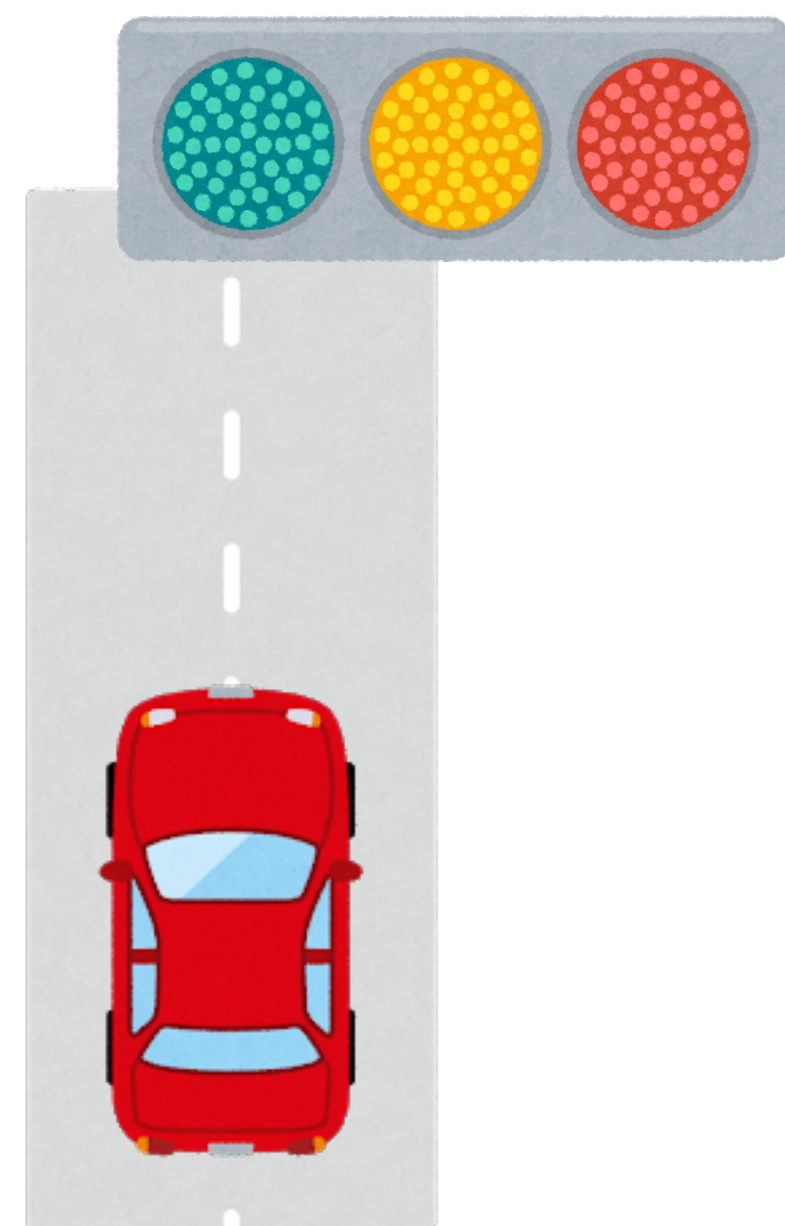
# Web通信のおさらい



# Web通信 ・ http

フォームには必須のhttp通信

Hyper Text Transfer Protocol (ルール)



クライアント



サーバー



Laravel



# HTTPリクエストとレスポンス

## HTTPリクエスト

- HTTPリクエスト行 (メソッド)
- HTTPヘッダー
- データ本体



## HTTPレスポンス

- レスポンス状態行 (状態コード)
- HTTPヘッダー
- データ本体

# httpメソッド

---

GET ・ ・ URLに表示される。(検索条件など) → クエリーストリング

POST ・ ・ 見られてはNGなデータはこっち

[https://qiita.com/7968/items/  
4bf4d6f28284146c288f](https://qiita.com/7968/items/4bf4d6f28284146c288f)

# CrossOriginResourcePolicy

<https://aaa.com:80>



サードパーティ  
API

CORS

Access-Control-Allow-Origin

クロスオリジン



同一オリジン



サーバー



An abstract background on the left side of the slide, composed of numerous overlapping, semi-transparent geometric shapes (polygons) in various shades of green and yellow, creating a textured, mosaic-like effect.

# 簡易サーバーと Network



# 簡易サーバーとNetwork

---

VSCode ・ ・ Live Server

Web Server for Chrome

Webpack (環境構築必要)

XAMPP/MAMP (Apache / nginx) ・ ・ PHPなど

通信状況はChromeのNetworkで確認

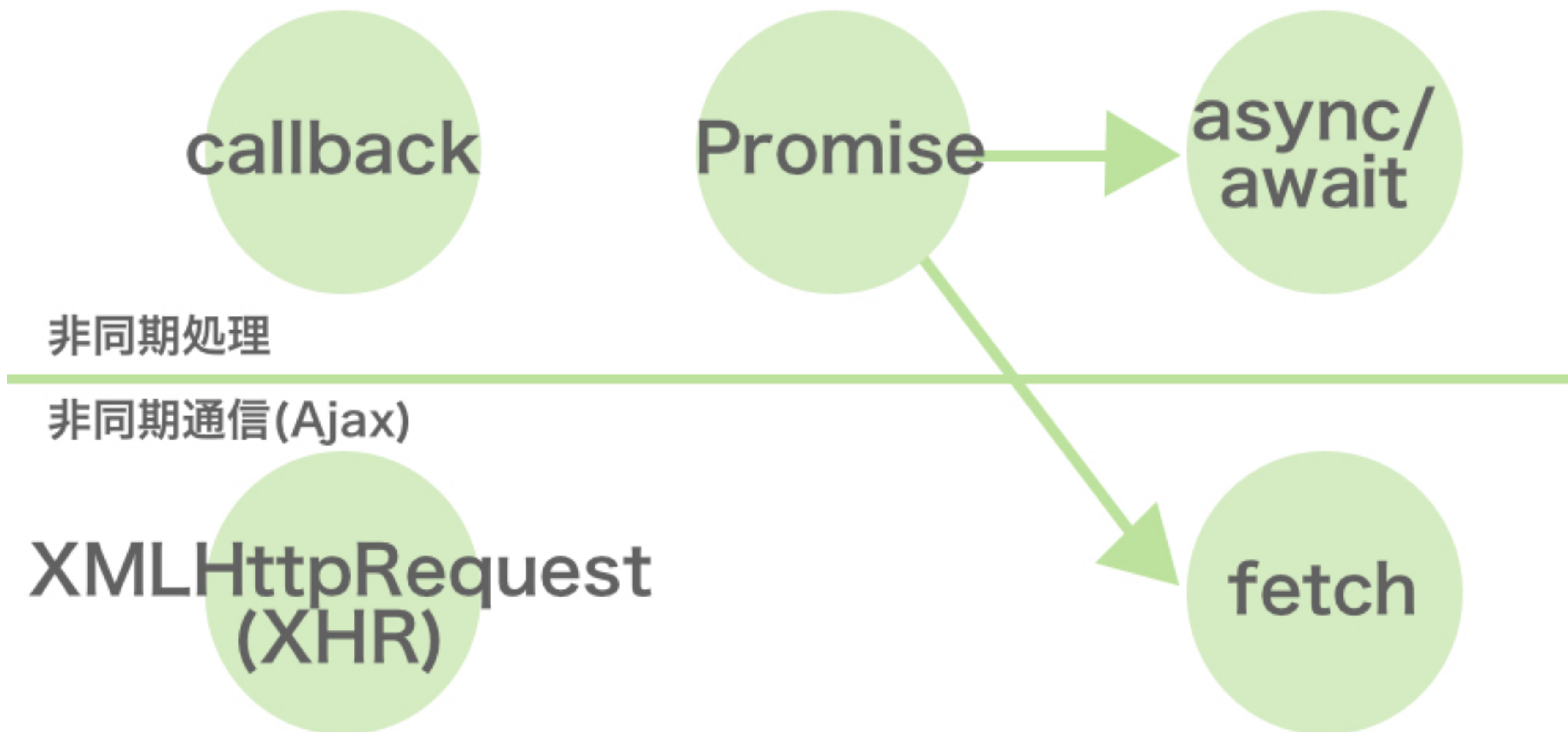


An abstract background on the left side of the slide, composed of numerous overlapping, semi-transparent triangles in various shades of green and yellow, creating a mosaic-like effect.

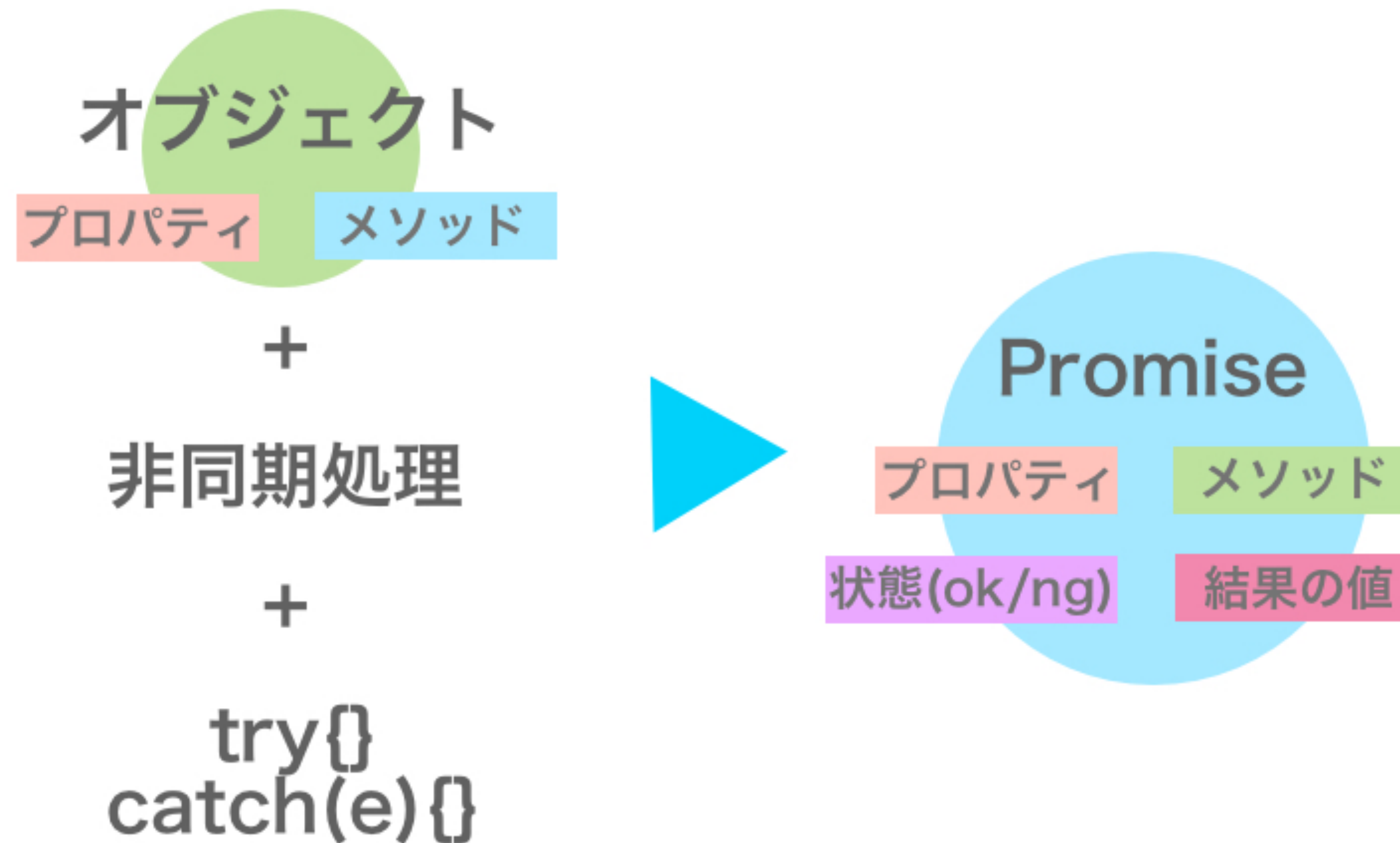
Promise/  
async/await/fetch



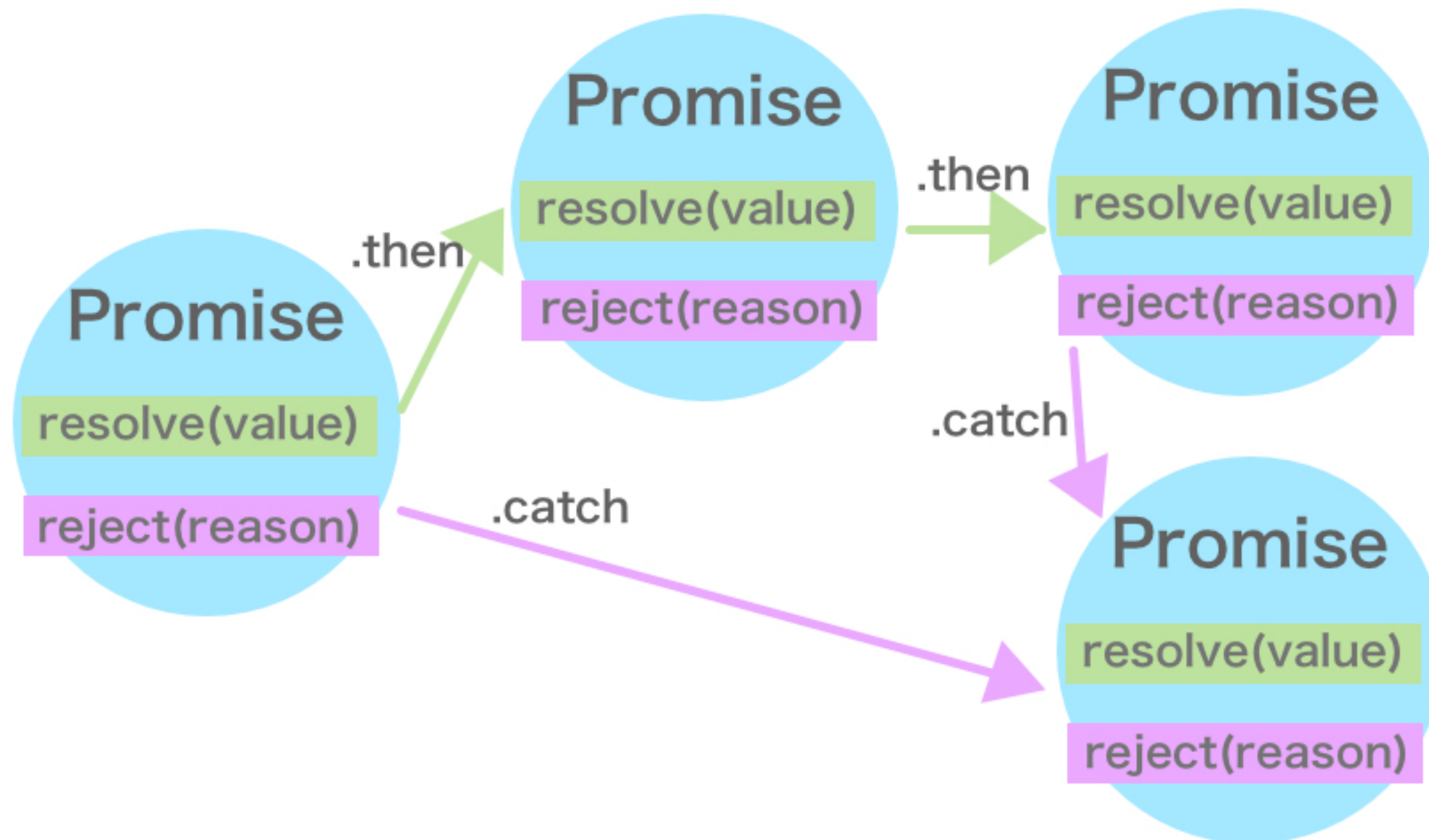
# 非同期処理と非同期通信



# Promiseの正体



# Promiseチェーン



# async / await

Promiseをわかりやすく書き換えたもの  
元は C# (2012) 様々な言語に派生

```
async function name() {
```

-> returnはPromise (.thenなども使える)

await 関数名() はasyncの中でのみ使える



# fetch API

---

fetch(url, options)

返り値はPromise (Responseオブジェクト)

Response.jsonで自動的にパースされる

よく使うoptions (Request)

method, mode, credentials, body

[https://developer.mozilla.org/ja/docs/Web/  
API/Fetch\\_API/Using\\_Fetch](https://developer.mozilla.org/ja/docs/Web/API/Fetch_API/Using_Fetch)



# Vue.jsでAjax

# 非同期通信(Ajax) 簡易表

タイミング	画面表示時	クリック時	リアルタイム
イベント (ライフサイクルフック)	created mounted (\$nextTick)	@click, @submit	@input
オプション API	methods	methods	watch

※サーバーと通信が発生するため  
ある程度(1秒～3秒)間隔を開けて実行する  
lodash.js の `_.debounce` / `_.throttle`

# async/await/fetch (シンプル版)

---

```
methods:{  
  async getDogImage(){  
    const response = await fetch(url,  
options)  
    .then( response => response.json() )  
    this.dogImage = response.message  
  }  
}
```

# Lodash



JavaScriptの便利ライブラリ

`_.debounce(fn, 1000)` //間隔を空けて実行  
`_.throttle(fn, 1000)` //実行後 指定msは実行しない

<https://lodash.com/>



# watch/created

---

```
watch:{  
  watchTest(){  
    this.watchDogImage() } }
```

```
created(){  
  this.watchDogImage = _.debounce(fn, 1000)  
}
```

Vueインスタンスにプロパティとして事前追加