

JavaScriptファイル ↓
外部スクリプトファイル
のURL

```

<script
src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js">
</script>
<!-- <script src="vue.js"></script> ローカルで vue.js を用いる場合 -->

```

Vueを読み込み、使えるようにしてる。

HTML

```

<div id="app">
  <p>
    {{message}}
  </p>
</div>

```

id属性について
・1ページ内に同じ名前複数回X
・「# id名」を参照できる(外から)

マスタッシュ構文 {{...}}
バインディング

el: element?

elに要素に要素内でVueが有効になる

Vueインスタンス生成

```

<script>
new Vue({
  el: '#app', // # で id と関係づける
  data: {
    message: 'HelloWorld! This is the first Vue.js app.'
  }
})
</script>

```

idが「app」のHTML要素内でVueが有効になる。

<div id="app">
タグ内部でVueが有効になる。

「モデル」と言う。data部分(モデル)に変更が加えられると、htmlにリアルタイムで反映される。

HTML・CSSを操る役割

JSコードをHTMLから呼び出し方法

その①

直接

HTMLにJSコードを書く

- ・bodyタグの最後に書く
- ・HTMLファイルが長くなりメンテナンス性が低い

その②

外部ファイルとして読み込む

- ・scriptのsrc属性にファイルの場所とファイル名を書く。

```
src="js/sample.js"></script>
```

ファイル名 ファイル名

```
JS
<script
  Vue.jsを外部から読み込む
  src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js">
</script>

<!-- <script src="vue.js"></script> ローカルで vue.js を用いる場合 -->

HTML
<div id="app">
  <p>{{message}}</p>
  <!-- {{x}}でVueのxを受け取る -->
  <button v-on:click="reverseMessage">メッセージを反転</button>
</div>

JS(Vue)
<script>
new Vue({
  idが"app"の要素にVueを適用させる。
  el: '#app', // # で id と関係づける
  data: {
    message: 'HelloWorld! This is the first Vue.js app.'
  },
  methods: {
    reverseMessage: function() {
      this.message = this.message.split('').reverse().join('')
    }
  }
})
</script>
```

イベントハンドラ
イベント(ユーザーによる一連の挙動)に
対応して発動させる関数
V-onディレクティブを使用し
イベントハンドラと、どのイベントで
発動するかを指定する。

V-onディレクティブの書き方
ver①
V-on: イベント名 = 'ハンドラ名' (内数)
ver②
① イベント名 = 'ハンドラ名'

V-on
ディレクティブ

クリックされたとき
関数と発火

イベントハンドラ
... イベントにより発動する処理

②はプログラムの
変数にアクセスできる

、(空白)で
文字列と区切る

西2列の要素を
逆転させる

西2列の要素を
逆転させる

イベントの種類★

click	クリック
change	変更
input	入力開始
select	フォーム送信
dblclick	ダブルクリック

-- まだまだある。

< 01_基礎・テンプレート構文

003_テンプレート構文.html

Vue.js/01_基礎・テンプレート構文



```
<!doctype html>

<html lang="ja">

<head>

  <title>インデックス</title>

  <meta charset="utf-8">

  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script> <!-- Vue.js を読み込む -->

  <!-- <script src="vue.js"></script> ローカルで vue.js を用いる場合 -->

</head>

<body>
  <div id="app">

    <h1>Webアプリ研究会へ</h1>

    <p>{{ message }}</p>          <!-- {{x}}でVueのxを受け取る -->

    <p>{{ number + 5 }}</p>      <!-- JavaScript を直接 {{}} 内に書くことができる -->

    <p>{{ ok ? 'YES' : 'NO' }}</p> <!-- 論理式にも対応している -->

    <p>{{ sayHi() }}</p>        <!-- {{}} 内には関数も入れれる -->

  </div>

  <script>
    new Vue({
      el: '#app',    // # で id と関係づける
      data: {
        message: 'きょうは初めてのWeb会議です',
        number: 3,
        ok: true
      },
      methods: {
        sayHi: function() {
          return 'Hi';
        }
      }
    })
  </script>
</body>
```

id='app'の手続きで
Vueに通知する

javascriptの関数の書き方

×ソート
内数) 変数

check

```
<!doctype html>
<html lang="ja">

<head>
  <title>インデックス</title>
  <meta charset="utf-8">
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script> <!-- Vue.js を読み込む -->
  <!-- <script src="vue.js"></script> ローカルで vue.js を用いる場合 -->
</head>

<body>
  <div id="app">
    <p>{{ sayHi() }}</p> <!-- {{}} 内には関数も入れれる -->
  </div>
  <script>
    new Vue({
      el: '#app',    // # で id と関係づける
      data: {
        message: 'きょうは初めてのWeb会議です'
      },
      methods: {
        sayHi: function() {
          return this.message; // data ないの message を引用するには、this.message とする
        }
      }
    })
  </script>
</body>
</html>
```

同じインスタンス内の変数を使うときは
「this.変数名」とする

```

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

<!-- <script src="vue.js"></script> ローカルで vue.js を用いる場合 -->

<div id="app">
  <!-- <p>{{message}}</p> -->
  <p v-text="message"></p> <!-- 上の行と全く同じ機能 -->
</div>

<script>
new Vue({
  el: '#app', // # で id と関係づける
  data: {
    message: 'HelloWorld! This is the first Vue.js app.'
  }
})
</script>

```

js(vue.js)のデータがhtmlにリアルタイム反映

バインディングは2つの方法がある!

① Mustache 構文

html: <p>{{ message }}</p>
 vue: data: {
 message: " ~ "

他にもモデルバインディングはcss

V-text
 V-html
 V-show
 V-class
 V-on
 V-style

② ディレクティブで属性にする

html: <p v-text = "message"></p>
 vue: data: {
 message: " ~ "

```
<!doctype html>

<html lang="ja">

<head>

  <title>インデックス</title>

  <meta charset="utf-8">

  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script> <!-- Vue.js を読み込む -->

  <!-- <script src="vue.js"></script> ローカルで vue.js を用いる場合 -->

</head>

<body>
  <div id="app">
    <h1>Webアプリ研究会へ</h1>
    <p>{{ message }}</p>
    <p>{{ sayHi() }}</p>
  </div>
  <script>
    new Vue({
      el: '#app', // # で id と関係づける
      data: {
        message: 'きょうは初めてのWeb会議です',
        number: 3,
        ok: true
      },
      methods: {
        sayHi: function() {
          this.message = "来週二回目のWeb会議をします"
          return 'Hi';
        }
      }
    })
  </script>
</body>
```

html

vue

<!-- {{x}}でVueのxを受け取る -->

<!-- {{}} 内には関数も入れれる -->


```
<!doctype html>
<html lang="ja">

<head>

  <title>インデックス</title>

  <meta charset="utf-8">

  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script> <!-- Vue.js を読み込む -->

  <!-- <script src="vue.js"></script> ローカルで vue.js を用いる場合 -->
```

```
</head>
```

```
<body>
  <div id="app">
```

```
    <h1>Webアプリ研究会へ</h1>
```

```
    <p v-once>{{ message }}</p> <!-- v-once を入れると一度だけしか実行されない -->
```

```
    <p>{{ sayHi() }}</p> <!-- v-once があるために、この関数の1行目は実行されない -->
```

```
</div>
```

```
<script>
```

```
  new Vue({
    el: '#app', // # で id と関係づける
    data: {
      message: 'きょうは初めてのWeb会議です',
      number: 3,
      ok: true
```

```
    },
```

```
    methods: {
```

```
      sayHi: function() {
```

```
        this.message = "来週二回目のWeb会議をします" // これにより {{message}} が上書きされる
```

```
        return 'Hi';
```

```
      }
```

```
    }
```

```
  })
```

```
</script>
```

```
</body>
```

V-once テンプレート
1度だけ表示される。変数の値が
変わっても、2度と表示されない。

変数messageと「Hi」を表示する関数

上書きされるけど、すでに1度
表示されているので、表示されない
V-once 99%

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

```
<!-- <script src="vue.js"></script> ローカルで vue.js を用いる場合 -->
```

プレテキストとして
読み取られる...

`{{...}}` は中身とプレテキストとして扱う

```
<div id="app">
```

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

```
<!-- html は認識されない --> → '<h1>h1です</h1>'
```

```
<div v-html="html"></div>
```

```
<!-- html として認識されて表示する --> → h1です
```

```
</div>
```

html として
認識される!

V-html テンプレートは、既定で HTML オブジェクトを
html として扱う!

```
js(vue)
```

```
<script>
```

```
new Vue({
```

```
el: '#app', // # で id と関係づける
```

```
data: {
```

```
html: '<h1>h1です</h1>'
```

```
// v-html は「クロスサイトスクリプティング」という脆弱性がある
```

```
// https://cybersecurity-jp.com/security-measures/1842/
```

```
// v-html で html を引用するときには、絶対にユーザーから取得した情報を使ってはいけない
```

```
// 引用する html に悪意のあるコードを入れられる可能性があるため
```

システムの
悪用される不具合

ユーザーの入力値を出力する所には
Mustache 構文を使う!

```
}
```

```
})
```

```
</script>
```



```

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

<!-- <script src="vue.js"></script> ローカルで vue.js を用いる場合 -->

<div id="app">
  <p><a v-bind:href="url1">Yahoo</a></p> <!-- urlを引用する -->
  <p><a :href="url2">Amazon</a></p> <!-- v-bind は省略できる -->
  <!-- v-bind を 使うか省略するか、いずれかに統一した方がよい -->
</div>

```

V-bind デフォタグ
htmlの属性に対する
バインディング

v-bind: 属性名
= "プロパティ名"

: 属性名 = "プロパティ名"

'v-bind' は省略可
「:」だけでもOK.

```

<script>
  new Vue({
    el: '#app', // # で id と関係づける
    data: {
      url1: 'https://yahoo.co.jp',
      url2: 'https://amazon.co.jp'
    }
  })
</script>

```

htmlの「属性」とは?

- ・htmlの要素に対して性質を与える
- ・<要素名 属性名="設定内容">
- ・グローバル属性(すべての要素に使える)
 - class CSSのクラス名
 - id 一意の名称をつける
 - style CSSのクラス名
 - title 要素の概要
- ・非グローバル属性(特定の要素にのみ)
 - href a(ア-)タグの属性の
リンク先を指定する。

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

<!-- <script src="vue.js"></script> ローカルで vue.js を用いる場合 -->

html
<div id="app">
  <p><a v-bind:[attribute]="url">Yahoo</a></p> <!-- [] で attribute を引用 -->
  <p><a v-bind="amazonObject">Amazon</a></p>
  <!-- Chromeで右クリックして検証を選び出力されたHTMLをチェックせよ -->
</div>

js(Vue)
<script>
  new Vue({
    el: '#app',    // # で id と関係づける
    data: {
      url: 'https://yahoo.co.jp',
      attribute: 'href',
      amazonObject: {    // Object の定義
        href: 'https://amazon.co.jp',
        id: 'Web-app_group'
      }
    }
  })
</script>
```