プログラミング演習（JavaScript）

■Web システムと JavaScript

●Web システム

・クライアントとサーバ

Web システムは「クライアント」と「サーバ」が通信することで成り立つ。

Web システムにおけるクライアントとは、Web ブラウザを指し、サーバとはブラウザからの通信を受け HTML/CSS/JavaScript/画像ファイルなどの情報を提供するためのシステムを意味する。

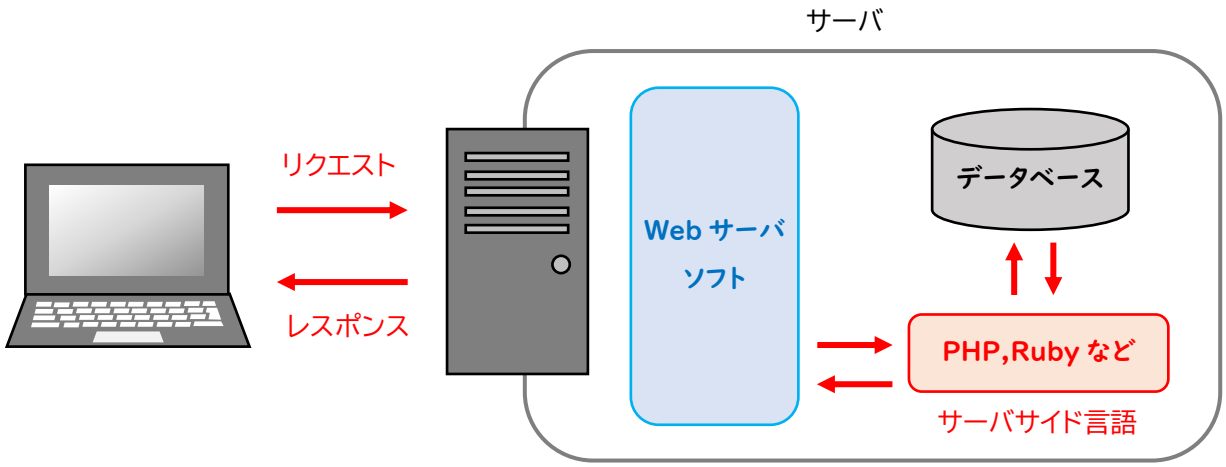


図1－1 Web システム

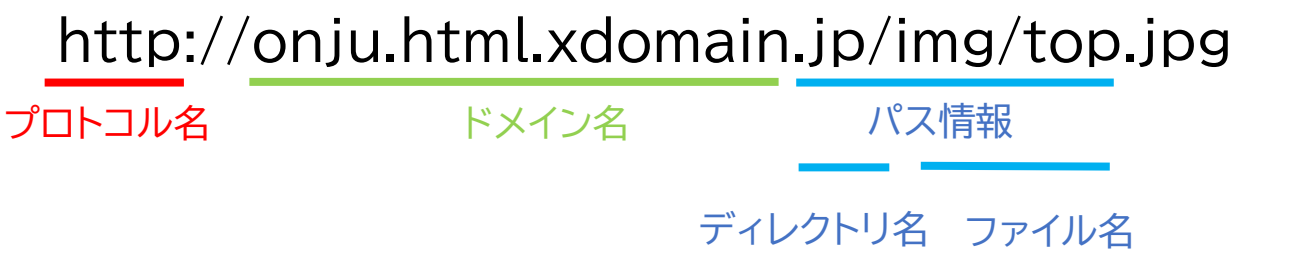
図1－1のように、クライアントはリクエストデータを送信（リクエスト）し、サーバはリクエストを解釈して Web サイトの閲覧や Web サイトの閲覧に必要な、HTML/CSS/JavaScript/画像ファイルと言った情報をクライアントに返す。

・パス情報

サーバの内部には、複数のディレクトリ（フォルダ）/ファイルが存在します。そのため、サーバを利用する際に、複数あるディレクトリのうち、どのディレクトリを公開するかを設定する必要がある。この公開先に指定したディレクトリをルートディレクトリと言う。

そのため、Web 上に公開する情報（HTML/CSS/JavaScript など）は、このルートディレクトリの下に配置することになる。通常、ドメイン名でのアクセスはルートディレクトリでのアクセスを意味する。URL 内のパス情報は、ルートディレクトリのどの階層（フォルダ）のどのファイルにアクセスを行うのかを表す情報になる。

例えば、次の URL で考える。



上にあげた URL のパス情報は、/img/top.jpg と言うことになる。これは、ルートディレクトリ直下に存在する「img」フォルダ内の「top.img」というファイルへのアクセスを意味する。

なお、http://onju.html.xdomain.jp/ とドメイン名の後に何も入力していない場合やフォルダ名を入力した場合は、ルートやフォルダの中の、「index.拡張子」ファイルを探してアクセスする。この index.拡張子ファイルは「ファーストドキュメント」と呼ばれ、ファイル名を省略してもアクセス可能である。http://onju.html.xdomain.jp/と入力してアクセスした場合は、http://onju.html.xdomain.jp/の中の index.htm ファイルが読み込まれる。

●JavaScript について

・サーバサイド言語

サーバ側で動く言語は、サーバサイド言語と呼ばれる。サーバサイド言語の主な仕事は、クライアントからのリクエストを解釈し、HTML などのレスポンスデータを作成してブラウザに返すことである。レスポンスデータを作成時には、必要に応じてデータベースとの通信を行うこともある。ブラウザは、サーバから返却されたレスポンスデータを解釈し、Web ページの表示を行う。
サーバサイド言語はいくつも存在し、日本では、PHP や Ruby などのサーバサイド言語が、多くのサービスで利用される。

・サーバサイド言語の使用例

メールの送信：フォームから問い合わせを送信したユーザへの自動送信
新規登録した人への自動送信
ログインなどの会員認証：SNS や EC サイトなどのログイン、ログアウト処理

・クライアントサイド言語

クライアント側（ブラウザ）で動く言語のことをクライアントサイド言語と言う。クライアント側で動くプログラミング言語は、現状では JavaScript が唯一だと言ってもよい。JavaScript の主な仕事は Web サイトの訪問者の操作などに応じてページの見た目を変更したり、ブラウザの代わりにサーバへリクエストを送ったりすることである。

なお、JavaScript はブラウザで動く言語なので、インストール作業が不要である。

■JavaScript を書き始める前に

●JavaScript を書く場所

JavaScript を書く場所は大きく分けて次の2つ

1. HTML ファイル
2. JavaScript ファイル

どちらに書くべきかはケースバーケースだが、2の場合が多い。※この演習ではファイルを増やしたくないので1で進める。

1. HTML ファイルの場合

HTML ファイルに JavaScript のコードを書いていく場合、「Script」タグの中にコードを書いていく。Script タグは head タグでも body タグの中であれば動くが、多くの場合 body の終了タグの直前に記述する。

```
001 <html>
002 <head>(省略)</html>
003 <body>
004   <!--Web サイトを構成する要素-->
005   (省略)
006   <script>
007    // ここに JavaScript を書く
008   </script>
009   </body>
010 </body>
011 </html>
```

2. JavaScript ファイルに書く

JavaScript ファイルにコードを書いていく場合は、JavaScript を記載するファイルを作成し、そのファイルを HTML から呼び出す必要がある。 script タグ内で外部の JavaScript ファイルを呼び出す場合には、下記のコードのように src 属性に JavaScript ファイルのパス情報を指定する。

```
001 <html>
002 <head>(省略)</html>
003 <body>
004   <!--Web サイトを構成する要素-->
005   <script src="呼び出したい JavaScript ファイルのパス情報"></script>
006 </body>
007 </html>
```

なお、JavaScript ファイルの拡張子は.js とし、JavaScript ファイルに書くコードは script タグでは囲まない。

●プログラムの作成上の基本ルール

・文字コードは UTF-8 にする

・半角英数字で記述する

・文(命令)の終わりには「;(セミコロン)」を付ける

・文字列は半角の「' (シングルコーテーション)」または、「"(ダブルコーテーション)」同士で囲む。

・インデントを付けて構造をわかりやすくする

・プログラムを説明するためのコメント文は、1行の場合は「//」を先頭に置き、複数行の場合は「/*」と「*/」で囲む。

●演算子

代入演算子		比較演算子		論理演算子	
=	代入	==	等しい	&&	かつ
算術・結合演算子		!=	等しくない		または
+	数値の足し算	<	未満		ではない
	文字列の結合	>	超過	増減演算子	
-	引き算	<=	以下	++	1増やす
*	掛け算	>=	以上	--	1 減らす
/	割り算	カンマ演算子		ドット演算子	
%	割り算の余り	,	式の区切り	.	～の

●変数

変数とは、数値や文字列などのデータを保管する箱のようなもので、オブジェクトごと変数に代入することもある。変数を使用するときは「var」を変数名の前につけて初めに宣言を行い、変数に値を代入するときは「=」を使う。

宣言の仕方

var 変数名; 変数=値;	または	var 変数名=値;

なお、ES2015以降は変数に let と const が使えるようになり、var より使われるようになっている。let と const の違いは下表のとおりである。

	let	const	var
初期値ありの変数宣言	○できる	○できる	○できる
再代入	○できる	×できない	○できる
繰り返し構文	○使える	×使えない	○使える
再宣言	×できない	×できない	○できる
スコープ	ブロック	ブロック	関数、グローバル

再代入ができる let、var は繰り返し構文で使えるが、再代入できない const では使えない。 スコープとは、宣言した変数を利用できる「範囲」のことで、let、const はブロックごとに作られるスコープの中で宣言したものをその外側で扱おうとするとエラーになる。 var は関数のブロック内で使える関数スコープや、すべてのスコープからアクセスできるグローバルスコープに該当し、外側で扱うことができる。

(参考)TECH PLAY 【JavaScript】let、const、var の違いと使い分け方法を徹底解説
https://techplay.jp/column/1619
桜庭洋之、望月幸太郎著「スラスラわかる JavaScript 新版」(翔泳社)

●DOM について

JavaScript の役割には大きく分けて以下の2つがある

1. Web サイト利用者の操作などに応じてページの見た目を変更する
2. ブラウザの代わりにサーバと通信を行う

この演習では1を基礎から学ぶ。1を学ぶにあたり、DOM(ドム:Document Object Model)は非常に重要な概念である。2は Ajax(エイジャックス)という技術を用いて実現するものだが、入門時点において難しい内容なので割愛する。

・DOM とは

まず、HTML についておさらいする。Web ページを作成するとき、最初に HTML という形式で Web サイトの骨格に当たる部分を構築する。この HTML ファイルをブラウザに展開することで、Web ページがブラウザ上に描画(レンダリング)される。

この時、ブラウザに HTML が表示されていると思っている人が多いが、実際はブラウザは HTML を読み込んだときに、「DOM」に変換した後、レンダリングをしている。DOM とは HTML などの文書を JavaScript のようなプログラムから扱えるようにするための仕組みを指す。

以下の HTML ファイルを例に考える

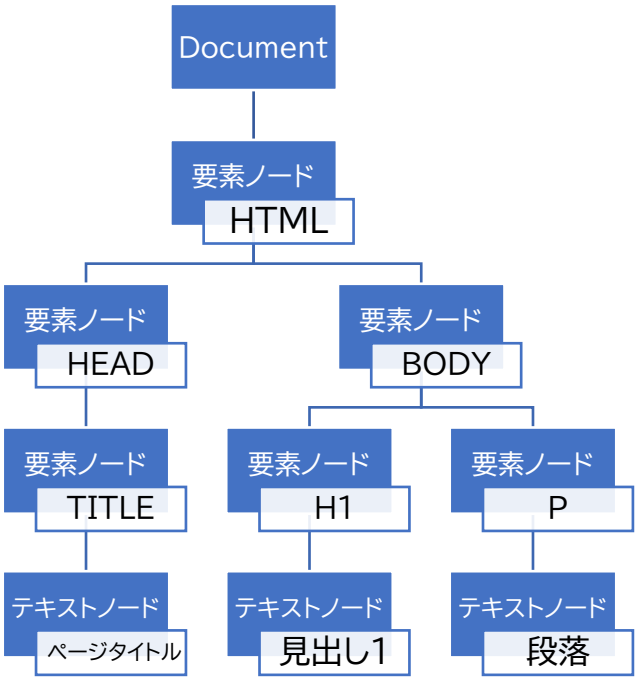
js0-1.htm

```
001 <!DOCUTYPE html>
002 <html lang=" ja" >
003 <head>
004    <meta charset=" UTF-8" >
005    <title>ページタイトル</title>
006 </head>
007 <body>
008   <h1>見出し 1 </h1>
009   <p>段落</p>
010 </body>
011 </html>
```

DOM では HTML 文書が次のようなツリー構造のドキュメントとして扱われる。document と言う情報を親に、HTML のタグやテキスト情報は DOM に変換される際に「ノード」という単位に分解される。

ノードには大きく分けて「要素ノード」と「テキストノード」が存在し、要素ノードは HTML タグ、テキストノードは HTML タグ内のテキスト情報に該当する。JavaScript で「ページの見た目を変更する」ことは「DOM を変更する」ことを意味する。

「DOM とは、JavaScript で Web ページに変更する仕組みである」ということを押さえておく。



（参考） 小笠原寛 著・知識ゼロからの JavaScript 入門（技術評論社）

●HTMLの要素の情報を変更して、文字を表示する

先ほどの HTML ファイルに JavaScript ファイルを追加して、「段落」の文字を変更する。

<p>タグに id 名を付ける。（id の値を text としている）

document.getElementById('text') で text の id 情報を持った要素の情報（つまり先ほどの id 名を付けた<p>の要素）を取得する。（document は JavaScript が元々持っているオブジェクトで、getElementById はそのメソッド（オブジェクトがプロパティとして持っている関数）になる）

getElementById の実行結果として、JavaScript から DOM を操作するためのオブジェクト（Element オブジェクト）が返り、それを el に代入している。（オブジェクトが代入されている）

innerHTML は Element オブジェクトが持つプロパティの一つで、要素が持つ情報（要素ノードやテキストノード）を変更することができる。今回は getElementById で取得した要素のテキスト情報を変更するので、「el.innerHTML」に新たに文字列を入れている。

js0-2.htm

```
001 <!DOCUTYPE html>
```

```
002 <html lang=" ja" >
003 <head>
004    <meta charset=" UTF-8" >
005    <title>ページタイトル</title>
006 </head>
007 <body>
008   <h1>見出し１</h1>
009   <p id=" text" >段落</p>
010 <script src="js0-2.js"></script>
011 </body>
012 </html>
```

大文字と小文字は区別される
〇getElementById
×GetElementByID
×GetelementbyiD

js0-2.js

```
001 var el = document.getElementById('text');
002 el.innerHTML='Hello JavaScript';
```

●HTMLの要素の情報を変更せず、ポップアップで文字を表示する

js0-3.htm

```
001 <!DOCUTYPE html>
002 <html lang=" ja" >
003 <head>
004    <meta charset=" UTF-8" >
005    <title>ページタイトル</title>
006 </head>
007 <body>
008   <h1>見出し１</h1>
009   <p>段落</p>
010   <script src="js0-3.js"></script>
011 </body>
012 </html>
```

js0-3.js

```
001 alert( 'Hello JavaScript' );
```

わざわざ JavaScript ファイルを参照させず、次のように書くこともできる

js0-3-2.htm

```
001 <!DOCUTYPE html>
002 <html lang=" ja" >
003 <head>
```

```
004     <meta charset=" UTF-8" >
005     <title>ページタイトル</title>
006   </head>
007   <body>
008     <h1>見出し１</h1>
009     <p>段落</p>
010     <script>alert( 'Hello JavaScript' );</script>
011   </body>
012   </html>
```

alertはJavaScriptがあらかじめ用意した関数の一つで、alertの後ろの()の中に引数として渡した情報をポップアップで表示させる。ここでは「'Hello JavaScript'」を引数として渡している。文字列として情報を渡したい場合は「'」(シングルクォーテーション)または「"」(ダブルクォーテーション)で囲まなくてはならず、囲まない場合は特別な意味を持った情報として解釈しようとする。ここでは['']で囲んでいるので、「Hello JavaScript」という文字列を表示させることになる。文末の「;」(セミコロン)は処理の区切りを意味する。

※数値＋数値　数値の「1」＋数値の「2」

```
009     <p>段落</p>
010     <script>alert(1+2);</script>
011   </body>
```

この場合、結果は「3」

※文字列＋文字列　文字の「1」＋文字の「2」

```
009     <p>段落</p>
010     <script>alert( "1" +" 2" );</script>
011   </body>
```

この場合、結果は「12」

※変数＋変数　aに数値の1、bに数値の2を入れてa+bの計算をさせる

```
009     <p>段落</p>
010     <script>var a,b;
011     a=1,b=2;
012     alert(a+b);</script>
013   </body>
```

この場合、結果は「3」

※文字列＋文字列　文字の「a」＋文字の「b」

```
009     <p>段落</p>
010     <script>var a,b;
011     a=1,b=2;
012     alert( "a" +" b" );</script>
```

```
013 </body>
```

この場合、結果は「ab」

●HTMLのリストの要素(li)を作成して、文字を表示する

js0-4.htm

```
001 <!DOCUTYPE html>
002 <html lang=" ja" >
003 <head>
004    <meta charset=" UTF-8" >
005    <title>ページタイトル</title>
006 </head>
007 <body>
008   <h1>見出し１</h1>
009   <ul id="name_list"></ul>
010   <script src="js0-4.js"></script>
011 </body>
012 </html>
```

js0-4.js

```
001 var ul = document.getElementById('name_list');
002 var li = document.createElement('li');
003 var text = document.createTextNode('情報太郎');
004 li.appendChild(text);
005 ul.appendChild(li);
```

HTML で箇条書きを書くコードは

```
<ul>
  <li>1 つめの項目</li>
  <li>2 つめの項目</li>
  <li>3 つめの項目</li>
</ul>
```

となり、

ブラウザでは、

・1つ目の項目
・2 つめの項目
・3つめの項目

と表示される。

参考サイト:サルワカ(https://saruwakakun.com/html-css/basic/ul-ol-li)

js0-4.js では、ul 要素の中に li 要素作り出す。
1 行目で、getElementById を利用して、id 情報から ul 要素の取得をして、
2 行目で、createElement(要素名)で、Element オブジェクト(li 要素)の作成をして、

3 行目で、createTextNode(テキスト情報)で、テキスト情報を作成して、
4行目で、appendChild で、3行目で作成したテキスト情報を li 要素のテキスト情報として追加している
最後に、5行目で、HTML 上に存在する ul 要素の子要素として、4行目で作成した li 要素を追加する。
ul.appendChild(li)が実行されると、ul 要素の子要素として、li 要素が追加され、プラウザのページ上から追加した要素が確認できる。

●配列を使って li 要素を生成する

HTML ファイルは下記のファイルを読み込むように、参照先のファイル名を変更する。

js0-4-2.js

```
001 var jname = ['情報太郎','情報次郎','情報三郎'];
002 var ul = document.getElementById('name_list');
003 for(var i=0;i < jname.length; i++){
004   var li = document.createElement('li');
005   var text = document.createTextNode(jname[i]);
006   li.appendChild(text);
007   ul.appendChild(li);
008 }
```

For 文を使って、document.createTextNode へ渡す引数を、配列から参照させている。

●クリック時にポップアップを表示する

js0-5.htm

```
001 <!DOCUTYPE html>
002 <html lang=" ja" >
003 <head>
004    <meta charset=" UTF-8" >
005    <title>ページタイトル</title>
006 </head>
007 <body>
008   <h1>見出し１</h1>
009   <p id="btn1">ここをクリック</p>
010   <script src="js0-5.js"></script>
011 </body>
012 </html>
```

js0-5.js

```
001 var jbtn = document.getElementById('btn1');
```

```
002 jbtn.addEventListener('click',function(){
003  alert('Hello JavaScript');
004 });
```

1 行目で、段落の要素を取得
2行目で、addEventListener というメソッドを使って、ユーザが段落をクリックしたときに実行したい処理を登録する

※addEventListener について
addEventListener は次のように使う

EventTarget.addEventListener(イベント名,リスナー);

addEventListener にはイベントが発生した時に実行してほしい処理を関数として渡す必要がある。「リスナー」はこの関数のことを指す。

今回の場合は、

p要素の Element オブジェクト.addEventListener(クリック,ボタンクリック時に実行したい処理);

と書かれている。
「EventTarget」は場合に応じて、document,windows,Element などが入り、今回は Element だが、document が入る場合もある。
addEventListener に渡している引数の一つ目は「click」だが、これは何らかの要素がクリックされたというイベント名で、文字列として渡す必要があるため、「"(ダブルクォーテーション)」で囲っている。このイベントも場合によってほかのものがある。
引数の二つ目はリスナーと呼ばれる関数で、一つ目の引数のイベント発生時に実行される処理を指す。また、リスナーのような「何らかのイベント発生時に実行するために、引数として渡される関数」はコールバック関数と呼ばれる。

●(＋α) jQuery を使って、コードを簡潔に書く

jQuery を使うと、コードが簡潔に書ける。
導入方法は、
① jQuery をダウンロードして使う
② CDN を使う
の2つで、①のメリットはオフラインで使えることで、②のメリットは外部のサイトから参照しているため、ファイルを用意しておかなくて済むことである。

①のやり方
https://jquery.com/download/
にアクセスをして、

Download the compressed, production jQuery 3.6.4 の上で右クリックして名前を付けてリンク先を保存を押して、HTML ファイルがあるフォルダに保存する。
ここでは、「jquery-3.6.4.min.js」が保存された

②のやり方でコードで書く。
jQuery のサイト https://releases.jquery.com/ にアクセスし、jQuery 3.x の「minified」をクリックして、
<script src="https://code.jquery.com/jquery-3.6.4.min.js"
をコピーして、(ヴァージョンが新しくなると、数字が変わる)
<script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>
とコードを HTML ヘッダーに記述する

(参考)
SAMURAI ENGEINEER
初心者でもわかる！jQuery の導入から基本的な使い方まで徹底解説！
https://www.sejuku.net/blog/104042

js0-6.htm

```
001 <!DOCTYPE html>
002 <html lang="ja">
003 <head>
004   <meta charset="UTF-8">
005   <title>ページタイトル</title>
006 </head>
007 <body>
008   <h1>見出し１</h1>
009   <p id="btn1">段落</p>
010   <script src="jquery-3.6.4.min.js"></script>
011   <script src="js0-6.js"></script>
012 </body>
013 </html>
```

js0-6.js

```
001 var jbtn = $('#btn1');
002 jbtn.on('click', function(){
003   alert('Hello JavaScript');
004 });
```

※jQuery の on を使わずに addEventLinstener を使う場合は js0-5.jsのコードの P 要素をボタン要素にして書く。

js0-6-2.htm、Js0-6-2.js 略

●段落とボタンを作り、クリックした対象に合わせて、表示する文字列を変える。

js0-6-3.htm

```
001 <!DOCTYPE html>
002 <html lang="ja">
003 <head>
004   <meta charset="UTF-8">
005   <title>ページタイトル</title>
006 </head>
007 <body>
008   <h1>見出し１</h1>
009   <p id="btn1">段落</p>
010   <button id="btn2">ボタン</button>
011   <script src="jquery-3.6.4.min.js"></script>
012   <script src="js0-6-3.js"></script>
013 </body>
014 </html>
```

js0-6-3.js

```
001 var jbtn1 = $('#btn'), jbtn2 = $('#btn2');
002 jbtn1.on('click', function(){
003   alert('段落がクリックされた');
004 });
005 jbtn2.on('click', function(){
006   alert('ボタンがクリックされた');
007 });
```

●(＋α) Vue.js3 を使う。

■Vue3とは

Vue.js は、Evan You 氏によって開発されたオープンソースのフレームワークで、現在「Vue3」と呼ばれる ver.3 がリリースされている。Vue3 は 2020 年9月に Vue2 からメジャーバージョンアップされたもので、しばらくは最新バージョンとして使われる。

Vue3 の特徴は、

① さまざまなデータが更新されると、自動的にそのデータを利用している画面の表示も更新される「リアクティブ」なプログラムであること、

② 「コンポーネント」と呼ばれる、部品として様々な機能を実装できること、

③ HTMLをベースとする「テンプレート」構文が用意されており、様々な表示をHTMLに埋め込むことができること

などがある。

(参考:掌田津耶乃・著「Vue. js3超入門」(秀和システム))

■Vue3を利用するには

Vue3を使うには、jQueryと同じ感覚で、下記の一文を書けばよい。

```
<script src="https://unpkg.com/vue@next"></script>
```

■Vue を使ってメッセージを表示させる

Hello Vue！という文字をブラウザ上に表示させるための簡単なプログラムを作る。ここでは、pタグに {{ message }} と入れ、これをメッセージに置き換えている。

js0-7.js

```
001 <!DOCTYPE html>
002 <html lang="ja">
003 <head>
004   <meta charset="UTF-8">
005   <title>ページタイトル</title>
006   <script src="https://unpkg.com/vue@next"></script>
007 </head>
008 <body>
009   <h1>見出し1</h1>
010   <p id="app">
011   {{ message }}
012   </p>
013 <script>
014   const appdata = {
015     data(){
```

```
016         return{
017           message: 'Hello Vue!'
018         }
019       }
020     }
021     Vue.createApp(appdata).mount('#app')
022 </script>
023 </body>
024 </html>
```

■実行結果

※ここでは、エディタでコードを書いたが、「Vue. js devtools」などの開発支援ツールを用意すると、本格的な開発を効率よく行うことができる。

■JavaScriptが学習できるサイト

mdn web docs ： JavaScript チュートリアル

https://developer.mozilla.org/ja/docs/Web/JavaScript