WEBアプリ エンジニアへの基礎知識

目次

[１　はじめに 3](#_Toc69940633)

[２　ＷＥＢアプリの概要 4](#_Toc69940634)

[２．１　サーバサイドとクライアントサイド 4](#_Toc69940635)

[２．２　ＨＴＭＬとＣＳＳとＪａｖａＳｃｒｉｐｔ間の関係 7](#_Toc69940636)

[２．３　フレームワークとは 9](#_Toc69940637)

[２．４　WEBエンジニアとして必要な知識 10](#_Toc69940638)

[３　開発環境 11](#_Toc69940639)

[４　画面デザインについて 12](#_Toc69940640)

[５　HTMLのルール 13](#_Toc69940641)

[６　CSSのルール 14](#_Toc69940642)

[７　DOMとは 15](#_Toc69940643)

[８　ＪａｖａＳｃｒｉｐｔのルール 16](#_Toc69940644)

[９　HTMLファイルを作成してみる 17](#_Toc69940645)

[１０　CSSファイルで見た目を整える 18](#_Toc69940646)

[１１　画面レイアウトを切り替える 19](#_Toc69940647)

# １　はじめに

　普段利用しているスマートホンやPCで見ている、ホームページやグーグルマップなど、ブラウザで見ているものの多くは、画像データや動画データ、HTML、CSS、JavaScriptなどを組み合わせて作成されています。

　WEB系アプリに対する需要は多く、対応できるエンジニアへの育成環境も急務となっています。

　エンジニアがある程度増えれば、チームとして業務を受注する事もできる筈・・・です。

　また、開発環境改善への要望もあり、HTMLやJavaScriptも毎年の様にルール改定がされており、以前は難しかった機能や、ブラウザ間での表現の誓いの緩和など、改善され続けています。

このため、全ての言語、ルールなどを覚えることは、現実的に大変であり、ラーニングコストもかかるため、割愛します。

最新仕様や使い方の詳細などは、WEB検索や、コミュニティなどを活用して対応願います。

　本資料では、各々のベースとなるような考え方を紹介し、WEBエンジニアになるための予備知識レベルのものを紹介します。

　WEBエンジニアへの一歩を踏み出す助けになれれば幸いです。

# ２　ＷＥＢアプリの概要

# ２．１　サーバサイドとクライアントサイド

1. 一般的なＷＥＢサーバの動き

ブラウザがサーバに対して特定データ（ＵＲＬ）のリクエストを送り、サーバがレスポンスデータ（ＨＴＭＬ）を編集し、レスポンスデータを送り返します。

フレームワークを使用しない一般的な動作の場合、ブラウザが行うリクエストと、サーバが返すレスポンスは一対一の関係になります。

ＷＥＢブラウザでＵＲＬを指定した時の動き

ＷＥＢサーバ側

リクエスト

（ＵＲＬ、要求元アドレスなど）

クライアント側（ＷＥＢブラウザ）

ＷＥＢ

アプリ

ＷＥＢ

アプリ

ＷＥＢ

アプリ

レスポンス

（ステータス、ＨＴＭＬデータ、要求元アドレスなど）

データベースなど

　＜ＵＲＬでの指定の例（ＣＹＮＣＳトップページ）＞

https://cyncs.technopro.com/ctec.html

① 　　　　 ②

1. ドメイン（ＩＰアドレスをＤＮＳサーバで置き換える）
2. フォルダ名 or ファイル名。省略時は“ｉｎｄｅｘ．ｈｔｍｌ”になる。

　　①は、ＷＥＢサーバ側の公開フォルダに割り付いているイメージです。

　　実際、②を抜いて指定すると、メニューページが表示されます。

＜ＣＹＮＣＳのindex.html＞



　また、ＵＲＬの中にクエリパラメータ（関数での引数に相当）を指定することもあります。

　＜クエリパラメータを使用したＵＲＬの例＞

[ｈｔｔｐ：//○○/△△/](http://○○/△△/)ｉｎｄｅｘ？ｎａｍｅ＝ｈｏｇｅ＆ａｄｒ＝ｓｅｎｄａｉ

　　　　　　　　　　　　　　　　　⇓

ＷＥＢアプリへ“ｎａｍｅ＝ｈｏｇｅ、ａｄｒ＝ｓｅｎｄａｉ”を渡す。

　ただし、クエリパラメータを多用すると、ＵＲＬが長くなってしまいます。

1. フレームワークでのＷＥＢサーバの動き

# ２．２　ＨＴＭＬとＣＳＳとＪａｖａＳｃｒｉｐｔ間の関係

　ＷＥＢブラウザで表示しているページは、基本的に３つの概念で成り立っています。

1. 内容（コンテンツ）
2. 表現（プレゼンテーション）
3. 動作（ビヘイビア）

　内容はＨＴＭＬ、表現はＣＳＳ、動作は主にＪａｖａＳｃｒｉｐｔが相当します。

1. 内容（コンテンツ）

ＷＥＢブラウザに表示したい文章（記事）や写真、動画などを指します。

ただし、そのまま張り付けただけでは、それが記事なのか、メールアドレスなのか、図なのか、ＷＥＢブラウザでは判別できません。

このため、“タグ（意味のあるマーク）”と“属性（付帯情報）”を使って、書かれている内容に意味づけをするなど、グループ分けなどを行います。

ＨＴＭＬファイルは、１ページ１ファイルで作成します。

1. 表現（プレゼンテーション）

文字サイズや配色、表示レイアウトなど、コンテンツの見た目を指定します。

コンテンツに付けられたタグを使って対象を決め、対象の文字サイズや色、書体やフォント指定、背景色などを変更します。

また、コンテンツ間の並び方（右寄せ、左寄せ、均等割付、折り返し有無、折り返し方向など）なども指定します。

基本的には静的な定義ですが、画面幅を取得することで、スマートホン用のレイアウトとＰＣ用のレイアウトを指定することもできます。

上記指定は、ＨＴＭＬ内に埋め込むことも出来ますが、規模が大きくなるにつれ煩雑になり、保守性が大きく下がってしまうため、ＣＳＳで制御するのが一般的です。

ＣＳＳファイルは、ＨＴＭＬのheadタグ内で読み込みます。（複数可）

1. 動作（ビヘイビア）

内容と表現に対し、イベント（ユーザの操作など）に対応した振る舞いを定義します。

ＨＴＭＬとＣＳＳでも、ページの移動やレイアウトの変更、背景色などを変更することもできますが、変更できる機能は限定的です。

ビヘイビアでは、ＷＥＢブラウザがＨＴＭＬを読み込んだ際に内部作成されるＤＯＭ**（※１）**を使い、ＪａｖａＳｃｒｉｐｔなどでタグや属性状態を調べるなど、内容を書き換えることにより、表示内容を動的に変更します。

ＪａｖａＳｃｒｉｐｔファイルは、ＨＴＭＬのheadタグ内で読み込みます。（複数可）

1. 概略図

＜ＷＥＢページの元イメージ＞

**タイトル**

・・・・・・・

・・記　事・・

・・・・・・・

内容に合わせてタグを付ける

＜ＣＳＳファイル＞

h1 { ← ｈ１タグを選択

color =”green” (文字をgreenで指定)

}

p { ← ｐタグを選択

font size=”16px” (文字ｻｲｽﾞを16pxで指定)

｝

＜ＨＴＭＬファイル＞

<h1> ←（大見出しタグ開始）

“タイトル”

</h1> ←（大見出しタグ終了）

<p> ←（段落タグ開始）

・・・記　事・・・

</p> ←（段落タグ終了）

<img “src=(図の参照元)”>

コンテンツの見た目を指定

　　＜ＤＯＭイメージ＞

html

h1

　　“タイトル”

　　　　color : ”green”

　　p

　　　　“・・・記　事・・・”

　　　　font size : “16px”

WEBブラウザが、HTMLとＣＳＳを解析してＤＯＭツリーを生成

ＷＥＢブラウザがＤＯＭの内容を画面に表示

イベント発生時にＤＯＭの内容を書き換える

＜ＪａｖａＳｃｒｉｐｔファイル＞

let kiji = document.getElementByTagName(“p”);

（DOMのＰﾀｸﾞ要素を指定）

kiji.addEventListener("mouseover",function( event )

(Ｐﾀｸﾞにﾏｳｽが重なった時のｲﾍﾞﾝﾄで、実行する)

{

event.target.style.backcolor = “orange”;

(記事の背景色をorangeに変える)

｝;

**タイトル**

・・・・・・・

・・記　事・・

・・・・・・・

# ２．３　フレームワークとは

# ２．４　WEBエンジニアとして必要な知識

　WEBエンジニアになるため、下記の知識習得が必須になります。

　１　HTMLの基礎知識

　２　CSSの基礎知識

　３　ＪａｖａＳｃｒｉｐｔの基礎知識（DOMの基礎知識含む）

　あくまでも“基礎知識”です。全てを覚えると大変時間がかかりますので、必要知識の範囲外とします。

　その代わり、感覚を掴むために、各々の入門書を２～３回読み直しましょう。

　ＶｕｅＪＳなどのフレームワークでは、便利な機能を搭載していますが、基礎知識があることが前提です。

# ３　開発環境

# ４　画面デザインについて

# ５　HTMLのルール

# ６　CSSのルール

# ７　DOMとは

# ８　ＪａｖａＳｃｒｉｐｔのルール

# ９　HTMLファイルを作成してみる

# １０　CSSファイルで見た目を整える

# １１　画面レイアウトを切り替える