Webアプリ エンジニアへの予備知識

～　ソフト０８ ＷＥＢエンジニア化プロジェクト資料　～

目次

[１　はじめに 3](#_Toc70162410)

[２　Webアプリの概要 4](#_Toc70162411)

[２．１　WebサイトとWebアプリ 4](#_Toc70162412)

[２．２　サーバサイドとクライアントサイド 5](#_Toc70162413)

[２．３　ＨＴＭＬとＣＳＳとＪａｖａＳｃｒｉｐｔ間の関係 10](#_Toc70162414)

[２．４　フレームワークとは 12](#_Toc70162415)

[２．５　Webエンジニアとして必要な知識 13](#_Toc70162416)

[３　開発環境 14](#_Toc70162417)

[４　画面デザインについて 15](#_Toc70162418)

[５　HTMLのルール 16](#_Toc70162419)

[６　CSSのルール 17](#_Toc70162420)

[７　DOMとは 18](#_Toc70162421)

[８　ＪａｖａＳｃｒｉｐｔのルール 19](#_Toc70162422)

[９　HTMLファイルを作成してみる 20](#_Toc70162423)

[１０　CSSファイルで見た目を整える 21](#_Toc70162424)

[１１　画面レイアウトを切り替える 22](#_Toc70162425)

# 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アプリについては、２．２で説明します。

　ＨＴＭＬ、ＣＳＳ、ＪａｖａＳｃｒｉｐｔの説明は、２．３項などで説明します。

　Web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。

　ＷＥＢサイト作成には、幅広い知識が必要となるため、最初は広く浅く覚えておき、用途に合わせて調べましょう。

　ＪａｖａＳｃｒｉｐｔの規格は、ここ数年、毎年更新されています。　本格的なエンジニアとなった際は、最新情報に気を付ける事が重要です。

ただし、ビギナーはやはり基本が大事！！各種入門書で地力を付けましょう。

　本資料では、各々のベースとなるような考え方、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。

　普通の入門書ではあまり紹介出来ない、ウンチクレベルの物を中心に記載しました。

　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　WebサイトとWebアプリ

　WebサイトとWebアプリの間に明確な線引きはありませんが、ざっくりした切り分けはあります。

Webサイト：静的なページ。情報提供するだけのホームページやWebページを指します。

　　ＨＴＭＬとＣＳＳで、表示する内容とデザインをＷＥＢブラウザに伝えます。

例：　ウィキペディアなど。（宣伝部分は除く）

Webアプリ：本来静的なWebサイトを、動的に加工します。（注　動画にする訳ではありません。）

ユーザの操作や時間経過などをイベントとして捉え、イベント発生時のプログラムを定義し、ページ内の内容を変更します。

データベースサーバと連携し、検索を行ったり、ボタン操作でメニュー表示の切り替えを行ったりと、プログラマブルに様々な制御を行います。

クライアント側で操作する場合は、WEBブラウザの内部データ（DOMデータ、DOMツリーとも呼びます）の内容を書き換えることで、表示を変更します。

サーバ側で操作する場合は、クライアント側に送信するHTMLデータそのものを変更することも出来ます。

Webアプリは、Webブラウザが有れば表示出来るため、インストールは行いません。

　　　　　　 　例：　検索サイトなど。

アプリ、ネイティブアプリ：通常のアプリです。

App Storeなどでインストールして用います。

　例：　新型コロナ感染防止アプリ“ＣＯＣＯＡ”など

　Webアプリエンジニアは、Webアプリの開発を行います。

　又、Webアプリエンジニアには、フロントエンドエンジニアとバックエンドエンジニア（又はフルスタックエンジニア）がいます。

　フロントエンドエンジニアは、端末側により近いため、デザイン関連の知識が必要になります。

　バックエンドエンジニアは、サーバの配置や保守など、システム全体に及ぶ知識が必要になります。

　何れも幅広い知識が必要となり、人手が不足していると言われています。（需要大。）

# ２．２　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

ブラウザのアドレス欄にＵＲＬを指定すると、リクエスト情報（※）がＷＥＢサーバに送信され、ＷＥＢアプリなどに通知されます。

ＷＥＢアプリなどは、要求された内容を元に情報収集し、要求されたリソース（ＨＴＭＬデータや画像データなど）をレスポンス情報（※）としてブラウザに返信します。

通常、このやり取りを何度か行い、１つのＷＥＢページが表示されます。

* リクエスト情報とレスポンス情報も、後で説明します。

WebブラウザでＵＲＬを指定した時の動き

Webサーバなど

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

レスポンス

（ステータス、ＨＴＭＬデータ、要求元アドレスなど）

データベースサーバ

Web

アプリ

JavaやJavaScript、PHP、Python、Rubyといったプログラミング言語があり、

　時々、”フロントエンド”、”バックエンド”と言う表現を使いますが、端末側を“フロントエンド”または“クライアントサイド”、　サーバ側を“バックエンド”または“サーバサイド”と呼びます。

　フロントエンドのエンジニアを“フロントエンドエンジニア”、バックエンドのエンジニアを“バックエンドエンジニア”または“フルスタックエンジニア”と言います。

フロントエンドエンジニアは、Webアプリケーションなどでユーザーが画面越しに触れる部分、つまりフロントエンドの設計や構築を行う職種です。 フロントエンドエンジニアはデザイナーが設計したデザインを元にHTML、CSS、JavaScriptを駆使して、ブラウザに表示できるようにコーディングを実施します。

フロントエンドでの開発で使う言語は、HTML、CSS、JavaScriptです。

バックエンドエンジニアは、システムやサイトを利用するユーザーの目には見えることのない部分を担当する職種です。 サーバーサイドや、データベースの処理をおこなうなど、影に隠れた部分の処理を担当する役割がメインです。

バックエンドでの開発で使う言語は、JavaやJavaScript、PHP、Python、Rubyなどです。

　フルスタックエンジニアは、WEB開発やシステム開発、運用から保守まで、エンジニアリングの業務におけるすべての工程を1人でできるエンジニアのことです。

　一人で様々な業務を抱え込むため、まぎれもなく激務になります。

　通常のＷＥＢアプリは、サーバ側でデータを編集し、端末側にＨＴＭＬデータなどを送り返しますが、リクエストの都度やり取りが発生するため、画面表示に遅延が発生することが発生しています。

　このため、一部のフレームワークでは、端末側でレスポンスを編集するものも出てきています。

1. フロントエンドでのＷｅｂアプリ（ＶｕｅＪＳなど）

WebブラウザでＵＲＬを指定した時の動き

Webサーバなど

データベースサーバ

Web

アプリ

クライアント側（Webブラウザ）

Web

アプリ

Web

ブラウザ

リクエストとレスポンスは、端末内で折り返す。（ＨＴＭＬなど）

Web

アプリ

Ｗｅｂアプリ間では、データだけやり取りする。

1. リクエスト情報やレスポンス情報について

　ブラウザがWebアプリにリソースを要求する場合、下記の動作になります。

　　１　ブラウザからWebアプリに処理をリクエストする。

　　２　Webサーバが、静的コンテンツ（ｃｓｓや画像など）か動的コンテンツかを判断し、

　　　　静的コンテンツであればＷｅｂサーバでレスポンスデータを返信する。

　　　　動的に作り出すコンテンツであれば、ＡＰサーバに処理を依頼する。

　　３　ＡＰサーバ（アプリケーションサーバ）は、必要であればデータベースからデータを

　　　　取得し、レスポンスデータを編集し、Ｗｅｂサーバに返信する。

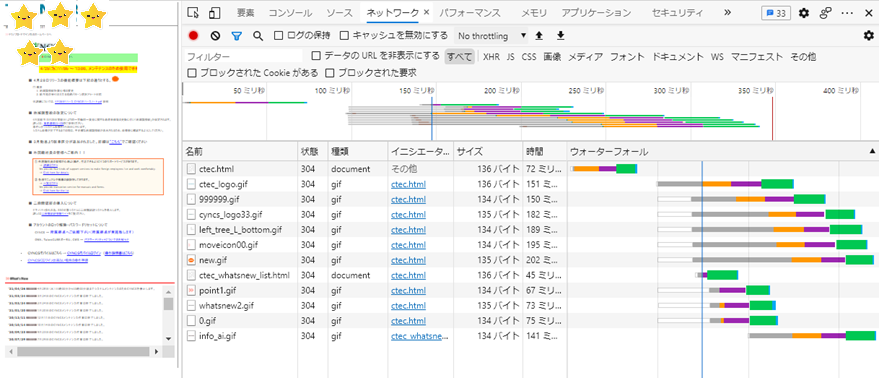
　　４　ＡＰサーバに依頼した場合は、ＡＰサーバからのデータをＷｅｂサーバが返信する。

　　５　ブラウザは、Ｗｅｂサーバからのレスポンスデータを元に、ページを表示する。

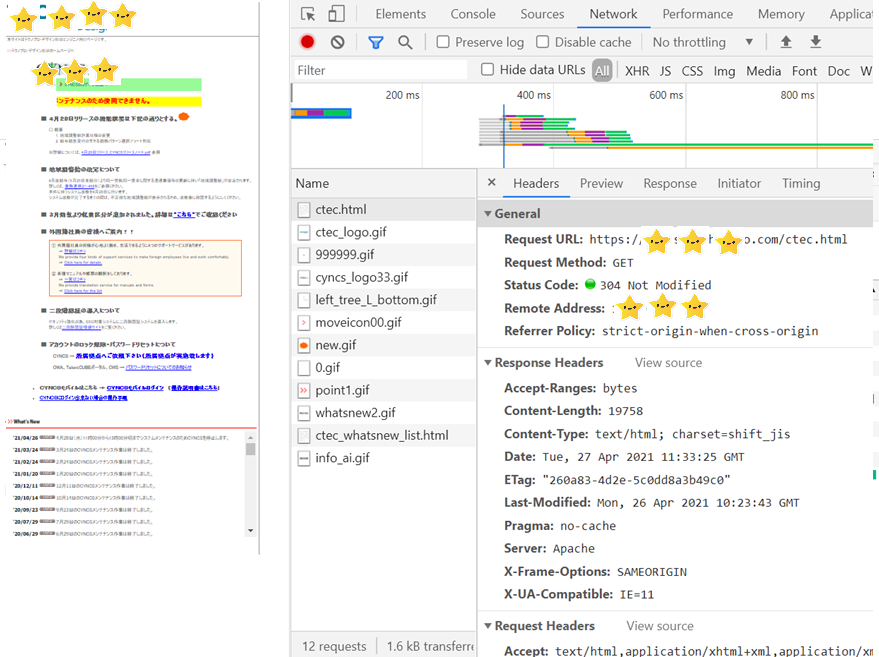
　　ブラウザからＷｅｂサーバへの要求を“リクエスト”、応答を“レスポンス”という。

＜身近な某サイトの例＞

クロームやエッジでF12（デベロッパーツール起動）を押し、“ネットワーク”タブ選択後、F5を押した所。トップページ（ctec.html）の後、gifファイルなどがやりとりされている　。



クリックすると詳細が見える。



GETクエリ

1. URLについて

URLは、Webサイトのアドレスのようなものです。

ブラウザでURLを指定すると、Webサーバなどの一意な場所が指定され、データを返信します。

＜ＵＲＬでの指定の例（ＣＹＮＣＳトップページ）＞

　④

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

　　　　　　　　③　　　　　　　　①　　　　　　　　　　②

1. ドメイン名（ＩＰアドレスをＤＮＳサーバで置き換える）
2. ディレクトリ名 or ファイル名。
3. プロトコル名（通信規約の種類）
4. ＵＲＬ (Uniform Resource Locator)

単純なＷＥＢサイトの場合、②の部分はＨＴＭＬファイル名と一対一になります。

＜公開フォルダ内の構成＞

（公開フォルダ）\index.html ・・・　メニューページ

　　　　　　　　　　\ctec.html　　 ・・・　ＤＳ用ページ

　　　　　　　　　　\home\demo.html　　 ・・・　デモページ

＜ＵＲＬでの指定＞

<https://cyncs.technopro.com/index.html> ・・・　メニューページのＵＲＬ

[https://cyncs.technopro.com](https://cyncs.technopro.com/index.html) ・・・　メニューページのＵＲＬ（省略時）

<https://cyncs.technopro.com/ctec.html> ・・・　ＤＳ用ページのＵＲＬ

<https://cyncs.technopro.com/home/demo.html> ・・・　デモページのＵＲＬ

＜イメージ図＞

Ｗｅｂサイト

クライアントのブラウザ ‘xx’サイト

xx/menu

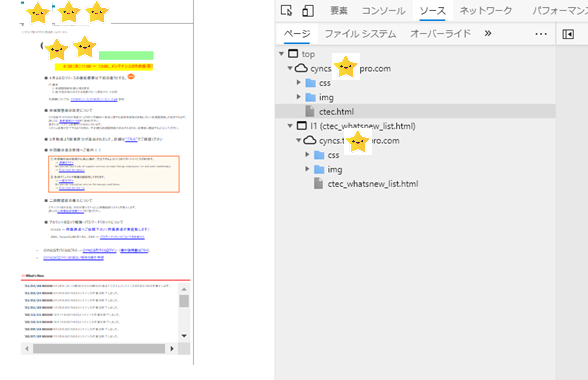
‘menu’のHTML

xx/naiyou

‘naiyou’のHTML

又、実際に運用されているWebページを見ると、１つのページ内を幾つかのHTMLファイルで表現している事もあるようです。

＜身近な某サイト＞



クロームやエッジで“F12”ボタンを押すと、開発用ツールが表示され、上記の情報が見えます。

『単純なＷＥＢサイトの場合、②の部分はＨＴＭＬファイル名と一対一』と書きましたが、フレームワークでは、②部分とキーワードが一致した時に関数コールするテーブルを定義し、

キーワードを元にＷｅｂページを組み立てて返信するものも在ります。

＜フレームワーク“Ｄｊａｎｇｏ”での例＞

(urls.py内)

urlpatterns = { 　　　　　　キーワード　　呼び出す関数名など

path( ‘hello/aaa’, hello.aaa ), 　←　hello/aaa hello.aaa関数

path( ‘hello/bbb’, hello.bbb ), 　←　hello/bbb hello.bbb関数

path( ‘ccc/’, ccc.index ),　　　　　←　ccc/ ccc.index関数

}

キーワードと動作を紐づけるテーブル。

②とキーワードが一致した時、子アプリや関数を呼び出す。

子アプリの場合、子アプリ内のurls.pｙで、呼び出す関数を決める。

呼び出された関数では、使用するＨＴＭＬのテンプレートを指定したり、変数の内容を書き換えたり、データベース情報を更新したりし、最終的にレスポンスデータを出力する。

＜イメージ図＞

Ｗｅｂアプリの例（フレームワーク“Ｄｊａｎｇｏ”）

クライアントのブラウザ

②とキーワードが一致したら、関数を呼び出す。

関数内でHTMLの内容を組み立てる。

“hello”アプリ

(urls.py内) urlpatterns

aaa関数

path( ‘hello/aaa’, hello.aaa )

bbb関数

path( ‘hello/bbb’, hello.bbb )

“ccc”アプリ

path( ‘ccc/’, ccc.index )

Index関数

　　HTTPリクエスト HTTPレスポンス

データベース 呼び出された関数内でテンプレート（※）や

コンテンツを組合せ、レスポンスデータを生成する。

　　　　　　　　　　　　（呼び出されてから作成する。）

　　　　 ※　HTML＋CSSで作成した、ページのベース。

　　ページの固定的な部分だけ定義する。

ブラウザ

urls.py

views.py

template

models.py

②の部分には、ページへの付帯情報を追加することが出来ます。

動的ページへ付加する場合、ページデータ要求（クエリ）に付ける情報のため、“クエリパラメータ”と呼びます。

＜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(index)へ、“ｎａｍｅ＝ｈｏｇｅ、ａｄｒ＝ｓｅｎｄａｉ”を渡す。

＜検索サイトで”technopro”を検索した時のページのURL＞

https://www.google.co.jp/search?q=technopro&sxsrf=ALeKk029oT4LVpqg8IYxQTsUmiehXgusg%3A1619150247553&source=hp&ei=p0WCYImiH4vWmAX9tqnwBQ&iflsig=AINFCbYAAAAAYIJTt6QTBmpLYAbqWenDNsqw\_zaLugyp&oq=technopro&gs\_lcp=Cgdnd3Mtd2l6EAMyAggAMgIIADICCAAyAg　・・・

　静的ページに付加する場合、無効値として“ダミーパラメータ”と呼ばれます。

　ダミーパラメータは、どのページから呼ばれたかを集計する際に、リンク元に識別用データを付与しておく、などの使い方をします。

クエリパラメータ、ダミーパラメータの何れでも、‘？’が識別子になります。

# ２．３　ＨＴＭＬとＣＳＳとＪａｖａＳｃｒｉｐｔ間の関係

1. 概略図

＜Webサイトの元イメージ（一ページ分）＞

**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・・

・・・・・・・

　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（ビヘイバー）

　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するなど、グループ分けなどを行います。

ＨＴＭＬファイルは、一ページ一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（ビヘイバー）

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。

ビヘイバーでは、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　ＷＥＢブラウザの内部データで、これを元に表示します。ツリー構造のデータ。

# ２．４　フレームワークとは

# ２．５　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。

　通しで読み直すと、各々の関連性が見えてきます。

　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。

　フレームワークで生成されるのは、車でいえば車体だけ。製品にするには、塗装をしたり、エアバッグを付けたり、カーナビを付けたりと、仕上げ作業が必要です。

　基本が判らないと、『言われたことしか出来ない』どころか、『言われたことが判らない』ことになります。（仕事になりません。）

　色付けなどで悩まなくて良いよう、基本をしっかり身に付けましょう。

# ３　開発環境

# ４　画面デザインについて

『ＷＥＢサイトのデザインは、ＷＥＢデザイナーが決める！！』　が理想だと思いますが、個人或いは小規模な物件などは、エンジニアがデザイン決めから作業することがあるかもしれません。

Ｗｅｂデザイナーのインターネットスクールのサイトから転記します。

<https://www.internetacademy.jp/it/design/web-design.html> より

Webデザイナーの仕事内容

Webデザイナーの主な仕事内容は、企業や個人などのクライアントから依頼されたWebサイトのデザインを担当することです。クライアントが思い描いているWebサイトを制作できるよう、全体の構成を考えながらデザインを構築していきます。それでは、Webデザイナーの業務内容を具体的に見ていきましょう。

１．Webサイトの構成とレイアウトを決める

まずは、制作するWebサイトの大まかなレイアウトを決めます。基本的な構成や全体のボリュームなどを考慮し、キーとなるビジュアルやWebサイト全体のイメージを決定します。

２．Webサイトのデザインを作る

続いて行うことは、Webサイトの色合いや装飾決めです。「Illustrator」や「Photoshop」などのグラフィックソフトを用いて、Webサイトの配色やロゴ・アイコンの配置などを決めます。基本的に、ここまでの作業はクライアントと相談を重ねながら進めることが一般的です。

３．Webサイトのコーディングを行う

Webサイトの大まかなレイアウトにクライアントの合意が得られた後、Webサイトを構成する言語であるHTML、CSS、JavaScriptを 使用してコーディングを行います。HTMLは文章や画像を表示し、CSSは配置、フォント、文字サイズ、色などを指定し、JavaScriptで動きを表 現します。 ユーザーの使いやすさや見た目の美しさも意識しつつ、事前に決めた大まかなレイアウトに沿ってロゴやアイコンなどのデザインを行います。最終的なデザイン の調整はミリ単位で行われるため、Webデザイナーには集中力と根気が不可欠です。

パソコンと向き合う時間が長いと思われがちなWebデザイナーですが、クライアントとのミーティングなど、人と接する機会も多くあります。Webサイトに対するクライアントの要望をしっかりとヒアリングする必要があるため、デザイン能力に加えてコミュニケーション能力も試される仕事です。

Webデザイナーになるには？

Webデザイナーになるために、特別な資格は必要ありません。知識とスキルがあれば、未経験でも仕事に就くことができます。もちろん、過去にWeb関係の業務経験があったり、あらかじめWebスクールに通い高度なスキルを身につけていたりする場合には、即戦力として歓迎されます。Webデザイナーを志す場合は、HTMLとCSSに関する基礎的な知識が最低限必要です。HTMLとCSSは、Webサイトの構築やデザインをする際に用 いられる言語です。比較的簡単な言語であるため、基礎的な書き方は簡単に身につけることができます。マークアップ言語に加えて、 「Illustrator」や「Photoshop」の使い方も学んでおきましょう。

# 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