昨今のフロントエンド事情

研修では静的ページをメインに学習してきました。

研修が終わったかもう勉強しなくて良いではなく、自身の知識、技術力向上に努めてください。

それを実務で生かし、経験を積めば自身の単価アップ、選べる案件が増える可能性もあります。

IT業界は常に新しいものが生まれているのでキャッチアップする習慣をつけましょう。

JSフレームワーク

昨今ではバックエンドの実装があるフロント開発でJSフレームワークを採用するケースが増えてきました。

代表的なフレームワークは...

- React
- Vue
- Angular

動作が早いやコンポーネント開発ができるものもちろんですが、jQueryを使用した場合だと、DOMの操作をすべて開発者側でコーディングする必要がありました。

たとえば、入力フォームと送信ボタンがあったとします。

入力フォームが未入力の場合、送信ボタンが押せない仕様だとします。

jQueryの場合

1. 変更イベント検知のために入力フォームのDOMを取得する必要がある
2. 入力フォームの変更を検知する
3. 入力フォームの長さを判定する
4. その結果によって送信ボタンのdisabled属性を書きかえろと命令する

JSフレームワークの場合

1. 入力フォームの変更を検知する
2. 入力フォームの長さを判定する
3. その結果によって送信ボタンのdisabled属性を書きかえろと命令する

文字だけだと理解し難いですが、JSフレームワークの場合だと1度もDOMの取得をしていません。

また「その結果によって送信ボタンのdisabled属性を書きかえろと命令する」の処理ですが、jQueryの場合だとif、elseで`true`を代入する処理と`false`を代入する処理をそれぞれ書く必要があります。

JSフレームワークの場合、`true`になる条件を書くだけで完成します。（条件に当てはまらなかった場合は`false`を返すので）

と言ったようにDOMを気にせずロジック実装が行えます。

また、PHP、Ruby on Railsと言ったフレームワークで開発となるとフロント側は、サーバーサイドフレームワークに準じたテンプレートを使用してフロント開発を行わななければならない。

簡単に説明するとHTMLとサーバーサイト言語が混在した状態でフロント開発を行うイメージ。

フロントエンジニアはサーバーサイドを使用する機会が少ないのでそう言った開発ではサーバーサイドのコードを読む必要が出てくるなど学習コストがかかってきます。

JSフレームワークを使用すると...

サーバーサイドの役割

JSON形式のレスポンスを返す「APIサーバ」としての役割

フロント側から指定したエンドポイントのAPIを実行し、それに対応した値を返す役割

フロントエンドの役割

ルーティングやレンダリングなどすべてを担う

フロントの仕事量は増えてしまいますが、お互いの得意領域に専念して開発が行えるようになり、APIのレスポンスもフロント側で自由に加工ができます。

ただ、想定したAPIとできあがったAPIが違うこともあるのでバックエンドとやり取りがまったくない訳ではない。

案件としては既存のシステムをフロント側をJSフレームワークに作り替えたものへ移行さえる案件が最近では多いです。（リプレース案件）

研修課題として作るかはまだ未定ですが、勉強しておいて損はないので勉強してみてください。

個人的なオススメはReactかVueです。（案件、日本語のドキュメントが多いので）

TypeScript

最近採用する会社、案件が少しずつ増えてきています。

JavaScriptとの大きな違いは、JavaScriptが動的型付けに対して、TypeScriptは静的型付けを行なっている部分です。

たとえばバックエンドから値を受け取ってそれを変数に代入するとします。

```
// バックエンドの値:RETURN_VALUEに100が格納されてフロント側が受け取るとする
RETURN_VALUE = 100

// JavaScript
const value = RETURN_VALUE

// TypeScript
const value: number = RETURN_VALUE
```

変数宣言を見た時にJavaScriptはなんの値を受け取っているのか分かりません。

確認する方法としては、コードの下に`console.log()`を書いて実際に動かして確認する方法です。

それに対してTypeScriptは`value: number`とnumber型が割り当ててあるので動かさなくてもnumber型を受け取るのが分かります。

変数だけでなく、オブジェクト、配列、引数、メソッドの返り値などにも型を割り当てることができます。

VsCodeを使用している場合は、型推論と言ってVsCode側で自動的に型を割り当てる機能があるので明示的に指定しなくても型の恩恵を受けることができます。

これにより、途中から参加する案件などでもコードを読み取る時にどう言った型の値を取り扱っているのかが動かさなくてもある程度把握できます。

また、実装時にJavaScriptだと動かしてみるまで分からないバグもTypeScriptだと動かす前にコード上に表示してくれるのでローカルサーバーを再起動させることなく修正ができます。（動かさないといけない部分も多少はある。）

もう1つ違いを挙げると、JavaScript側で今後の更新で追加されるであろう機能をTypeScriptはすでに追加してある部分です。

場合によっては、JavaScript側で複雑な処理を自分で考えて作らなければならない処理をTypeScriptではメソッドを使うことで簡単に実装できる場合もあります。

以上のことからJavaScriptよりもTypeScriptの方がバグが発生しづらく安定感があります。

また、昨今ではJSフレームワークと一緒に導入することも増えたのでこちらも勉強しておいて損はないです。