

フロントエンド入門 Vue.js編

2017年3月15日

オトナのプログラミング勉強会
協力 未来会議室

自己紹介

- 村上 卓
- 29歳
- フリーランス
- AngularJS(1系)/Ruby On Rails

モダンフロントエンドはこれから勉強

この勉強会について

オトナのプログラミング勉強会

<http://otona.pro>

- 2016年8月から開始
- 月2回（第1水曜、第3水曜）
- ジャンル（？）
 - プログラム言語、機械学習、Web系...

フロントエンドとは

Vue.js

The Progressive JavaScript Framework

- Approachable (親しみやすい)
- Versatile (融通が効く)
- Performant (高性能)

Vue.jsの特徴

- ビューに特化
- 軽量
- Virtual DOM (仮想DOM)
- コンポーネントシステム
- 日本語ドキュメント

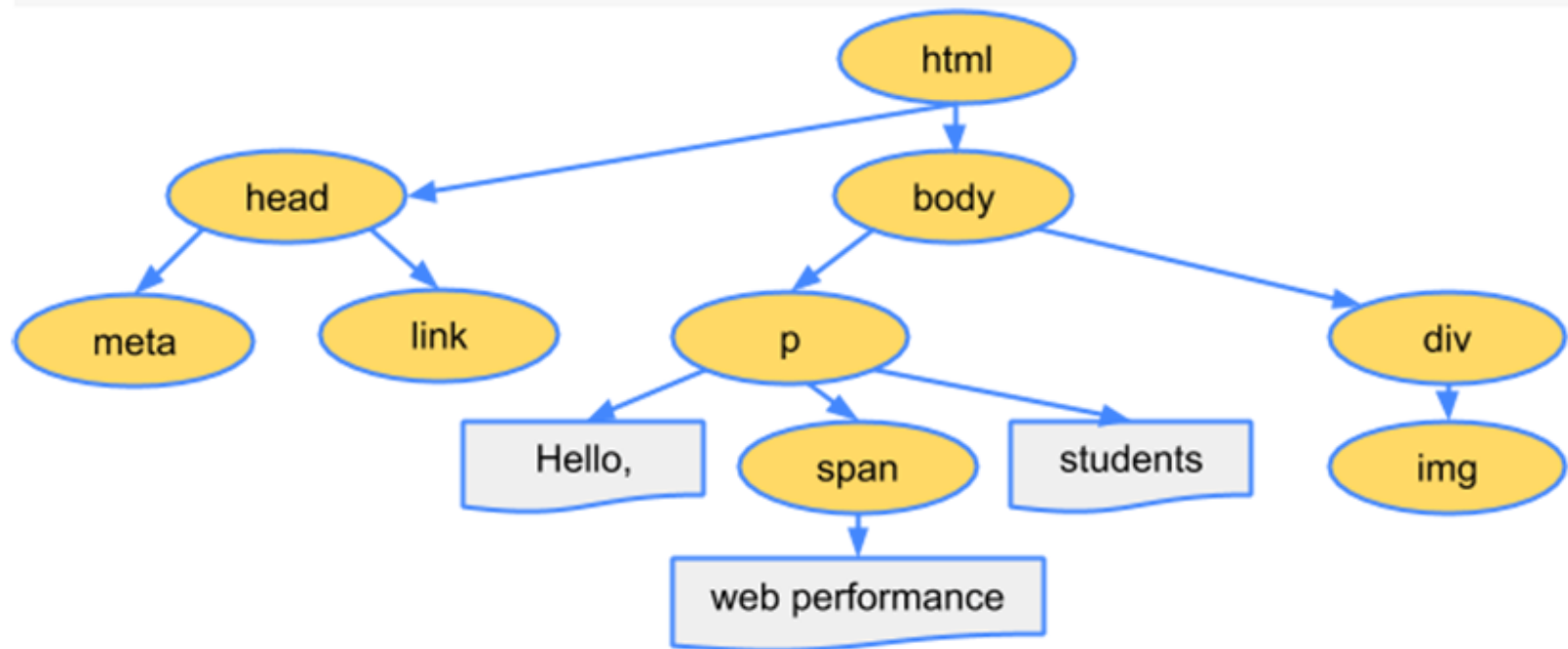
ざっくりブラウザ 周りのお話

DOMとは

- Document Object Model
- ドキュメントツリーを操作するAPI
- DOMを利用することでHTMLを自由に操作できる
- ブラウザはドキュメントツリーを作成後表示処理を行う

ドキュメントツリー

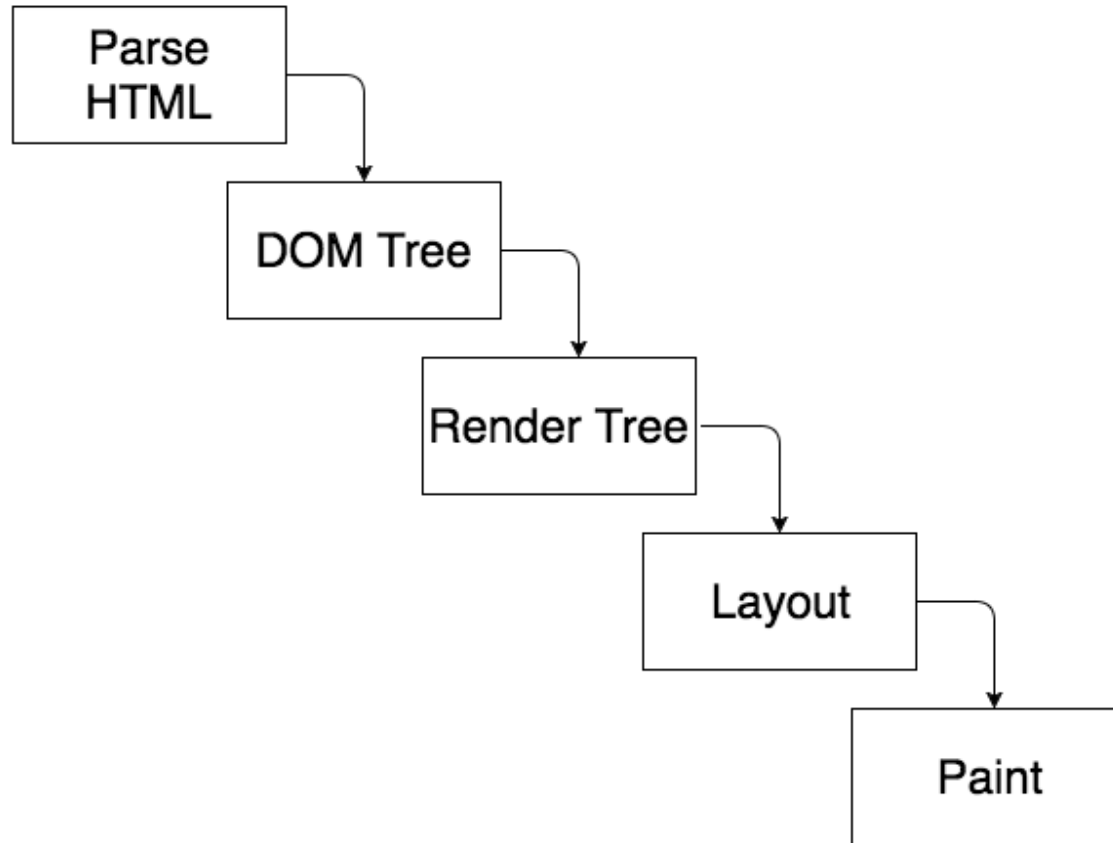
```
<html>
  <head>
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <link href="style.css" rel="stylesheet">
    <title>Critical Path</title>
  </head>
  <body>
    <p>Hello <span>web performance</span> students!</p>
    <div></div>
  </body>
</html>
```



cc 3.0 attribution

<https://developers.google.com/web/fundamentals/performance/critical-rendering-path/constructing-the-object-model>

レンダリング の流れ

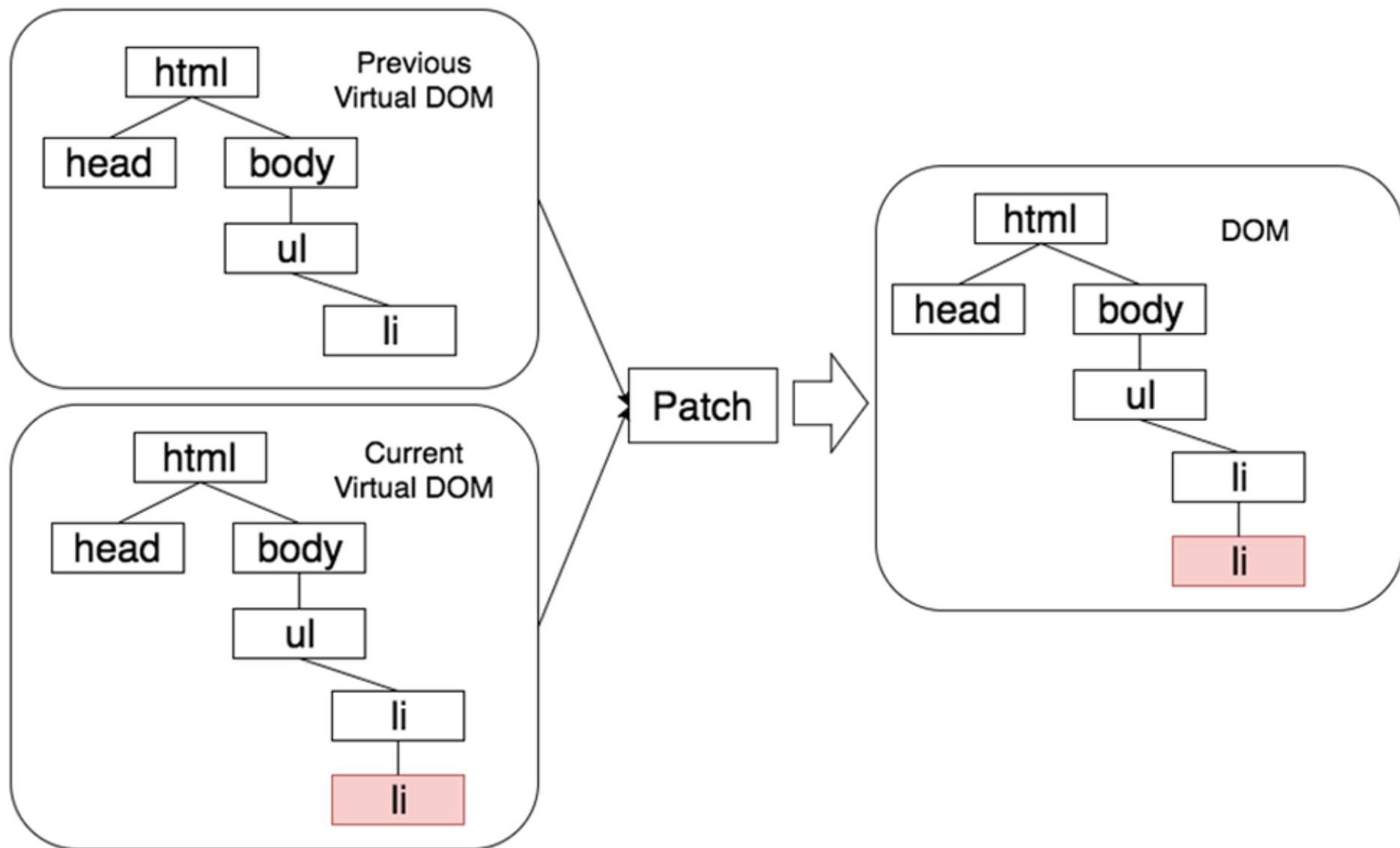


戻ります

Virtual DOM (仮想DOM)

Version 2(2016年10月リリース)から仮想DOMを採用

- 仮想的なDOMを作成
- DOMの変更を差分で行う
- 仮想DOMについてコードを書く必要はない



Vue.jsを使おう

Vue.jsの初め方

- CDNを読み込む
- vue-cliを使う
- ビルドシステムを利用して自分で作る

JSFiddle

ブラウザで直接HTML/CSS/JSを実行・確認できる
(goo.glはショートURL)

<https://jsfiddle.net/sugumura/e54kpf5h/>
<https://goo.gl/cSL3oF>

Meta

Vue.js

Description

to make the fiddle

Resources

Requests

Credits and Links

[Fiddle Roadmap](#)

and vote for features

Fiddle?

running by
using JSFiddle in
blocker.

erving quality, tech-
ads only.

ou!

```
1 <!-- ここにHTMLを書きます -->
2 <!-- Vue.jsの読み込み -->
3 <script src="https://unpkg.com/vue/dist/vue.js">
  </script>
4
5 <!-- ここから -->
6 <div id="app">
7   <h1>こんにちは</h1>
8 </div>
```

HTML

```
1 // ここにJavascriptを書きます
2
3
```

JAVASCRIPT

こんにちは

Hello (html)

テンプレートの書き方 `{{}}` でプロパティ名を囲む

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

Hello (js)

Vueインスタンスを生成

```
new Vue({  
  el: '#app',  
  data: {  
    message: 'Hello Vue!'  
  }  
});
```

実行

Hello Vue!が表示される

- elオプションにDOMを指定
- dataオプションにリアクティブにするオブジェクト

リアクティブ (html)

dataで指定したオブジェクトはVueがプロキシする

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!',
    count: 0
  }
})

// 1秒毎にカウントアップ
setInterval(function () {
  app.count = app.count + 1;
}, 1000)
```

リアクティブ (js)

オブジェクトの変更を検知して表示を更新

```
<div id="app">  
  {{ message + count }}  
</div>
```


ディレクティブ

- v-から始まる特別な属性
- ディレクティブを使うことで様々な機能が利用可能

v-bind

属性を反映する

```
<div id="app2">  
  <a v-bind:href="url" target="_blank">Click me</a>  
</div>
```

```
new Vue({  
  el: '#app2',  
  data: {  
    url: 'https://www.google.com'  
  }  
})
```

v-for (html)

配列の繰り返しを扱う

```
<div id="app3">
  <ol>
    <li v-for="item in list">
      {{ item.label }}
    </li>
  </ol>
</div>
```

v-for (js)

```
new Vue({  
  el: '#app3',  
  data: {  
    list: [  
      { label: 'Vue.js' },  
      { label: 'Angular.js' },  
      { label: 'React' }  
    ]  
  }  
});
```

v-model (html)

入力を反映する

```
<div id="app4">  
  <p>{{ message }}</p>  
  <input v-model="message">  
</div>
```

v-model (js)

```
new Vue({  
  el: '#app4',  
  data: {  
    message: 'Hello Vue!'  
  }  
})
```

ディレクティブを

ドキュメントで眺めよう

<https://jp.vuejs.org/v2/api/#ディレクティブ>

多い？少ない？

作ってみると便利さを実感できます
ちょっとした機能を作ってみましょう

Yes Or No?

あなたが決断する手助けをしてくれます
ページを開いてみましょう

<https://yesno.wtf>

ポイント

<https://yesno.wtf/api>

- APIが提供されている
- GETするだけでJSONを返却
- Cross-Origin Resource Sharing OK(重要)

YesNoリストを作るう

- 質問を入力するフォームを設置
- YesNo APIで返却された答えのリストを作成
- 回答は削除可能
- 回答件数を表示

ひな形

<https://jsfiddle.net/sugumura/6yo6x0L5/>

<https://goo.gl/2aDMWD>

イベントを受け取るう

Inputを紐付ける

```
<input type="text" v-model="question">
```

```
new Vue({  
  el: '#app',  
  data: {  
    question: ''  
  }  
});
```

送信ボタンでイベント

```
<button v-on:click="add()">送信</button>
```

```
data: {  
  question: ''  
},  
methods: { // メソッドはmethodsに記述  
  add: function () {  
    console.log(this.question);  
  }  
}
```

APIリクエスト

- axiosというライブラリを使用
- Promise形式
- 詳細説明は省略させていただきます

APIリクエスト

```
methods: { // メソッドはmethodsに記述
  add: function () {
    axios.get('https://yesno.wtf/api')
      .then(function (response) {
        console.log(response.data);
      });
  }
}
```

APIデータ

```
{  
  "answer": "no",    // または "yes"  
  "forced": false,  
  "image": "https://yesno.wtf/....gif"  
}
```

データを表示しよう

データを表示(html)

src="{{item.image}}"はエラー

```
<tr>
  <td>{{item.date}}</td><!-- 日時 -->
  <td>{{item.question}}</td><!-- 質問 -->
  <td>{{item.answer}}</td><!-- 回答 -->
  <td></td>
  <td><button>削除</button></td>
</tr>
```

データを表示(js)

```
data: {  
  item: {},  
  question: ''  
},  
methods: { // メソッドはmethodsに記述  
  add: function () {  
    var vm = this; // this参照用  
    axios.get('https://yesno.wtf/api')  
      .then(function (response) {  
        vm.item = response.data;  
        vm.item.date = new Date();  
        vm.item.question = vm.question;  
      });  
  }  
}
```

フィルターを作ろう

フォーマットフィルター

Vue.filterで任意のフィルターを作成
Vueを生成前(new)に追加

```
// YYYY-mm-dd HH:mm:ss
Vue.filter('date-filter', function (val) {
  if (!val) return;
  return [val.getFullYear(), (val.getMonth()+1), val.getDate()]
    .join('-') + ' ' +
    [val.getHours(), val.getMinutes(), val.getSeconds()].join(':')
})
```

Dateをフォーマット表示

```
<tr>
  <td>{{item.date | date-filter}}</td><!-- 日時 -->
  <td>{{item.question}}</td><!-- 質問 -->
  <td>{{item.answer}}</td><!-- 回答 -->
  ...

```


リストを表示しよう

リスト表示(html)

```
<tr v-for="item in items">
  <td>{{item.date | date-filter}}</td><!-- 日時 -->
  <td>{{item.question}}</td><!-- 質問 -->
  <td>{{item.answer}}</td><!-- 回答 -->
  ...
```

リスト表示1(js)

データ保存を配列にする

```
data: {  
  items: [],      // itemからitemsに変更  
  question: ''  
},
```

リスト表示2(js)

```
add: function() {  
  var vm = this; // this参照用  
  axios.get('https://yesno.wtf/api')  
    .then(function(response) {  
      var item = { // オブジェクトを作成  
        date: new Date(),  
        question: vm.question,  
        answer: response.data.answer,  
        image: response.data.image  
      };  
      vm.items.push(item); // 配列に追加  
    });  
}
```

削除

削除(html)

v-forを引数使いインデックスを取得する

```
<tr v-for="(item, index) in items">  
  ...  
  <td><button v-on:click="remove(index)">削除</button></td>  
</tr>
```

削除(js)

```
methods: {  
  add: {  
    ...  
  },  
  remove: function(index) {  
    this.items.splice(index, 1);  
  }  
}
```

件数を表示する

件数を表示(html)

```
<!-- 合計行 -->  
<tr>  
  <td colspan="5">合計: {{total}}件</td>  
</tr>
```

件数を表示(js)

件数は配列の長さ
算出プロパティを利用

```
methods: {  
  ...  
},  
computed: {  
  total: function () {  
    return this.items.length;  
  }  
}
```

算出プロパティ

- プロパティライクに扱える
- ()の呼び出し不要
- 依存関係が更新されなければキャッシュする

スライドはここまで

時間がある場合、機能追加します

これから

日本語ドキュメントを読みましょう

<https://jp.vuejs.org>

資料

- Vue.js入門 —最速で作るシンプルなWebアプリケーション
 - Webアプリを作る前に
- Vue.js 2.0 Server Side Rendering
 - レンダリング参考になります

バージョンについて

- v2は2016年10月
- ネットではv1情報が多いので日付に注意
- 最新は英語ドキュメントを確認

おつかれさまでした