ما هي Vue.js

هذا المقال هو جزء من سلسلة (جافا سكربت)، يمكنكم الإطلاع على أجزائها الأخري لاستكمال الفهم عبر الروابط التالية: جافا سكربت، مكتبة Node.JS، ما هو Angular JS ، ما هي الـjQuery؟.

vue.js هي بيئة عمل لبناء واجهة المستخدم، وخلافاً لبيئات العمل الأخرى فإن **vue** صممت من الألف إلى الياء ليكون بناؤها تدريجياً. وتُركز النواة الرئيسية للمكتبة على طبقة العرض **view** فقط، وهي شديدة السهولة للجلب، والاستخدام مع المكتبات الأخرى، أو في المشاريع المسبق انشاؤها.

ومن جهة أخرى، إنّ **vue** قادرة بشكل كامل على بناء مواقع الصفحة الواحدة عندما تستخدم مع أدوات مناسبة وتضاف لها المكتبات

اللازمة. وإذا كنت محترفاً في تطوير واجهات مواقع الويب وتريد مقارنة **vue** مع المكتبات الأخرى فيمكنك قراءة مقارنة مع المواقع الأخرى من هنا

البداية

لكي تبدأ في تعلم**vue** ، يتوقع منك معرفة متوسطة في كل من **HTML/CSS**، و **Javascript**. واذا كنت مستجداً بشكل كامل على تطوير واجهات مواقع الويب فمن الأفضل عدم القفز مباشرة لتعلم بيئات العمل المخصصة لذلك.
إن الطريقة الأسهل لتجريب **vue** هي عن طريق موقع تعلم الجافاسكربت JSFiddle Hello World example، أو يمكنك بكل بساطة انشاء ملف **html** عادي وتضمين مكتبة **vue** فيه عن طريق السطر التالي:

```
<script src="https://unpkg.com/vue/dist/vue.js"></script>
```

الإظهار (Declarative Rendering)

إن نواة **Vue.js** تسمح لك بإظهار البيانات باستخدام مفهوم القوالب **template** مثال:
في قسم الـ **HTML** لدينا الأكواد التالية:

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

وفي قسم الجافا سكربت نهاية الصفحة أضفنا الكود التالي لا تنسى وضعها بداخل وسم الجافا سكربت

```
<script>here</script>
```

أولا نقوم بانشاء استدعاء لمكتبة vue ثم نحدد الـ id للعنصر الذي سيعمل vue بداخلة لكي لا تتأثر باقي عناصر الصفحة وأخيرا نقوم بتمرير البيانات للصفحة لكي يعرضها لنا في المتصفح

```
var app = new Vue({
        el: '#app',
            data: {
'       message: 'Hello Vue
                  {
                     ({
```

والنتيجة تكون:

Hello Vue

عبر هذا المثال نكون قد أنشأنا أول تطبيق لنا باستخدام vue، قد يبدو هذا التطبيق بسيطاً وغير ذي أهمية ولكن في الحقيقة فإن vue قد قامت بالكثير من الأمور في الخفاء، حيث أصبح ظهور البيانات في الصفحة ديناميكياً وليس مباشراً، ولفهم الأمور أكثر دعنا نجرب مثالاً

ثانياً:

في صفحة **HTML** التي أنشأناها مسبقا قم بإضافة الكود التالي:

```
<<div id="app-2">
    <span v-bind:title="message">
        ضع الماوس فوق هذا النص لعدة ثواني لترى جمال العرض الديناميكي
    </span>/>
</<div
```

وفي أسفل الصفحة بداخل الوسم **script**

```
var app2 = new Vue({
    el: '#app-2',
    data: {
        message : 'لقد قمت بفتح هذه الصفحة بتاريخ : ' + new Date()
    }
})
```

والنتيجة ستكون ظهور هذا السطر:

ضع الماوس فوق هذا النص لعدة ثوان لترى جمال العرض الديناميكي

وعندما تضع الماوس فوق هذا السطر لعدة ثواني ستظهر لك الرسالة مع التوقيت الذي وضعت الماوس فيه.
وقد استخدمنا في المثال السابق ترميزاً جديداً على لغة **HTML** وهو **v-bind** حيث يرمز حرف v- الى مكتبة **vue** ومن ثم اسم الخاصية المراد استخدامها والتي هي الـ **bind** ومعناها الربط ولكي نحدد ما الذي سنربطه قمنا بإضافة خاصية الـ HTML المعروفة **title** وهي النص الذي يظهر عند وضع الماوس فوق النص ثم حددنا لهذه الـ **title** قيمة ديناميكية هي **message**

في قسم الجافا سكربت قمنا أولاً بتحديد الوسم الذي سيعمل **vue** بداخله عبر السطر

```
,'el: '#app-2
```

ومن ثم حددنا قيمة الـ **message** التي أضفناها مسبقاً بشكل ديناميكي

الشروط والحلقات (Conditionals & Loops)

الشروط: من السهل جداً تحديد شرط لظهور العناصر عن طريق **vue** ولفهم هذه الخاصية لنبدأ بالمثال التالي:

في قسم الـ **HTML** اضف الكود التالي:

```
<<div id="app-3">
    <p v-if="seen">الان يمكنك ان ترى النص</p>
</<div
```

وفي قسم الجافا سكربت أضف الكود التالي

```
var app3 = new Vue({
        el: '#app-3',
            data: {
        seen: true
            {
            ({
```

والنتيجة سيظهر لك السطر التالي:

الآن يمكنك أن ترى النص

ولكنك لو غيرت قيمة السطر التالي **seen:true** إلى **seen:false** سيختفي النص
وكما وضحنا سابقاً فقد استخدمنا أولاً الترميز الخاص بـ **vue** عبر الرمز **v-** ثم أضفنا الشرط **if** وحددنا له قيمة ديناميكية اسمها **seen**. والآن في قسم الجافاسكربت عندما نحدد قيمة **seen** على أنها **true** سيظهر النص، أما اذا كانت **false** فلن يظهر النص.

الحلقات: سنستخدم في هذا المثال الرمز **v-for** لعرض المصفوفات **arrays** ضمن قسم الـ **HTML** بدون أي عناء.
في قسم الـ **HTML** أضف الكود التالي:

```
div id="app-4">>
<ol>
<li v-for="todo in todos">
{{ todo.text }}
</li>
</ol>
<</div
```

وفي قسم الجافاسكربت اسفل الصفحة اضف الكود التالي:

```
var app4 = new Vue({
        el: '#app-4',
            data: {
            todos: [
    { text: 'Learn JavaScript' },
        { text: 'Learn Vue' },
{ '        { text: 'Build something awesome
            [
            {
            ({
```

والنتيجة ستظهر:

1. Learn JavaScript

2. Learn Vue
3. Build something awesome

في المثال السابق قمنا باستخدام الترميز **v-for** وحددنا بأنه ما دامت هناك بيانات بداخل المصفوفة **todos** فإنه يتم عرضها، واستخدمنا للإشارة لهذه البيانات كلمة **todo** ويمكنك استبدالها بأي كلمة مناسبة مثل **data** أو **value**

ومن ثم في قسم الجافاسكربت قمنا بتحديد المصفوفة المسماة **todos** وأضفنا بيانات بداخلها، لتظهر بشكل ديناميكي. وفي هذه الحالة يمكنك تخيل بيانات مرسلة من قبل السيرفر **Server** مثل استدعاءات من قواعد البيانات أو مثل طلبات الأجاكس والتي يقوم بها أيضاً **vue** ولكننا في هذا السلسلة قمنا بتقديم تعريف أولي عن المكتبة أو بيئة العمل، وتركنا لكم مهمة التعمق في المكتبة التي تعجبكم عندما تختارونها.

- يمكنكم مراجعة المصدر الرسمي لتعليم بيئة العمل هنا**vue.js**للمزيد من البيانات حول مكتبة
- أو إذا كنت مهتما بدروس فيديو تعلمك المكتبة يمكنكم الاستعانة بهذه الدورة المجانية باللغة الانكليزية:
  - دورة حول VUE
  - دورة عن تحديثات VUE

- التاريخ: 2017-03-22
- التصنيف: تكنولوجيا

#vue# #Vue.js #لغة Javascript# HTML #جافا سكربت

المصادر

- vuejs

المساهمون

- ترجمة
  - عصام عباس
- مُراجعة
  - ريم المير أبو عجيب
- تحرير
  - أنس عبود

- تصميم
  - Tareq Halaby
- نشر
  - Tareq Halaby
  - مي الشاهد