إليكم ستة أسباب تجعل من تعلم مطوري الويب لـ JavaScript ES6 أمراً ضرورياً

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

تعتبر **ECMAScript 6** اواختصاراً (**ES6**) والتي تُعرف أيضاً باسم **ECMAScript 2015**، الإصدار الأحدث من **ECMAScript** القياسية. وتعتبر **ES6** التحديث الأبرز، ويمكن القول أنه الأول للغة، منذ توحيد مقاييس **ES5** عام 2009. يتم تطبيق هذه الميزات في محركات الجافا سكربت الرئيسة الآن.

إذا كنت مُطوراً للويب، فلا بد أن تكون قد سمعت قليلاً عن **ECMAScript 6**. للوهلة الأولى قد يبدو الأمر مربكاً لكن أهمية (**ES6**)

تكمن في أنها النسخة الجديدة من لغة **JavaScript**، التي توفر ميزات جديدة ستتمكن من استخدامها على المدى الطويل.

تدعم متصفحات الحواسيب المكتبية **ES6** بشكل جيد، حيث تصل نسبة الدعم في متصفح **Chrome** إلى 90%، وإلى 80% في **Edge** ، وإلى 54% في **Safari**. أما أجهزة الهاتف المحمول بنظام أندرويد 5.1 فتدعم **ES6** بنسبة 31% فقط، وتصل نسبة الدعم إلى 54% في الأجهزة بنظام **IOS**.

لذلك يمكن استخدام ميزات **ES6** الآن من خلال استخدام معالج تمهيدي **Pre-Processor** مثل **Babel** من أجل الترجمة التقاطعية **cross-compile** للجافا سكربت إلى كود **ES5** المتوافق مع متصفحات أقدم، مما يعني أنه لا داعي لتأجيل تعلمها.

وفيما يلي أهم مميزات ES6

- المتغيرات والثوابت **Variables and constants**

أصبحت الجافا سكربت أخيراً تدعم الثوابت، وبذلك تستطيع أن تعيّن عدداً في الكود البرمجي الخاص بك بحيث يبقى ثابتاً طول زمن البرنامج ـإذا حاولت لاحقاً أن تُسند قيمة أخرى لذلك الثابت فسينتج لديك خطأ.

هذا أمر مفيد جداً عندما يكون لديك عدد ما في البرنامج لا يمكن أن يتغير مع الوقت وتطبيقك يعتمد على كونه ثابتاً. وهناك ميزة أخرى جديدة تدعى "**Let**" توفر طريقة جديدة لتعيين المتغيرات في الجافا سكربت. ربما قد اعتدت على تعيينها باستخدام "**var**" ولكن الفرق دقيق: المجال **Scope**.

عند تعيين متغير باستخدام "**Let**" فإن مجاله ينحصر في الوحدة **block** الموجود فيها، وليس كمتغير عام. يمكنك القراءة أكثر عن "**let**" والمتغيرات هنا

- التوابع السهمية **Arrow functions**

تتمثل ميزة إضافية جديدة لـ **ES6** في السماح بالتصريح عن تابع باستخدام سهم بسيط كالتالي **=>** ، بدلاً من التعبير التقليدي عن التوابع. قد يبدو الأمر محيّراً في البداية، ولكن لذلك فائدة من ناحية الاختصار في قواعد البرمجة الكبيرة.

تتصرف التوابع السهمية بشكل مختلف عن التوابع الاعتيادية عند الاستدعاء فهي تأخذ "**this**" و "**arguments**" من السياق المحيط بها مباشرة، بدلاً من البحث عنها بداخلها. إن ميزات التوابع السهمية مشابهة لميزات اللغات **C#**، و **Java8**، و **CoffeeScript**.

ولدينا الكود البرمجي التالي كمثال

```
// Expression bodies

var odds = evens.map(v => v + 1);

var nums = evens.map((v, i) => v + i);
```

```
var pairs = evens.map(v => ({even: v, odd: v + 1}));


// Statement bodies
nums.forEach(v => {
  if (v % 5 === 0)
    fives.push(v);
});


// Lexical this
var bob = {
  _name: "Bob",
  _friends: [],
  printFriends() {
    this._friends.forEach(f =>
      ;((console.log(this._name + " knows " + f
    {
  {
```

- **الوحدات البرمجية Modules**

الوحدات البرمجية **Modules** هو مفهوم آخر مثير للاهتمام في **ES6** يستخدمه مطوّرو الجافا سكربت منذ أعوام. تسمح لك بعض المشاريع مثل Webpack باستغلال البنية البرمجية **Modularization**. ولكن، **ES6** وللمرة الأولى توفر البنية البرمجية كجزء من بيئة جافا سكربت المحلية في المتصفح.

إضافة البنية البرمجية سبب هام جداً للبدء في تعلم **ES6** الآن، حيث أنها ستغير من طريقة عملك كلياً للأفضل إن لم تقم باستخدامها من قبل. تُخزَّن وحدات **ES6** في ملفات منفصلة ويمكن أن يكون هناك وحدة برمجية واحدة في كل ملف، والذي لا زال يمتلك الامتداد **js**.

```
lib/math.js //
export function sum(x, y) {
return x + y;
}
export var pi = 3.141593;

// app.js
import * as math from "lib/math";
alert("2π = " + math.sum(math.pi, math.pi));

// otherApp.js
import {sum, pi} from "lib/math";
alert("2π = " + sum(pi, pi));
Some additional features include export default and export *:
// lib/mathplusplus.js
export * from "lib/math";
export var e = 2.71828182846;
export default function(x) {
return Math.log(x);
}
// app.js
Import In, {pi, e} from "lib/mathplusplus";
;(Alert ("2π = " + In(e)*pi*2
```

أهم ما يجب معرفته أن الوحدات البرمجية تساعدك على صقل الكود البرمجي الخاص بك ليصبح كتلة يسهل التعامل مع محتواها من التعليمات البرمجية.

- **معاملات الامتداد Spread**

إن معاملات الامتداد معاملات بسيطة جداً ولكنها ملائمة، حيث تضاف من أجل توسيع مصفوفة حالما يتم استدعائها. ببساطة تستخدم '...' لتوسيع المصفوفة وبدلاً من الحاجة لاستخراج قيم المصفوفة ستجد الأمر جاهزاً بالنسبة لك.

إنها ليست تغيير جذري، ولكنها طريقة للتوقف عن استخدام الحيل الشائعة، وفقاً لِـ مستندات المطورين الخاصة بمتصفح موزيلا.

- **حلقات التكرار For-of**

حدثت أخيراً تحسينات على الحلقات، الآن يمكنك التكرار عبر المتغيرات بمجهود أقل. كانت **For-in** موجودة في السابق ولوقت طويل، ولكن تمت إضافة **For-of** حديثاً لِـ **ES6**. للمزيد انقر هنا.

- **الصفوف Classes**

في الواقع إنه أمر لا يصدق ولكن جافا سكربت أصبحت بالفعل تدعم الصفوف مع إطلاق **ES6**، رغم أن ذلك كان مثيراً للجدل.

للمزيد انقر هنا.

تشجع صفوف **ES6** الأنماط كائنية التوجه القائمة على النماذج **prototype-based object oriented pattern**، وكذلك تدعم طرق الوراثة **inheritance methods**، والطرق البانية **constructors methods**، والطرق الثابتة **static methodss** والكثير غيرها.

تمتلك الصفوف هنا نمط وحيد للتصريح عنها مما يجعلها سهلة الاستخدام.

وإليكم المثال التالي

```
class SkinnedMesh extends THREE.Mesh {
    constructor(geometry, materials) {
        super(geometry, materials);

this.idMatrix = SkinnedMesh.defaultMatrix();
                this.bones = [];
            this.boneMatrices = [];
                        //...
                        }
                update(camera) {
                        //...
                 super.update();
                        }
                get boneCount() {
            return this.bones.length;
                        }
            set matrixType(matrixType) {
    this.idMatrix = SkinnedMesh[matrixType]();
                        }
                    static defaultMatrix() {
        ;()     return new THREE.Matrix4
                                {
                                  {
```

- التاريخ: 2017-03-29
- التصنيف: تكنولوجيا

#Javascript #جافا سكربت #ES6 #ECMAScript 6

المصطلحات

- **الأيونات أو الشوارد (Ions)**: الأيون أو الشاردة هو عبارة عن ذرة تم تجريدها من الكترون أو أكثر، مما يُعطيها شحنة موجبة.وتسمى أيوناً موجباً، وقد تكون ذرة اكتسبت الكتروناً أو أكثر فتصبح ذات شحنة سالبة وتسمى أيوناً سالباً

المصادر

- thenextweb
- github

المساهمون

- ترجمة
  - ريم المير أبو عجيب
- مُراجعة
  - دانا أسعد
- تحرير
  - أنس عبود
- تصميم
  - محمد نور حماده
- نشر
  - مي الشاهد