

عنوان:	RequireJS
نویسنده:	مسعود پاکدل
تاریخ:	۸:۳۰ ۱۳۹۲/۰۶/۳۰
آدرس:	<a href="http://www.dotnettips.info">www.dotnettips.info</a>
برچسب‌ها:	requirejs, module

در طراحی و توسعه پروژه‌های تحت وب در مقیاس بزرگ برای اینکه مدیریت پروژه راحت‌تر شود کدهای مورد نظر را در چند ماژول قرار می‌دهند در نتیجه کدهای پروژه در بلاک‌های کوچک‌تر قرار خواهند داشت. نوشتن پروژه به صورت ماژولار قابلیت استفاده مجدد از کدهای برنامه را افزایش می‌دهد، علاوه بر آن مدیریت پروژه در فاز نگهداری آسان‌تر خواهد شد؛ از طرفی دیگر وابستگی بین ماژول‌ها و تامین آن‌ها، همواره مهم‌ترین مفهوم برای توسعه دهندگان پروژه‌های وب است. RequireJS یکی از فریم ورک‌های محبوب برای مدیریت وابستگی‌های بین ماژول‌ها است و کاربرد اصلی آن راحت سازی مفهوم modularity در اینگونه پروژه هاست.

پروژه‌های بزرگ عموماً دارای یک یا چند فایل جاوااسکریپ هستند که برای استفاده از آن‌ها در صفحات از تگ script استفاده می‌شود. اگر این فایل‌ها دارای وابستگی به هم باشند، ترتیب فراخوانی این فایل در تگ script مهم است. برای مثال: یک پروژه دارای فایل‌های زیر خواهد بود:

purchase.js

```
function purchaseProduct(){
    console.log("Function : purchaseProduct");
    var credits = getCredits();
    if(credits > 0){
        reserveProduct();
        return true;
    }
    return false;
}
```

product.js

```
function reserveProduct(){
    console.log("Function : reserveProduct");
    return true;
}
```

credits.js

```
function getCredits(){
    console.log("Function : getCredits");
    var credits = "100";
    return credits;
}
```

همان طور که در فایل‌های بالا مشاهده می‌کنید در فایل purchase.js از دو فایل دیگر استفاده شده است. در فایل main.js پروژه کد زیر را برای استفاده از فایل‌های بالا می‌نویسیم:

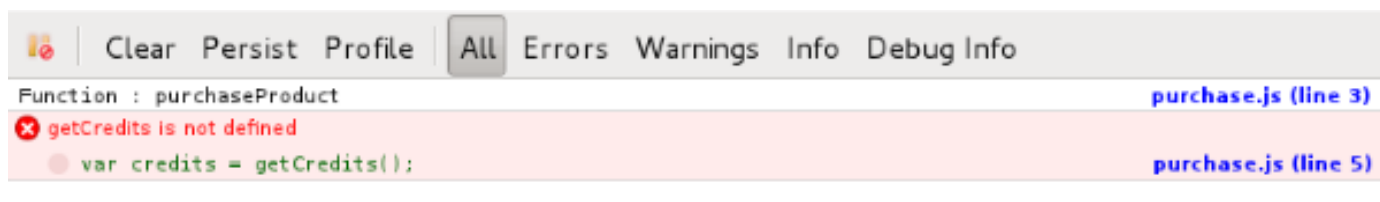
main.js

```
var result = purchaseProduct();
```

فرض کنید در فایل HTML تگ‌های script ما به صورت زیر باشد:

```
<script src="products.js"></script>
<script src="purchase.js"></script>
<script src="main.js"></script>
<script src="credits.js"></script>
```

از آنجا که لود فایل purchase.js وابستگی مستقیم به لود فایل credits.js دارد و از طرفی دیگر به دلیل این که فایل credit.js بعد از تمام فایل‌ها لود می‌شود در نتیجه برنامه اجرا نخواهد شد و با خطای زیر روبرو می‌شویم:



فقط کافایت تصور کنید که تعداد و حجم کدهای این فایل‌ها در یک پروژه زیاد باشد یا حتی این فایل‌ها توسط یک برنامه نویس دیگر تهیه و تدوین شده باشد؛ در نتیجه به خاطر سپردن این وابستگی‌ها به طور قطع کار سخت خواهد بود و در خیلی موارد طاقت فرسا.

RequireJS چگونه برای حل این مشکل به ما کمک می‌کند؟

با استفاده از فریم ورک RequireJS کدهای ما به ماژول‌های کوچک‌تر شکسته می‌شود و وابستگی ماژول‌ها در تنظیمات لود فایل ثبت می‌شود. در ضمن این فریم ورک با مرورگرها جدید و محبوب کاملاً سازگار است. برای شروع فایل‌های مورد نیاز را از [اینجا](#) دانلود نمایید. البته می‌توانید از nuget هم استفاده کنید:

```
PM> Install-Package RequireJS
```

در مثال بالا فایل‌های جاوااسکریپت به صورت زیر تغییر خواهد کرد:

purchase.js

```
define(["credits","products"], function(credits,products) {
    console.log("Function : purchaseProduct");
    return {
        purchaseProduct: function() {
            var credit = credits.getCredits();
            if(credit > 0){
                products.reserveProduct();
                return true;
            }
            return false;
        }
    }
});
```

همان طور که مشاهده می‌کنید در فایل بالا از دستور define برای تعریف ماژول استفاده شده است و نام دو ماژول products و credits را به صورت پارامتر برای مشخص کردن وابستگی ماژول‌ها تعریف کردیم.

products.js

```
define(function(products) {
    return {
        reserveProduct: function() {
            console.log("Function : reserveProduct");
            return true;
        }
    }
});
```

credits.js

```
define(function() {
```

```
console.log("Function : getCredits");
return {
  getCredits: function() {
    var credits = "100";
    return credits;
  }
};
```

به دلیل این که فایل‌های بالا وابستگی به ماژول‌های دیگر ندارند در نتیجه دستور define فقط شامل تعریف تابع است. کفایت در فایل main.js، برای استفاده از فایل purchase.js از دستور require استفاده کنید.

```
require(['purchase'], function( purchase ) {
  var result = purchase.purchaseProduct();
});
```

مهم‌ترین مزیتی که این روش دارد این است که دیگر نیازی به نوشتن تگ‌های Script (آن هم به ترتیب درست) برای فراخوانی فایل‌های جاوااسکریپتی نخواهد بود؛ از طرفی دیگر وابستگی بین این فایل‌ها در هنگام تعریف ماژول مشخص خواهد شد. از آن به بعد وظیفه تامین فایل‌های مورد نیاز برای لود ماژول بر عهده RequireJs است.

تفاوت بین دستور require و define

دستور define صرفاً برای تعیین وابستگی ماژول استفاده می‌شود در حالی که دستور require برای فراخوانی و لود ماژول کاربرد دارد و به نوعی به عنوان نقطه شروع اجرای برنامه خواهد بود. در پست بعدی به پیاده سازی مثال بالا به کمک RequireJs در قالب یک پروژه Asp.Net MVC خواهیم پرداخت.