عنوان: **شروع کار با Dart - قسمت 1**
نویسنده: میثم خوشبخت
تاریخ: ۱۳۹۳/۰۲/۰۲ ۱۴:۵۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Web Design, Dart

Dart کتابخانه ای است که توسط شرکت گوگل ارائه شده است و گفته می‌شود، قرار است جایگزین جاوا اسکریپت گردد و از آدرس https://www.dartlang.org قابل دسترسی می‌باشد. این کتابخانه، دارای انعطاف پذیری فوق العاده بالایی است و کد نویسی Java Script را راحت‌تر می‌کند. در حال حاضر هیچ مرورگری به غیر از Chromium از این تکنولوژی پشتیبانی نمی‌کند و جهت تسهیل در کدنویسی، باید از ویرایشگر Dart Editor استفاده کنید. این ویرایشگر کدهای نوشته شده را به دو صورت Native و JavaScript Compiled در اختیار مرورگر قرار می‌دهد. در ادامه با نحوه‌ی کار و راه اندازی Dart آشنا خواهید شد.

ابتدا Dart و ویرایشگر مربوط به آن را توسط لینک‌های زیر دانلود کنید:

دانلود
نسخه 64 بیتی دارت + ویرایشگر

دانلود
نسخه 32 بیتی دارت + ویرایشگر

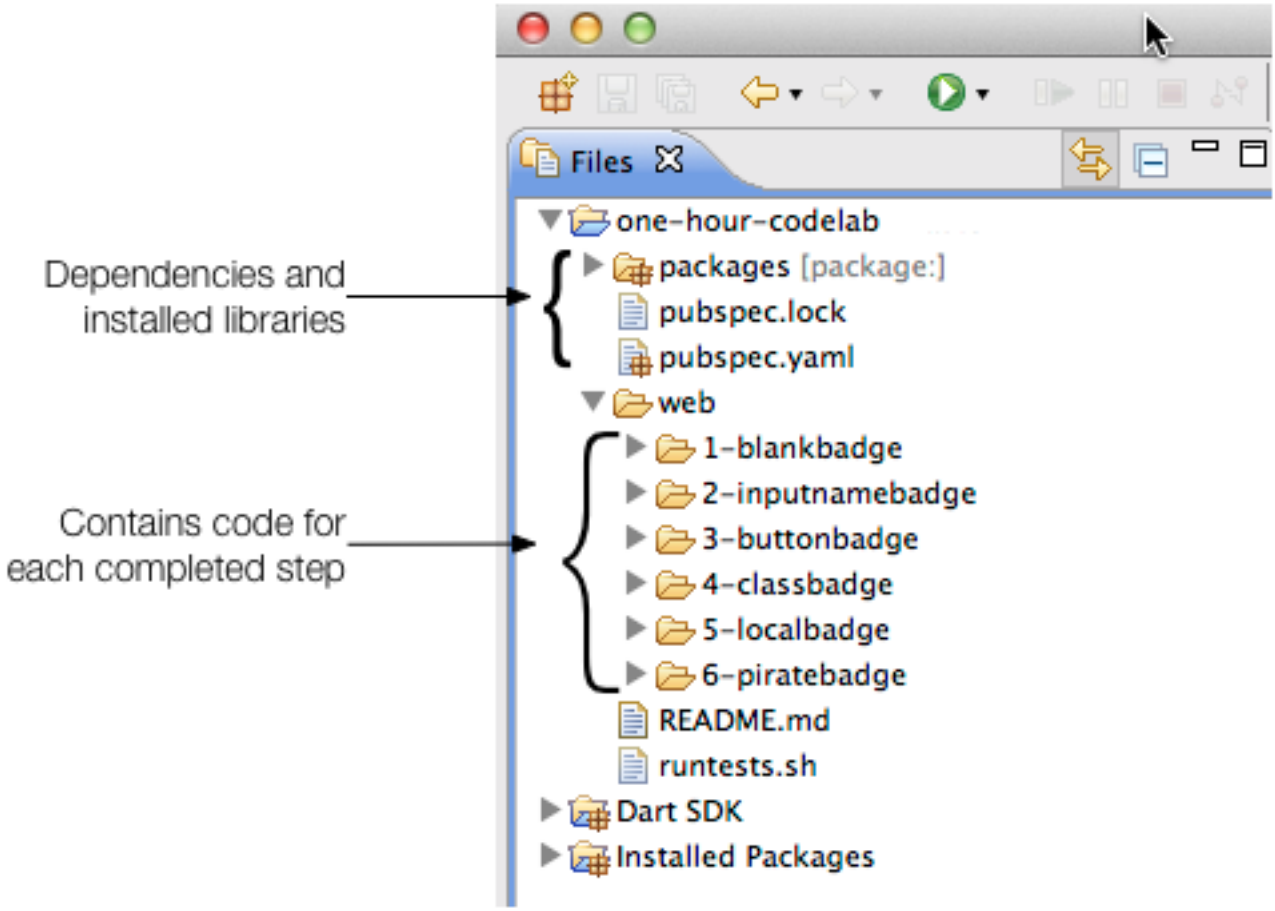
بعد از اینکه فایلهای فوق را از حالت فشرده خارج کردید، پوشه ای با نام dart ایجاد می‌نماید. وارد پوشه dart شده و DartEditor را اجرا کنید.

توجه: جهت اجرای dart به JDK 6.0 یا بالاتر نیاز دارید

در مرحله بعد نمونه کدهای Dart را از لینک زیر دانلود نمایید و از حالت فشرده خارج کنید. پوشه ای با نام one-hour-codelab ایجاد می‌گردد.

دانلود
نمونه کدهای دارت

از منوی File > Open Existing Folder... پوشه one-hour-codelab را باز کنید .



توضیحات

- پوشه packages و همچنین فایلهای pubspec.yaml و pubspec.lock شامل پیش نیازها و Package هایی هستند که جهت اجرای برنامه‌های تحت Dart مورد نیاز هستند. Dart Editor این نیازمندی‌ها را به صورت خودکار نصب و تنظیم می‌کند.

توجه: اگر پوشه Packages را مشاهده نکردید و یا در سمت چپ فایلها علامت X قرمز رنگ وجود داشت، بدین معنی است که package ها به درستی نصب نشده اند. برای این منظور بر روی pubspec.yaml کلیک راست نموده و گزینه Get Pub را انتخاب کنید. توجه داشته باید که بدلیل تحریم ایران توسط گوگل باید از ابزارهای عبور از تحریم استفاده کنید.

- 6 پوشه را نیز در تصویر فوق مشاهده می‌کنید که نمونه کد piratebadge را بصورت مرحله به مرحله انجام داده و به پایان می‌رساند.

- Dart SDK شامل سورس کد مربوط به تمامی توابع، متغیرها و کلاس هایی است که توسط کیت توسعه نرم افزاری Dart ارائه شده است.

- Installed Packages شامل سورس کد مربوط به تمامی توابع، متغیرها و کلاس‌های کتابخانه‌های اضافه‌تری است که Application به آنها وابسته است.

گام اول: اجرای یک برنامه کوچک

در این مرحله سورس کدهای آماده را مشاهده می‌کنید و با ساختار کدهای Dart و HTML آشنا می‌شوید و برنامه کوچکی را اجرا

می‌نمایید.

در Dart Editor پوشه blankbadge-1 را باز کنید و فایلهای piratebadge.html و piratebadge.dart را مشاهده نمایید.

کد موجود در فایل piratebadge.html

```
<html>
  <head>
    <meta charset="utf-8">
    <title>Pirate badge</title>
    <meta name="viewport"
          content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="piratebadge.css">
  </head>
  <body>
    <h1>Pirate badge</h1>

    <div>
      TO DO: Put the UI widgets here.
    </div>
    <div>
      <div>
        Arrr! Me name is
      </div>
      <div>
        <span id="badgeName"> </span>
      </div>
    </div>

    <script type="application/dart" src="piratebadge.dart"></script>
    <script src="packages/browser/dart.js"></script>
  </body>
</html>
```

توضیحات

- در کد HTML ، اولین تگ <script> ، فایل piratebadge.dart را جهت پیاده سازی دستورات dart به صفحه ضمیمه می‌نماید

- Dart VM (Dart Virtual Machine) کدهای Dart را بصورت Native یا بومی ماشین اجرا می‌کند. Dart VM کدهای خود را در Dartium که یک ویرایش ویژه از مرورگر Chromium می‌باشد اجرا می‌کند که می‌تواند برنامه‌های تحت Dart را بصورت Native اجرا کند.

- فایل packages/browser/dart.js پشتیبانی مرورگر از کد Native دارت را بررسی می‌کند و در صورت پشتیبانی، Dart VM را راه اندازی می‌کند و در غیر این صورت JavaScript کامپایل شده را بارگزاری می‌نماید.

کد موجود در piratebadge.dart

```
void main() {
  // Your app starts here.
}
```

- این فایل شامل تابع main می‌باشد که تنها نقطه ورود به application است. تگ <script> موجود در piratebadge.html برنامه را با فراخوانی این تابع راه اندازی می‌کند.

- تابع main() یک تابع سطح بالا یا top-level می‌باشد.

- متغیرها و توابع top-level عناصری هستند که خارج از ساختار تعریف کلاس ایجاد می‌شوند.

جهت اجرای برنامه در Dart Editor بر روی piratebadge.html کلیک راست نمایید و گزینه Run in Dartium را اجرا کنید. این فایل توسط Dartium اجرا می‌شود و تابع main() را فراخوانی می‌کند و صفحه ای همانند شکل زیر را نمایش می‌دهد.

گام دوم: افزودن فیلد input

توجه داشته باشید که در این مرحله یا می‌توانید تغییرات مورد نظر خود را در طی آموزش بر روی پوشه‌ی blankbadge-1 اعمال کنید و یا به پوشه‌های تهیه شده در نمونه کد موجود در همین پروژه مراجعه نمایید.

در این مرحله یک تگ <input> به تگ <”div class=”widgets> اضافه کنید.

```
...
<div>
  <div>
    <input type="text" id="inputName" maxlength="15">
  </div>
</div>
...
```

سپس کتابخانه dart:html را به ابتدای فایل piratebadge.dart اضافه کنید.

```
import 'dart:html';
```

توضیحات

- دستور فوق کلاس‌ها و Resource های موجود در کتابخانه dart:html را اضافه می‌کند.

- از حجیم شدن کدهای خود نگران نباشید، زیرا فرایند کامپایل کدهای اضافی را حذف خواهد کرد.

- کتابخانه dart:html شامل کلاسهایی جهت کار با عناصر DOM و توابعی جهت دسترسی به این عناصر می‌باشد.

- در مباحث بعدی یاد می‌گیرید که با استفاده از کلمه کلیدی show فقط کلاسهایی را import کنید که به آن نیاز دارید.

- اگر کتابخانه ای در هیچ بخش کد استفاده نشود، خود Dart Editor به صورت warning اخطار می‌دهد و می‌توانید آن را حذف

کنید.

دستور زیر را در تابع main بنویسید تا رویداد مربوط به ورود اطلاعات در فیلد input را مدیریت نمایید.

```
void main() {
  querySelector('#inputName').onInput.listen(updateBadge);
}
```

توضیحات

- تابع querySelector() در کتابخانه dart:html تعریف شده است و یک المنت DOM را جستجو می‌نماید. پارامتر ورودی آن یک selector می‌باشد که در اینجا فیلد input را توسط #inputName جستجو نمودیم که یک ID Selector می‌باشد.

- نوع خروجی این متد یک شی از نوع DOM می‌باشد.

- تابع onInput.Listen() رویدادی را برای پاسخگویی به ورود اطلاعات در فیلد input تعریف می‌کند. زمانی که کاربر اطلاعاتی را وارد نماید، تابع updateBadge فراخوانی می‌گردد.

- رویداد input زمانی رخ می‌دهد که کاربر کلیدی را از صفحه کلید فشار دهد.

- رشته‌ها همانند جاوا اسکریپت می‌توانند در " یا ' قرار بگیرند.

تابع زیر را به صورت top-level یعنی خارج از تابع main تعریف کنید.

```
...

void updateBadge(Event e) {
  querySelector('#badgeName').text = e.target.value;
}
```

توضیحات

- این تابع محتوای المنت badgeName را به محتوای وارد شده در فیلد input تغییر می‌دهد.

- پارامتر ورودی این تابع شی e از نوع Event می‌باشد و به همین دلیل می‌توانیم این تابع را یک Event Handler بنامیم.

- e.target به شی ای اشاره می‌کند که موجب رخداد رویداد شده است و در اینجا همان فیلد input می‌باشد

- با نوشتن کد فوق یک warning را مشاهده می‌کنید که بیان می‌کند ممکن است خصوصیت value برای e.target وجود نداشته باشد. برای حل این مسئله کد را بصورت زیر تغییر دهید.

```
...

void updateBadge(Event e) {
  querySelector('#badgeName').text = (e.target as InputElement).value;
}
```

توضیحات

- کلمه کلیدی as به منظور تبدیل نوع استفاده می‌شود که e.target را به یک InputElement تبدیل می‌کند.

همانند گام اول برنامه را اجرا کنید و نتیجه را مشاهده نمایید. با تایپ کردن در فیلد input به صورت همزمان در کادر قرمز رنگ نیز نتیجه تایپ را مشاهده می‌نمایید.