新版Firebase 介紹文件

上上週的課程中，我們方才學完firebase的操作方法，5/18的google創新大會上就發表了關於firebase的更新。在新版的firebase當中，我們可以看到更多更強大的功能，包含多媒體檔案儲存、更強大的線上測試、流量分析...等新的功能。不過以上所提到的相關功能其難度各方面對於初起步的web學習者未免太過複雜，所以並非我們本次練習會觸及的範疇。本次作業主要針對其Database及授權的部分進行練習，以下將逐步簡述其操作邏輯與舊版之差異。

Database

首先，我們從建立資料庫連線開始介紹：

1.建立資料庫連線

我們先來回顧一下舊版API的連線建立方法：

new Firebase(“https://<your-app-name>.firebaseio.com/”)

但嚴格來說，透過這行程式，我們就可以指向到資料庫的任何一個節點資料，不需要任何其他的驗證資料。

在新版的firebase API中，我們在與資料庫建立起連線之前，需先行驗證，並執行部分初始化的動

```
var config = {
    apiKey: “<your-App-Key>”,
    authDomain: “<your-domain>.firebaseapp.com",
    databaseURL: “https://<your-database>.firebaseio.com",
    storageBucket: “<your-bucket>.appspot.com",
  };
firebase.initializeApp(config);
```

作。長得像這樣：
或許你會對這團東西感到一頭霧水，但別擔心，打開你的新版firebase app帳號頁面，點選開發www的選項，firebase就會幫妳生成出專屬妳的configuration，照著複製貼入即可

2.選取資料作用的位置

一樣先來看一下舊版的code：

new Firebase(“https://<your-app-name>.firebaseio.com/”)

沒錯與上方相同，若我們想存取items 節點的資料，或許我們會改為：

new Firebase(“https://<your-app-name>.firebaseio.com/items”)

新版的語法概念其實是一樣的，只是寫法稍有不同。若我們與上放相同，要存取items節點的資料，新版的寫法會長成這樣：

```
firebase.database().ref("items")
```

沒錯，看起來好像差很多，但其實概念是相同的。首先，由於新版firebase增加了很多功能，存取資料庫不再只有JSON檔，所以我們必須先說明我們是要存取的事database()中的資料，而後ref()的概念就與先前相同了。所以如果要操作items旗下的firstItems，我們的code就會寫作這樣：

```
firebase.database().ref(“items/firstItems")
```

不過在這裡要注意一件事情，ref(<path>)僅適用在既存的路徑。舊版的API在寫入資料時，若發現使用者所填入的路徑不存在，會幫使用者自動生成這些不存在的節點，讓符合使用者設定的結構存入。然而在新版的api當中，若路徑不存在，就會立刻跳出錯誤並停止寫入。

3. 創造資料

完全沒改變，只是前方路徑的選擇方法改變了。以下為新版範例。

```
firebase.database().ref(“items/firstItems”).set(data);
```

```
firebase.database().ref(“items/firstItems”).push(data);
```

4.存取資料

沒改變，一樣是前方路徑選擇的方法改變了，以下為新版範例。

```
firebase.database().ref("items").orderByChild("price").sta
rtAt(10000).once("value",reProduceAll);
```

5.更新資料

在更新資料這個功能上，大概是資料庫讀寫中改變幅度比較大的一個部分。我們先來回顧一下舊版的寫法：

```
item.update({"title":title, "price":price, "descrip":descrip})
```

firebase會自動比對與資料庫重複的屬性，並將其覆蓋。這樣的方法在新版依舊可以使用，不過新版的API另外新增了這樣的方法，讓使用者更容易更新位於不同節點的資料，甚至透過這種方法創造節點路徑。

```
var data = {};
data["/messages/"+ uid+"/message"]= word;
data["/users/"+"/"+uid+"/record"]= word;
firebase.database().ref().update(messa);
```

上方這個案例想要同時更新在messages節點下某位使用者的message屬性及users結點下某位使用者的record屬性，便是採用新版firebase所提供的特殊API。這個API的用法是這樣的，先建立一個物件，把要更新資料的路徑寫在屬性中，便可以同時更新放置在不同節點的資料。
另外，我們若想要新增資料屬性也可以利用這種方法達成。例如上方的範例中users中的個別使用者本來是沒有record這個屬性的，利用這個方法，api會幫你創造目前不存在的節點或屬性。

6. 刪除資料

在刪除資料的方法上，也與原先的方法相同，以下為新版範例。

```
firebase.database().ref(“items/firstItems”).remove();
```

auth（使用者登入）

firebase在使用者登入這個部分做了蠻大幅度的變動，在舊版的api中，使用者的登入與否是註冊在資料節點上進行監聽的。新版的api則將其獨立出來，並新增一個頁面供使用者查看目前已註冊的人員身份，功能更臻完整。

初始化開通的設定與之前完全相同。在開通設定後，便可以開始來練習本次的作業。

首先，若要採用facebook或其他第三方進行授權登入，在新版的API中，我們需要先創造一個提供者的物件像這樣：

```
var fbProvider = new firebase.auth.FacebookAuthProvider();
```

透過這個物件firebase才有辦法判斷你將使用哪種方式進行第三方資料授權。有了這個我們就可以開始讓使用者執行登入了！

1.使用者登入（跳出視窗）

使用者在點擊signin之後，我們希望跳出一個視窗讓使用者執行登入的動作，這時該怎麼做呢？

```
firebase.auth().signInWithPopup(fbProvider).then(function
(result) {
    //登入後的頁面行為
   });
  }).catch(function (error) {
    various errorCode = error.code;
    var errorMessa = error.message;
    console.log(errorCode,errorMessa);
  })
```

利用firebase.auth()這個屬性可以存取到與使用者登入、資料等相關的方法，而使用signInWithPopup()這個方法，firebase的library就會自動啟動一個跳出的facebook視窗。signInWithPopup()這個方法的括號就需放入我們先前所創造的提供者物件。登入完的結果就會傳入then()的result變數當中，登入後的頁面行為就在then當中執行。而result中就可以存取到登入相關的資訊。利用：

result.user.displayName 存取使用者臉書名稱，字串
result.user.uid 使用者在臉書系統中的特殊編號，字串
result.user.photoURL 使用者的臉書大頭貼，字串，提供暫時性的圖片連結

2.使用者登出

點下signout後，要讓頁面登出又該怎麼做呢？

```
firebase.auth().signOut().then(function() {
    //登出後所執行之行為
  },function (error) {
    console.log(error.code);
  });
```

概念其實與前者相當，只不過是利用signOut()這個方法。

3.監測登入與登出狀態

使用者在進入頁面時，可能已經登入的狀態。若還需要用點擊signIn才能正常顯示頁面，在流程上會顯得有些不便，firebase也提供了偵測登入狀態的API讓我們能夠利用簡單的function瞭解使用者目前的登入狀況。

```
firebase.auth().onAuthStateChanged(function (user) {
      if(user){

      }else{

      }
}
```

這個方法在幾個狀態下會觸發：1.登入狀態改變時 2.進入頁面時
當使用者沒有登入時，user變數的值會是null，登入後user就會變成一個物件透過先前提到的屬性名稱就可以存取到自己所需要的資料。

本次作業所運用的方法不外乎就是以上幾種了，立刻拿出作業來練習看看吧！