"await"

تستخدم في JavaScript للانتظار لاستجابة دالة غير متزامنة (مُعلنة بـ "async") واسترداد القيمة التي تم إرجاعها منها. يتم استخدام "await" داخل الدالة المُعلنة بـ "async" لتوقف تنفيذ الكود حتى يتم استكمال الدالة الغير متزامنة وإرجاع القيمة.

على سبيل المثال، إذا كان لدينا دالة غير متزامنة تقوم بجلب بيانات من خادم API:

javascript

```
async function fetchData() {
const response = await
fetch('https://api.example.com/data');
const data = await
response.json();
  return data;
}
```

في المثال أعلاه، يتم استخدام "await" للاستمرار في تنفيذ الكود حتى يتم استجابة الخادم API وإرجاع البيانات بصيغة JSON. يمكن استخدام "await" للانتظار لأي دالة غير متزامنة، مثل قراءة ملف أو استجابة طلب HTTP.

ويجب أن يتم استخدام "await" فقط داخل الدوال المُعلنة بـ "async". إذا تم استخدامها خارج هذه الدوال، ستحدث خطأ في الجافا سكريبت.

"async"

وهو مصطلح يستخدم في برمجة JavaScript للإشارة إلى الوظائف الغير متزامنة.

وظيفة "async" تعني أنها تعمل بشكل غير متزامن، أي أنها تستمر في التنفيذ دون الانتظار لاستكمال الوظائف الأخرى. هذا يسمح للبرنامج بالاستمرار في التنفيذ والقيام بأعمال أخرى في الوقت نفسه بدلاً من الانتظار لانتهاء الوظيفة.

تستخدم كلمة "async" قبل تعريف الدالة في JavaScript. على سبيل المثال:

```
javascript
async function fetchData () {
const response = await fetch('https://api.example.com/data');
const data = await
response.json؛()
processData(data);
doSomethingElse؛()
}
```

في المثال أعلاه، يتم استخدام الكلمة المفتاحية "async" قبل تعريف الوظيفة "fetchData". هذا يشير إلى أن الوظيفة تعمل بشكل غير متزامن. يتم استخدام الكلمة المفتاحية "await" داخل الوظيفة للانتظار لاستجابة الخادم API وتحويل البيانات إلى صيغة JSON.

باستخدام "async" و "await"، يمكنك تنفيذ عمليات متعددة بشكل غير متزامن في JavaScript، مما يسمح بتحسين أداء التطبيق وتجنب تجميد واجهة المستخدم أثناء انتظار استجابة الخادم.