شرح async & await في جافا سكريبت:

1. مفهوم async:

- async: هو معيار جديد في جافا سكريبت يتم استخدامه لتعريف دالة متزامنة (synchronous)، وهي دالة تستخدم مفهوم الوعد (Promise) لإدارة التنفيذ الغير متزامن.

- قبل كلمة async، يتم تعريف الدالة لكي تعمل بشكل غير متزامن، وبالتالي يمكنها الاستفادة من الوعود والانتظار على العمليات الغير متزامنة.

2. مفهوم await:

- await: يستخدم داخل دالة متزامنة مُعرّفة باستخدام async.

- يُستخدم للانتظار على إكمال وعود (Promises) الغير متزامنة واسترجاع القيمة المسترجعة منها.

- عند استخدام await داخل دالة متزامنة، يتم تعليق تنفيذ الدالة حتى يتم إنجاز الوعد (Promise) واسترجاع النتيجة.

3. الفوائد والاستخدامات:

- يسهل async و await كتابة رمز جافا سكريبت أكثر قراءةً وفهمًا للعمليات غير المتزامنة.

- يقلل من استخدام التعشيش المتعدد للوعود (Promise chaining)، مما يساعد على تبسيط الشفرة.

- يمكن استخدام async و await معًا للتعامل مع العمليات الغير متزامنة مثل استدعاءات API، القراءة/الكتابة من قواعد البيانات، والعمليات الزمنية الأخرى.

4. ملاحظات إضافية:

- يجب أن يتم استخدام await فقط داخل دالة متزامنة مُعرّفة بـ await.
- يمكن استخدام try/catch للتعامل مع الأخطاء أثناء استخدام await.
- يمكن استخدام await مع الوعود (Promises) المخصصة أو الوظائف المتزامنة المعرفة باستخدام async.

5. أمثلة:

-مثال async:

```
async function fetchData(){
// الكود هنا
}
```

-مثال await:

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

شرح Utils:

- "Utils" هي اختصار لكلمة "Utilities" وتشير إلى مجموعة من الأدوات العامة أو الوظائف الفائدة في البرمجة.

1- نطاق استخدام Utils:

- تحويل البيانات: يمكن استخدام Utils لتحويل البيانات من نوع لآخر، مثل تحويل السلاسل إلى أرقام أو تنسيق التواريخ والأوقات.

- التلاعب بالسلاسل: توفر Utils وظائف لمعالجة السلاسل مثل قص السلاسل، وتجزئتها، وتعويض النصوص.

- إدارة المصفوفات: يمكن استخدام Utils لأغراض مثل فرز المصفوفات، وإزالة العناصر المكررة، وتحويل المصفوفات إلى تنسيقات أخرى.

- التعامل مع التواريخ والوقت: توفر Utils وظائف للتعامل مع التواريخ والأوقات، مثل تحويل التواريخ إلى سلاسل، وحساب الفروق الزمنية، وتنسيق الوقت.

- إدارة الملفات والمجلدات: يمكن استخدام Utils لإجراء عمليات مثل قراءة وكتابة الملفات، وإنشاء وحذف المجلدات، ونسخ ونقل الملفات.

- التعامل مع الشبكة: يوفر Utils وظائف للتعامل مع الشبكة، مثل إجراء طلبات HTTP، وإرسال بيانات POST/GET، وتحليل استجابات الشبكة.

2- مكونات Utils:

تختلف المكونات الرئيسية التي تشكل Utils من لغة برمجة إلى أخرى وحسب السياق:

- في جافا (Java):

- java.lang.Matz: يحتوي على عدة وظائف رياضية مفيدة مثل الجذر التربيعي والجيب الزائد وحساب الأرقام العشوائية.
- java.util.Arrays: يوفر وظائف لإجراء عمليات على المصفوفات مثل الفرز والبحث والمقارنة.
- java.util.Date: يسمح بإنشاء وإدارة الأوقات والتواريخ.

- <u>في بايثون (Python):</u>
- random: يوفر وظائف لإنشاء أرقام عشوائية.
- datetime: يسمح بإنشاء وإدارة الأوقات والتواريخ.
- os: يوفر وظائف للتفاعل مع نظام التشغيل مثل الوصول إلى الملفات والمجلدات.

- <u>في جافا سكريبت (JavaScript):</u>
- Math: يحتويعلى وظائف رياضية مثل الجذر التربيعي والجيب الزائد وحساب الأرقام العشوائية.
- String: يحتوي على وظائف للتعامل مع السلاسل مثل قص السلاسل وتجزئتها.
- Array: يوفر وظائف لإجراء عمليات على المصفوفات مثل الفرز والبحث والتحويل.

3- <u>الفوائد والاستخدامات:</u>

- توفر Utils وظائف وأدوات جاهزة للتعامل مع المهام الشائعة والتكرارية، مما يوفر الوقت والجهد في كتابة الشفرة من الصفر.

- تعزز إعادة الاستخدام والتنسيق في الشفرة، حيث يمكن استخدام مكونات Utils في مشاريع مختلفة وعلى مستوى مختلف من التعقيد.

- تساعد في تبسيط وتنظيم الشفرة، حيث يمكن تجميع الوظائف ذات الصلة في وحدات Utils واستخدامها بشكل مركزي في المشروع.

- تعزز قابلية الصيانة وتحسين الأداء، حيث يمكن تحسين وتحديث المكونات المستخدمة في Utils بشكل مستقل بدون تأثير على بقية الشفرة.

- استخدام Utils يساعد المطورين على تحقيق الكفاءة والإنتاجية في عملية التطوير البرمجي من خلال توفير وظائف مفيدة وأدوات قوية للتعامل مع العمليات الشائعة.