

شرح for loop



شرح loop في for خطوه بخطوه



أكيد! الـ `for loop` تُستخدم لتكرار مجموعة من التعليمات عددًا معينًا من المرات أو للمرور عبر عناصر مجموعة معينة (مثل قائمة أو سلسلة). سأشرحها خطوة بخطوة:

الصيغة العامة لـ `for loop` :

python

```
for variable in iterable:  
    # كتلة التعليمات
```

- `variable` : متغير مؤقت يأخذ قيمة كل عنصر من عناصر المجموعة (iterable) في كل دورة.
- `iterable` : المجموعة التي سنمر عبر عناصرها مثل قائمة (list)، مجموعة (set)، سلسلة (string)، أو نطاق (range).
- **كتلة التعليمات**: التعليمات التي سيتم تنفيذها خلال كل دورة.

خطوات التنفيذ:

1. تحديد المجموعة أو النطاق (iterable)

- حدد البيانات التي ستقوم بالدوران خلالها. يمكن أن تكون قائمة أرقام، نصوص، أو حتى سلسلة نصوص.
- على سبيل المثال:

python

```
numbers = [1, 2, 3, 4, 5]
```

2. تهيئة الـ loop

- حدد متغيرًا مؤقتًا سيحمل قيمة العنصر الحالي في كل دورة.
- على سبيل المثال:

python

```
for num in numbers:
```

3. تنفيذ التعليمات

- داخل جسم الحلقة (body of the loop)، يتم تنفيذ التعليمات باستخدام قيمة المتغير المؤقت.
- مثال:

python

```
print(num)
```

4. التكرار التلقائي

- ينتقل البرنامج تلقائيًا إلى العنصر التالي في المجموعة. إذا انتهت المجموعة، يتوقف التكرار.

مثال عملي:

الكود:

python

```
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    print(f"الرقم الحالي هو: {num}")
```

الخطوات:

1. الدورة الأولى:

```
`num = 1` ○
`print(f"الرقم الحالي هو: 1")` ○
```

2. الدورة الثانية:

```
`num = 2` ○
`print(f"الرقم الحالي هو: 2")` ○
```

3. الدورة الثالثة:

```
`num = 3` ○
`print(f"الرقم الحالي هو: 3")` ○
```

النتيجة:

```
الرقم الحالي هو: 1
الرقم الحالي هو: 2
الرقم الحالي هو: 3
الرقم الحالي هو: 4
الرقم الحالي هو: 5
```

مثال مع النطاق (range):

python

```
for i in range(1, 6):
    print(f"القيمة الحالية: {i}")
```

- `range(1, 6)`: ينشئ نطاقًا يبدأ من 1 وينتهي عند 5 (غير شامل للـ 6).
- النتيجة:

```
القيمة الحالية: 1
القيمة الحالية: 2
القيمة الحالية: 3
القيمة الحالية: 4
القيمة الحالية: 5
```

هل تريد شرحًا أعمق أو تطبيقًا محددًا؟ 😊



عايزه اسئلته وابعثلك حل واريد منك ان تقول لي اجابه صح ام خطأ



بالتأكيد، سأكون سعيدًا بمراجعة إجاباتك! أرسل الأسئلة مع إجاباتك، وسأخبرك إذا كانت الإجابة صحيحة أم لا مع شرح إذا لزم الأمر. 😊



ابعثيلي عن اسئلته عن loop في حاله for



بالطبع! إليك مجموعة من الأسئلة عن `for loop` مع تنوع في الصعوبة:

الأسئلة:

1. **السؤال الأول:** ماذا ستكون النتيجة عند تنفيذ الكود التالي؟

python

```
for i in range(3):  
    print(i)
```

2. **السؤال الثاني:** أكمل الكود التالي لطباعة جميع العناصر في القائمة:

python

```
fruits = ["apple", "banana", "cherry"]  
_____ fruit __ fruits:  
    print(fruit)
```

3. **السؤال الثالث:** ماذا تفعل الحلقة التالية؟

python

```
for char in "hello":  
    print(char)
```

4. **السؤال الرابع:** اكتب كودًا يستخدم `for loop` لطباعة جميع الأرقام الزوجية بين 1 و10.

5. **السؤال الخامس:** ما هي المشكلة في الكود التالي؟ وإذا لم يكن هناك مشكلة، فما هي النتيجة؟

python

```
for i in range(1, 5):  
    print("Hello")
```

6. **السؤال السادس:** إذا كانت لديك قائمة كالتالي:

python

```
numbers = [10, 20, 30, 40, 50]
```

كيف يمكن كتابة كود يستخدم `for loop` لجمع كل القيم في القائمة؟

7. السؤال السابع: ماذا ستكون النتيجة النهائية لهذا الكود؟

python

```
for i in range(2, 8, 2):
    print(i)
```

8. السؤال الثامن: ما هو دور الكلمة `break` داخل `for loop`؟ وضح الإجابة بمثال.

9. السؤال التاسع: اكتب برنامجًا يستخدم `for loop` لطباعة الأرقام من 1 إلى 10، مع تخطي الرقم 5 باستخدام `continue`.

10. السؤال العاشر: هل يمكن استخدام `else` مع `for loop`؟ إذا كانت الإجابة نعم، فقدم مثالاً.

ارسل إجاباتك وسأراجعها! 😊



```
} for (let i = 1; i < 4; i++) {
    console.log(i)
}
```



إجابتك ليست صحيحة إذا كنت تقصد كتابة الكود بلغة **Python**، حيث أن الكود الذي كتبته هو بصيغة JavaScript. صيغة الكود الصحيح في **Python** هي:

python

```
for i in range(3):
    print(i)
```

النتيجة:

```
0
1
2
```

تفسير النتيجة:

- `range(3)` ينشئ نطاقاً من القيم: `[0, 1, 2]` (تبدأ من 0 وتنتهي قبل 3).
- يتم تمرير كل قيمة إلى المتغير `i` ثم طباعتها باستخدام `print(i)`.

إذا كنت تريد كتابة الكود بلغة **JavaScript** مثلما فعلت:

javascript

```
for (let i = 1; i < 4; i++) {
  console.log(i);
}
```

النتيجة في هذه الحالة:

```
1
2
3
```

لكن صيغة السؤال كانت تخص **Python**، لذا تأكد من استخدام اللغة المناسبة! 😊



فقط لغه جافا سكريبت



فهمت الآن، سأعتمد JavaScript في جميع الشروحات القادمة. إجابتك صحيحة في هذه الحالة. إذا أردت شرحاً إضافياً أو أي سؤال آخر، فأنا هنا! 😊



شرح السؤال الثاني



السؤال الثاني كان كالتالي:

أكمل الكود التالي لطباعة جميع العناصر في القائمة:

javascript

```
const fruits = ["apple", "banana", "cherry"];
_____ fruit __ fruits {
  console.log(fruit);
}
```

الإجابة الصحيحة:

javascript

```
const fruits = ["apple", "banana", "cherry"];
for (const fruit of fruits) {
  console.log(fruit);
}
```

الشرح:

1. ``const fruits = ["apple", "banana", "cherry"]``
 - هنا يتم تعريف مصفوفة تحتوي على العناصر: ``"apple"`, `"banana"`, `"cherry"`.`
2. ``for (const fruit of fruits)``
 - تستخدم عبارة ``for...of`` للمرور على كل عنصر في المصفوفة `.fruits`.
 - في كل دورة، المتغير ``fruit`` يأخذ قيمة العنصر الحالي في المصفوفة.
3. ``console.log(fruit)``
 - في كل دورة، يتم طباعة قيمة المتغير ``fruit`` (العنصر الحالي في المصفوفة).

النتيجة عند التنفيذ:

```
apple
banana
cherry
```

ملاحظات إضافية:

- ``for...of`` يُستخدم للمرور على القيم في المصفوفة.
- إذا أردت المرور على الفهارس (indices) بدلاً من القيم، يمكنك استخدام ``for...in``:

javascript

```
for (const index in fruits) {
  console.log(index); // 2, 1, 0 طباعة الفهارس:
}
```

هل تحتاج المزيد من الأمثلة أو شرحًا أعمق؟ 😊



شرح السؤال الثالث



السؤال الثالث كان كالتالي:

ماذا تفعل الحلقة التالية؟

javascript

```
for (const char of "hello") {
  console.log(char);
}
```

الشرح:

1. السلسلة النصية `"hello"`:

○ هي سلسلة تحتوي على الأحرف: ``h`, `e`, `l`, `l`, `o``.

2. `for (const char of "hello")``:

○ تستخدم عبارة `for...of`` للمرور على كل عنصر من عناصر السلسلة النصية `"hello"`.

○ السلاسل النصية في JavaScript تعتبر **كائنات قابلة للتكرار (iterable)**، مما يعني أن كل حرف في النص يمكن التعامل معه كعنصر مستقل.

3. `console.log(char)``:

○ في كل دورة من الحلقة، يتم تخزين الحرف الحالي في المتغير `char``، ثم يُطبع باستخدام `console.log``.

ماذا تفعل الحلقة؟

- الحلقة تمر على كل حرف في النص `"hello"`، وتطبع كل حرف على سطر جديد.


```
h
e
l
l
o
```

توضيحات إضافية:

• كيف تعمل الحلقة؟

- الدورة الأولى: `char = 'h'` → طباعة `'h'`.
- الدورة الثانية: `char = 'e'` → طباعة `'e'`.
- الدورة الثالثة: `char = 'l'` → طباعة `'l'`.
- الدورة الرابعة: `char = 'l'` → طباعة `'l'`.
- الدورة الخامسة: `char = 'o'` → طباعة `'o'`.

• `for...of` مع النصوص:

- يمكنك استخدام `for...of` ليس فقط مع المصفوفات، بل أيضًا مع النصوص لتفكيك النص إلى أحرف فردية.

إذا كنت تريد تطبيقات أخرى على النصوص أو توضيحات إضافية، أخبرني! 😊



```
السؤال الرابع الكود:
for (let i = 0; i < 10; i++) {
  if (0 % 2 == 1)
    ;console.log("even")
  } else if (0 % 2 == 2) {
    ;console.log("odd")
  }
}
```



الكود الذي كتبته يحتوي على أخطاء منطقية، وسأوضحها خطوة بخطوة:

الكود الذي كتبته:

javascript

```
for (let i = 0; i < 10; i++) {
  if (0 % 2 == 1) {
```

```
console.log("even");
} else if (0 % 2 == 2) {
  console.log("odd");
}
}
```

الأخطاء في الكود:

1. الثابت `0` بدلاً من المتغير `i`:

- داخل الحلقة، تستخدم الثابت `0` في الشرطين:

javascript

```
if (0 % 2 == 1)
```

- هذا يعني أنك لا تتحقق من قيم المتغير `i` بل من الرقم الثابت `0` طوال الوقت، مما يؤدي إلى نتائج غير صحيحة.

2. الشرط الثاني غير منطقي:

- في الشرط:

javascript

```
else if (0 % 2 == 2)
```

- باقي القسمة على 2 لا يمكن أن يساوي 2 لأن الناتج دائماً يكون إما `0` أو `1`.

3. طباعة القيم الخاطئة:

- الكود لا يُحدد إذا كان الرقم فردياً أو زوجياً بشكل صحيح بسبب الأخطاء المذكورة.

الكود المعدّل والصحيح:

إذا كنت تريد طباعة الأرقام الزوجية فقط بين 1 و10، إليك الكود الصحيح:

javascript

```
for (let i = 1; i <= 10; i++) {
  if (i % 2 === 0) {
    console.log(`${i} is even`);
  } else {
    console.log(`${i} is odd`);
  }
}
```

شرح الكود الصحيح:

1. ``for (let i = 1; i <= 10; i++)``

○ تبدأ الحلقة من الرقم `1` وتنتهي عند الرقم `10`.

2. ``if (i % 2 === 0)``

○ يتحقق من إذا كان الرقم الحالي زوجيًا (باقي القسمة على 2 يساوي 0).

3. ``; console.log(`${i} is even`)``

○ إذا كان الرقم زوجيًا، يتم طباعته مع النص "is even".

4. ``else``

○ إذا لم يكن الرقم زوجيًا، فإنه فردي، ويُطبع مع النص "is odd".

النتيجة عند التنفيذ:

csharp

```
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
10 is even
```

إذا كنت تقصد شرطًا مختلفًا أو طريقة أخرى للحل، فقط أخبرني! 😊



```
for (let i = 0; i < 10; i++) {
  if (i % 2 == 0) {
    console.log("even");
  } else if (i % 2 == 1) {
    console.log("odd");
  }
}
```



الكود الذي كتبته صحيح، وسأوضح كيف يعمل خطوة بخطوة:

الكود:

```
for (let i = 0; i < 10; i++) {
  if (i % 2 == 0) {
    console.log("even");
  } else if (i % 2 == 1) {
    console.log("odd");
  }
}
```

الشرح:

1. الحلقة (for loop):

- `let i = 0` : يبدأ المتغير `i` بالقيمة 0.
- `i < 10` : تستمر الحلقة طالما أن `i` أقل من 10.
- `++i` : يتم زيادة `i` بمقدار 1 في كل دورة.

2. الشرط الأول (if):

- `i % 2 == 0` : يتحقق مما إذا كان باقي قسمة `i` على 2 يساوي 0 (أي أن الرقم زوجي).
- إذا تحقق الشرط، يتم طباعة `"even"`.

3. الشرط الثاني (else if):

- `i % 2 == 1` : يتحقق مما إذا كان باقي قسمة `i` على 2 يساوي 1 (أي أن الرقم فردي).
- إذا تحقق الشرط، يتم طباعة `"odd"`.

النتيجة عند التنفيذ:

- الدورة الأولى: `0 % 2 == 0` → `i = 0` → طباعة `"even"`.
- الدورة الثانية: `1 % 2 == 1` → `i = 1` → طباعة `"odd"`.
- الدورة الثالثة: `2 % 2 == 0` → `i = 2` → طباعة `"even"`.
- الدورة الرابعة: `3 % 2 == 1` → `i = 3` → طباعة `"odd"`.
- وهكذا...

النتيجة الكاملة:

```
even
odd
even
odd
even
odd
even
```

```
odd
even
odd
```

ملاحظات إضافية:

- إذا أردت طباعة الرقم مع النص، يمكنك تعديل الكود كالتالي:

javascript

```
for (let i = 0; i < 10; i++) {
  if (i % 2 == 0) {
    console.log(`${i} is even`);
  } else {
    console.log(`${i} is odd`);
  }
}
```

النتيجة مع الرقم:

csharp

```
0 is even
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
```

إذا كان لديك سؤال إضافي، أنا هنا للمساعدة! 😊



شرح السؤال الخامس



السؤال الخامس كان كالتالي:

ما هي المشكلة في الكود التالي؟ وإذا لم يكن هناك مشكلة، فما هي النتيجة؟

javascript

```
for (let i = 1; i < 5; i++) {
  console.log("Hello");
}
```

}

الشرح:

1. هل هناك مشكلة في الكود؟

- لا توجد مشكلة في الكود.
- الكود مكتوب بشكل صحيح وسيعمل كما هو متوقع.

2. كيف يعمل الكود؟

- الحلقة `for` تحتوي على ثلاثة أجزاء:
 1. `let i = 1` : يبدأ المتغير `i` بقيمة 1.
 2. `i < 5` : تستمر الحلقة طالما أن قيمة `i` أقل من 5.
 3. `++i` : يتم زيادة `i` بمقدار 1 بعد كل دورة.
- `console.log("Hello")` : في كل دورة، يتم طباعة النص `"Hello"`.

3. عدد مرات التنفيذ:

- يبدأ `i` من 1 وينتهي عند الوصول إلى 5 (ولكن الحلقة تتوقف عند أن تصبح `i = 5`، لأن الشرط `i < 5` لم يعد صحيحًا).

4. عدد المرات التي تطبع فيها `"Hello"`:

- القيم التي يمر بها `i`: 1, 2, 3, 4.
- لذلك، يتم طباعة `"Hello"` 4 مرات.

النتيجة عند التنفيذ:

```
Hello
Hello
Hello
Hello
```

ملاحظات إضافية:

- إذا أردت تضمين قيمة `i` مع النص المطبوع، يمكنك تعديل الكود كالتالي:

javascript

```
for (let i = 1; i < 5; i++) {  
  console.log(`Hello ${i}`);  
}
```

النتيجة مع تعديل الكود:

```
Hello 1  
Hello 2  
Hello 3  
Hello 4
```

هل تحتاج إلى أمثلة إضافية أو توضيحات أكثر؟ 😊



السؤال السادس



السؤال السادس كان كالتالي:

إذا كانت لديك قائمة كالتالي:

javascript

```
const numbers = [10, 20, 30, 40, 50];
```

كيف يمكن كتابة كود يستخدم `for loop` لجمع كل القيم في القائمة؟

الإجابة:

لحل هذا السؤال، يمكننا استخدام حلقة `for` للتكرار عبر جميع العناصر في المصفوفة وجمع القيم. سنحتاج إلى متغير لحفظ مجموع الأرقام.

الكود الصحيح:

```
const numbers = [10, 20, 30, 40, 50];
let sum = 0;

for (let i = 0; i < numbers.length; i++) {
  sum += numbers[i]; // جمع الرقم الحالي إلى المجموع
}

console.log("مجموع الأرقام هو:", sum);
```

الشرح:

1. `const numbers = [10, 20, 30, 40, 50];` لدينا مصفوفة تحتوي على الأرقام: `[10, 20, 30, 40, 50]`.
2. `let sum = 0;` نبدأ بتعيين المتغير `sum` إلى `0` لأننا نريد جمع الأرقام.
3. `for (let i = 0; i < numbers.length; i++)` تبدأ الحلقة من `i = 0` وتستمر حتى تصل إلى طول المصفوفة (`numbers.length`).
4. `sum += numbers[i];` في كل دورة من الحلقة، يتم إضافة الرقم الحالي في المصفوفة (`numbers[i]`) إلى المتغير `sum`.
5. `console.log("مجموع الأرقام هو:", sum);` بعد الانتهاء من جميع الدورات، يتم طباعة المجموع النهائي.

النتيجة عند التنفيذ:

مجموع الأرقام هو: 150

ملاحظات:

- `numbers.length` يعطي طول المصفوفة، أي عدد العناصر في المصفوفة.
- كل دورة من الحلقة تضيف العنصر الحالي إلى `sum` حتى يتم جمع جميع العناصر.

إذا كنت بحاجة إلى شرح آخر أو مثال إضافي، أخبرني! 😊



مطلوب ان تبعت لي سؤال واستنى لما ابغىلك اجابه عن for باستخدام جافا سكريبت



بالطبع! إليك السؤال:

السؤال:

اكتب كودًا باستخدام `for loop` في JavaScript للطباعة لجميع الأرقام الفردية بين 1 و 20 (بما في ذلك 1 و 20).
انتظر إجابتك! 😊



```
for (let i = 0; i < 20; i++) {
  if (i % 2 === 1) {
    console.log("add");
  } else console.log("even");
}
```



إجابتك قريبة من الصحيح، ولكن هناك بعض النقاط التي تحتاج إلى تعديل:

الملاحظات:

1. الحلقة تبدأ من `i = 0`:

- الرقم 0 ليس فرديًا، ولكنك بدأت من 0. إذا كنت ترغب في الطباعة للأرقام الفردية من 1 إلى 20، يجب أن تبدأ الحلقة من 1.

2. الطباعة:

- أنت تطبع `add` للأرقام الفردية، ولكن من الأفضل أن تطبع `odd` (لأن "odd" تعني "فردية").
- أما بالنسبة للأرقام الزوجية، فإن `even` هو المصطلح الصحيح.

التعديل على الكود:

javascript

```
for (let i = 1; i <= 20; i++) {
  if (i % 2 === 1) {
    console.log("odd");
  } else {
    console.log("even");
  }
}
```

الشرح:

1. ``for (let i = 1; i <= 20; i++)`` .
 ○ تبدأ الحلقة من `1` وتستمر حتى تصل إلى `20` (بما في ذلك 20).
 2. ``if (i % 2 === 1)`` .
 ○ يتحقق مما إذا كان الرقم فرديًا. إذا كان كذلك، يتم طباعة ``"odd"``.
 3. ``else`` .
 ○ إذا لم يكن الرقم فرديًا (أي أنه زوجي)، يتم طباعة ``"even"``.

النتيجة عند التنفيذ:

```
odd
even
odd
even
odd
even
odd
even
odd
even
odd
even
odd
even
odd
even
odd
even
odd
even
```

إذا كنت تحتاج إلى مزيد من التوضيح أو مثال آخر، لا تتردد في سؤالني! 😊



السؤال الثاني



بالطبع! إليك السؤال الثاني:

السؤال:

اكتب كودًا باستخدام ``for loop`` في JavaScript لطباعة الأرقام من 1 إلى 10 في تناقص.
 انتظر إجابتك! 😊



```
for (let i = 0; i > 10; i--) {
  console.log(i);
}
```



إجابتك تحتوي على خطأ بسيط في الشرط. لنقم بمراجعته:

الملاحظات:

1. الشرط في الحلقة:

- أنت استخدمت الشرط `i > 10`، وهو غير صحيح في هذه الحالة لأنك تبدأ من `i = 0` ولن يتحقق الشرط أبداً. يجب أن يكون الشرط `i >= 1` لكي تستمر الحلقة حتى تصل إلى الرقم 1.

2. بدء العد من 10:

- يجب أن تبدأ الحلقة من `i = 10` بدلاً من `i = 0` لأنك تريد الطباعة من 10 إلى 1.

الكود المعدل:

javascript

```
for (let i = 10; i >= 1; i--) {
  console.log(i);
}
```

الشرح:

1. `let i = 10;`

- تبدأ الحلقة من `i = 10`.

2. `i >= 1;`

- الحلقة ستستمر طالما أن `i` أكبر من أو يساوي 1.

3. `i--;`

- يتم تقليل قيمة `i` بمقدار 1 في كل دورة.

النتيجة عند التنفيذ:

```
10
9
8
7
```

6
5
4
3
2
1

هل تحتاج إلى توضيح آخر أو مزيد من الأمثلة؟ 😊



السؤال الثالث



إليك السؤال الثالث:

السؤال:

اكتب كودًا باستخدام `for loop` في JavaScript لطباعة جميع الأرقام التي هي قسمة للعدد 3 بين 1 و 30.

انتظر إجابتك! 😊



```
for (let i = 1; i < 30; i++) {
  if (i / 3) {
    console.log(i);
  } else if (!(1 / 3)) {
    console.log("nan");
  }
}
```



إجابتك تحتوي على بعض الأخطاء في الشروط والعمليات الحسابية. دعني أوضح لك التفاصيل:

الملاحظات:

1. الشرط `(i / 3)` في `if`:

- هذا الشرط لا يقوم بالتحقق مما إذا كان الرقم قابلاً للقسمة على 3. لأن $i / 3$ سيعطيك النتيجة بدون التحقق من باقي القسمة. عليك استخدام $i \% 3 === 0$ للتحقق مما إذا كان العدد قابلاً للقسمة على 3.

2. `(!(1 / 3))` في `else if`:

- هذا الشرط غير صحيح. التعبير $(3 / 1)!$ سينتج عنه `false` دائماً لأن $3 / 1$ ليس `NaN` أو قيمة غير قابلة للتحويل إلى `false`. هذا غير مرتبط بمفهوم القسمة على 3.

الكود المعدل:

javascript

```
for (let i = 1; i <= 30; i++) {
  if (i % 3 === 0) {
    console.log(i);
  }
}
```

الشرح:

1. `for (let i = 1; i <= 30; i++)`

○ تبدأ الحلقة من `i = 1` وتستمر حتى `i = 30`.

2. `if (i % 3 === 0)`

○ يتحقق إذا كانت النتيجة `باقي القسمة` للعدد `i` على 3 تساوي 0 (أي أن `i` قابل للقسمة على 3).

3. `console.log(i)`

○ في حال تحقق الشرط، يتم طباعة `i`.

النتيجة عند التنفيذ:

```
3
6
9
12
15
18
21
24
27
30
```

هل هناك شيء آخر ترغب في توضيحه أو تحتاج إلى تعديل؟ 😊



السؤال الرابع



إليك السؤال الرابع:

السؤال:

اكتب كودًا باستخدام `for loop` في JavaScript لطباعة جميع الأرقام الزوجية بين 1 و 50 (بما في ذلك 50).

انتظر إجابتك! 😊



```
for (let i = 0; i < 50; i++) {
  if (i % 2 === 0) {
    console.log("even");
  } else console.log("add");
}
```



إجابتك قريبة من الصحيح، لكن هناك بعض التعديلات التي يجب إجراؤها:

الملاحظات:

1. شرط طباعة "even" للأرقام الزوجية:

- صحيح أنك تتحقق من الأرقام الزوجية باستخدام `i % 2 === 0`، لكن الكود يطبع `even` لكل الأرقام الزوجية، بينما من الأفضل أن تطبع الرقم نفسه مع النص (مثلًا: `is even 2`).

2. الطباعة للأرقام الفردية:

- في حالة الأرقام الفردية، قمت باستخدام `add` بدلاً من `odd`. من الأفضل استخدام `odd` لأنه المصطلح الصحيح.

3. تعديل نطاق الحلقة:

- الحلقة يجب أن تشمل الرقم 50، لذا يجب تعديل شرط التكرار ليكون `i <= 50` بدلاً من `i < 50`.

الكود المعدل:

javascript

```
for (let i = 0; i <= 50; i++) {
  if (i % 2 === 0) {
    console.log(`${i} is even`);
  } else {
    console.log(`${i} is odd`);
  }
}
```

الشرح:

1. `for (let i = 0; i <= 50; i++)`

○ تبدأ الحلقة من `0` وتستمر حتى `50` (بما في ذلك 50).

2. `if (i % 2 === 0)`

○ يتحقق إذا كانت قيمة `i` زوجية (أي أن باقي القسمة على 2 يساوي 0).

3. `console.log(`\${i} is even`);`

○ إذا كانت قيمة `i` زوجية، يتم طباعة الرقم مع النص `"is even"`.

4. `else`

○ إذا كانت قيمة `i` فردية، يتم طباعة الرقم مع النص `"is odd"`.

النتيجة عند التنفيذ:

csharp

```
0 is even
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
7 is odd
8 is even
9 is odd
10 is even
11 is odd
12 is even
13 is odd
14 is even
15 is odd
16 is even
17 is odd
18 is even
19 is odd
20 is even
21 is odd
22 is even
23 is odd
24 is even
25 is odd
26 is even
27 is odd
28 is even
29 is odd
30 is even
31 is odd
32 is even
33 is odd
34 is even
35 is odd
36 is even
37 is odd
38 is even
39 is odd
40 is even
41 is odd
42 is even
43 is odd
44 is even
45 is odd
46 is even
47 is odd
48 is even
```

49 is odd
50 is even

هل لديك أي استفسار آخر أو تحتاج إلى توضيح؟ 😊



السؤال الثاني



إليك السؤال الثاني:

السؤال:

اكتب كودًا باستخدام `for loop` في JavaScript لطباعة حاصل ضرب جميع الأرقام من 1 إلى 10.
انتظر إجابتك! 😊



طريقه حل



لحل هذا السؤال، يجب أن تقوم بحساب حاصل ضرب الأرقام من 1 إلى 10. هذا يعني أنه يجب عليك استخدام متغير لتخزين النتيجة المتزايدة من كل دورة في الحلقة.

الخطوات:

1. إعداد متغير لتخزين النتيجة:

◦ في البداية، يجب تعيين متغير مثل `result` ليكون 1 لأنه سيكون لدينا ضرب.

2. استخدام حلقة `for`:

◦ يمكنك استخدام `for loop` للتكرار من 1 إلى 10.

3. حساب حاصل الضرب داخل الحلقة:

◦ في كل دورة من الحلقة، نقوم بضرب قيمة `result` الحالية في `i` (قيمة الدورة الحالية).

4. طباعة النتيجة بعد انتهاء الحلقة:

◦ بعد أن تكمل الحلقة التكرار، نطبع قيمة `result`.

الكود:


```
let result = 1;

for (let i = 1; i <= 10; i++) {
  result *= i; // في كل دورة i ضرب النتيجة في
}

console.log("حاصل ضرب الأرقام من 1 إلى 10 هو", result);
```

الشرح:

- `let result = 1;` نبدأ بتعيين `result` إلى 1 لأننا نريد ضرب الأرقام معًا، و1 هو الحياد في الضرب.
- `for (let i = 1; i <= 10; i++)` نبدأ الحلقة من `i = 1` وتستمر حتى `i = 10`.
- `result *= i;` في كل دورة من الحلقة، نقوم بضرب `result` في قيمة `i` الحالية.
- `console.log(result);` بعد انتهاء الحلقة، نطبع النتيجة النهائية.

النتيجة عند التنفيذ:

حاصل ضرب الأرقام من 1 إلى 10 هو: 3628800

الشرح الرياضي:

- حاصل ضرب الأرقام من 1 إلى 10 هو:

$$1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 \times 9 \times 10 = 3628800$$
- هل تحتاج إلى مساعدة في شيء آخر؟ 😊



```
اجابه كذا صحيحه
let result = 1
for (let i = 1; i < 10; i++)
  console.log(i)
{
```



إجابتك تحتوي على بعض الأخطاء إذا كان هدفك حساب حاصل ضرب الأرقام من 1 إلى 10.

الملاحظات:

1. النطاق في الحلقة:

- في الكود الذي كتبته، تبدأ الحلقة من `i = 1` ولكنك قد نسيت تضمين الرقم 10 في النطاق. يجب أن يكون الشرط `i <= 10` بدلاً من `i < 10` لكي يتم تضمين 10 في التكرار.

2. عملية الضرب:

- في الكود الذي كتبته، تقوم فقط بطباعة قيمة `i` في كل دورة من الحلقة ولكنك لم تقم بحساب حاصل الضرب. يجب ضرب `result` في `i` في كل دورة، كما في الكود التالي.

الكود المعدل لحساب حاصل الضرب:

javascript

```
let result = 1;
for (let i = 1; i <= 10; i++) {
  result *= i; // ضرب النتيجة في i
}
console.log("حاصل ضرب الأرقام من 1 إلى 10 هو", result);
```

الشرح:

• `result *= i;`

- في كل دورة من الحلقة، تقوم بضرب قيمة `result` في قيمة `i` الحالية.

• `for (let i = 1; i <= 10; i++)`

- الحلقة تبدأ من `i = 1` وتستمر حتى `i = 10` (بما في ذلك 10).

• **النتيجة النهائية** هي `3628800` (حاصل ضرب الأرقام من 1 إلى 10).

إذا كنت ترغب في طباعة الأرقام أو القيام بشيء آخر أثناء الحساب، يمكننا تعديل الكود ليتناسب مع ذلك. أخبرني إذا كنت بحاجة لمزيد من التوضيح! 😊



اكتب ايه في حاصل ضرب الارقام



في حالة حساب حاصل ضرب الأرقام من 1 إلى 10، يجب عليك استخدام عملية ضرب عبر المتغير الذي نحتفظ فيه بالنتيجة. سنقوم بضرب الأرقام بشكل متتابع داخل الحلقة، والنتيجة النهائية هي حاصل ضرب جميع الأرقام. إليك كيفية كتابة النص في النتيجة:

الشرح:

إذا أردت أن تطبع النص الذي يعبر عن حاصل ضرب الأرقام بشكل مفصل (مثل كتابة كل عملية ضرب) يمكنك تعديل الكود كما يلي:

الكود:

javascript

```
let result = 1;
let text = "حاصل ضرب الأرقام من 1 إلى 10 هو ";

for (let i = 1; i <= 10; i++) {
  result *= i; // ضرب النتيجة في i
  if (i < 10) {
    text += i + " * "; // إضافة النص بين العمليات
  } else {
    text += i; // إضافة الرقم الأخير بدون "*" بعده
  }
}

text += " = " + result; // إضافة النتيجة النهائية
console.log(text); // طباعة النص
```

الشرح التفصيلي:

- `let result = 1;` نبدأ بـ `result`` وهو 1، لأننا نريد ضرب الأرقام معًا.
- `let text = "حاصل ضرب الأرقام من 1 إلى 10 هو ";` نبدأ بإنشاء النص الذي سنطبعه لشرح العملية.
- **داخل الحلقة:**
 - نقوم بضرب `result`` في قيمة `i`` في كل دورة.
 - نقوم بتحديث النص (`text``) بإضافة العمليات الحسابية (مثل `1 * 2 * 3 ...``).
- **الطباعة النهائية:**
 - بعد الانتهاء من العملية، نضيف النتيجة النهائية (`= 3628800``) إلى النص ونعرضه في النهاية.

النتيجة عند التنفيذ:

حاصل ضرب الأرقام من 1 إلى 10 هو: 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10 = 3628800

إذا كنت بحاجة إلى توضيح أكثر أو أي تعديل آخر، لا تتردد في سؤالني! 😊



```
let result = 1;
for (let i = 1; i < 10; i++) {
  result *= i;
}
console.log( result);
```

كده صح



الكود الذي كتبته قريب من الصحيح، ولكن هناك خطأ بسيط في النطاق الذي حددته للحلقة. الحلقة تبدأ من `i = 1` وتستمر حتى `i < 10`، مما يعني أنها ستنتهي عند `i = 9` ولن تشمل الرقم 10.

التعديل المطلوب:

لتشمل الأرقام من 1 إلى 10، يجب أن يكون الشرط في الحلقة هو `i <= 10` بدلاً من `i < 10`.

الكود المعدل:

javascript

```
let result = 1;
for (let i = 1; i <= 10; i++) { // تعديل هنا ليشمل 10
  result *= i;
}
console.log(result); // طباعة النتيجة
```

الشرح:

- `for (let i = 1; i <= 10; i++)`
 ◦ هذه الحلقة تبدأ من `i = 1` وتستمر حتى `i = 10` (بما في ذلك 10).
- `result *= i`;
 ◦ في كل دورة من الحلقة، يتم ضرب `result` في `i` وبالتالي جمع حاصل ضرب الأرقام من 1 إلى 10.

النتيجة:

3628800

التوضيح:

إذا كنت بحاجة إلى المزيد من الشرح أو التعديلات، لا تتردد في سؤالني! 😊



السؤال الثاني



إليك السؤال الثاني:

السؤال:

اكتب كودًا باستخدام `for loop` في JavaScript لطباعة كل رقم من 1 إلى 20 إذا كان الرقم فرديًا فقط.

انتظر إجابتي! 😊



```
for (let i = 0; i < 20; i++) {
  if (i % 2 === 1) {
    console.log(`${i} odd`);
  } else console.log(`${i} even`);
}
```



إجابتي قريبة من الصحيح، ولكن هناك بعض النقاط التي تحتاج إلى تعديل:

الملاحظات:

1. النطاق في الحلقة:

◦ إذا كنت تريد طباعة الأرقام من 1 إلى 20، يجب أن تبدأ من `i = 1` بدلاً من `i = 0`. الرقم 0 ليس فرديًا.

2. الطباعة:

◦ أنت استخدمت `add` بدلاً من `odd` للأرقام الفردية. من الأفضل استخدام `odd` لأن `odd` هو المصطلح الصحيح.

3. شرط التكرار:

◦ الحلقة تحتاج إلى أن تستمر حتى 20، لذا يجب أن يكون الشرط `i <= 20` بدلاً من `i < 20` إذا كنت تريد تضمين 20.

الكود المعدل:

```
for (let i = 1; i <= 20; i++) {
  if (i % 2 === 1) {
    console.log(`${i} odd`); // الأرقام الفردية
  } else {
    console.log(`${i} even`); // الأرقام الزوجية
  }
}
```

الشرح:

1. `for (let i = 1; i <= 20; i++)``

◦ تبدأ الحلقة من `1` وتستمر حتى `20` (بما في ذلك 20).

2. `if (i % 2 === 1)``

◦ يتحقق إذا كانت قيمة `i` فردية (أي أن باقي القسمة على 2 يساوي 1).

3. `console.log(`${i} odd`);``

◦ إذا كانت قيمة `i` فردية، يتم طباعة `i` مع النص `"odd"`.

4. `console.log(`${i} even`);``

◦ إذا كانت قيمة `i` زوجية، يتم طباعة `i` مع النص `"even"`.

النتيجة عند التنفيذ:

```
1 odd
2 even
3 odd
4 even
5 odd
6 even
7 odd
8 even
9 odd
10 even
11 odd
12 even
13 odd
14 even
15 odd
16 even
17 odd
18 even
19 odd
20 even
```

هل تحتاج إلى المزيد من التوضيح أو تعديل؟ 😊



السؤال الثاني



إليك السؤال الثاني:

السؤال:

اكتب كودًا باستخدام `for loop` في JavaScript لطباعة جميع الأرقام التي هي قسمة للعدد 5 بين 1 و 50. انتظر إجابتك! 😊



```
for (let i = 1; i < 51; i++){
  if (i % 5 === 0) {
    console.log(i)
  }else(console.log("no"))
}
```



إجابتك قريبة من الصحيح، ولكن هناك بعض النقاط التي يجب تعديلها:

الملاحظات:

1. الشرط في `else`:

- في جافا سكربت، لا يمكن استخدام `else` مع دالة `console.log` مباشرة. العبارة `else` يجب أن تحتوي على كتلة من الكود، وهي عادةً غير ضرورية هنا إذا كنت لا تريد طباعة `no`.

2. طباعة `no` للأرقام غير القابلة للقسمة على 5:

- إذا كانت الإجابة تقتصر فقط على طباعة الأرقام القابلة للقسمة على 5، فلا حاجة لاستخدام `else` هنا.

3. النطاق في الحلقة:

- الشرط `i < 51` صحيح لأنه يشمل الأرقام من 1 إلى 50.

الكود المعدل:

javascript

```
for (let i = 1; i <= 50; i++) {
  if (i % 5 === 0) {
    console.log(i); // طباعة الأرقام القابلة للقسمة على 5
  }
}
```

الشرح:

1. ``for (let i = 1; i <= 50; i++)``

○ تبدأ الحلقة من `1` وتستمر حتى `50` (بما في ذلك 50).

2. ``if (i % 5 === 0)``

○ يتحقق إذا كانت قيمة `i` قابلة للقسمة على 5 بدون باقي.

3. ``console.log(i)``

○ إذا كانت قيمة `i` قابلة للقسمة على 5، يتم طباعة `i`.

النتيجة عند التنفيذ:

```
5
10
15
20
25
30
35
40
45
50
```

إذا كنت تحتاج إلى المزيد من التوضيح أو التعديلات، لا تتردد في سؤالني! 😊



السؤال الثاني



إليك السؤال الثاني:

السؤال:

اكتب كودًا باستخدام ``for loop`` في JavaScript لطباعة الأرقام من 1 إلى 30، مع استثناء الأرقام القابلة للقسمة على 3.

انتظر إجابتك! 😊



```
for (let i = 1; i < 31; i++) {
  if (!(i % 3 === 0)) {
    console.log(i);
  }
}
```



```
} else console.log("no");
}
```



إجابتك تقريبًا صحيحة، لكن هناك بعض الملاحظات:

الملاحظات:

1. التعامل مع الأرقام غير القابلة للقسمة على 3:

- استخدمت `(i % 3 !== 0)` بشكل صحيح للتحقق من الأرقام التي ليست قابلة للقسمة على 3 (أي الأرقام التي لا يوجد بها باقي قسمة عند القسمة على 3).

2. طباعة `"no"`:

- لم يكن من الضروري طباعة `"no"` عند الأرقام التي قابلة للقسمة على 3 إذا كان المطلوب هو طباعة الأرقام فقط التي ليست قابلة للقسمة على 3.

3. إزالة `else`:

- لا حاجة لطباعة `"no"` عند الأرقام القابلة للقسمة على 3، لأن السؤال يطلب طباعة الأرقام فقط التي ليست قابلة للقسمة على 3.

الكود المعدل:

javascript

```
for (let i = 1; i <= 30; i++) {
  if (i % 3 !== 0) { // تقبل القسمة على 3
    console.log(i); // طباعة الأرقام التي لا تقبل القسمة على 3
  }
}
```

الشرح:

1. `for (let i = 1; i <= 30; i++)`:

- الحلقة تبدأ من `i = 1` وتستمر حتى `i = 30`.

2. `if (i % 3 !== 0)`:

- يتحقق من الأرقام التي لا تقبل القسمة على 3 (أي الأرقام التي باقي قسمة `i` على 3 لا يساوي 0).

3. `console.log(i)`:

- إذا كانت القيمة `i` غير قابلة للقسمة على 3، يتم طباعة `i`.

النتيجة عند التنفيذ:

```
1
2
4
5
7
8
10
11
13
14
16
17
19
20
22
23
25
26
28
29
```

هل تحتاج إلى المزيد من التوضيح أو المساعدة؟ 😊



السؤال الثاني



إليك السؤال الثاني:

السؤال:

اكتب كودًا باستخدام `for loop` في JavaScript لطباعة الأرقام من 1 إلى 50 فقط للأرقام الزوجية.

انتظر إجابتك! 😊



```
for (let i = 1; i < 51; i++) {
  if (i % 2 == 0) {
    console.log(i);
  }
}
```



إجابتك صحيحة! 😊

الشرح:

1. ``for (let i = 1; i < 51; i++)``

○ تبدأ الحلقة من ``i = 1`` وتستمر حتى ``i = 50``.

2. ``if (i % 2 == 0)``

○ يتحقق إذا كانت قيمة ``i`` زوجية (أي أن باقي قسمة ``i`` على 2 يساوي 0).

3. ``console.log(i)``

○ إذا كانت قيمة ``i`` زوجية، يتم طباعة ``i``.

النتيجة عند التنفيذ:

```
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
```

إجابة صحيحة! إذا كنت بحاجة إلى المزيد من التوضيح أو المساعدة، لا تتردد في سؤالني! 😊



السؤال الثاني



إليك السؤال الثاني:

السؤال:

اكتب كودًا باستخدام ``for loop`` في JavaScript لطباعة الأرقام من 1 إلى 100، ولكن إذا كان الرقم قابلاً للقسمة على 3 اطبع "Fizz"، وإذا كان قابلاً للقسمة على 5 اطبع "Buzz"، وإذا كان قابلاً للقسمة على 3 و 5 معًا اطبع "FizzBuzz".



```
for (let i = 1; i < 101; i++) {
  if (i % 3 === 0 && i % 5 === 0) {
    console.log(`${i} FizzBuzz `);
  } else if (i % 3 === 0) {
    console.log(`${i} Fizz`);
  } else if (i % 5 === 0) {
    console.log(`${i} Buzz`);
  }
}
```



إجابتك صحيحة تمامًا! 😊

الشرح:

1. `for (let i = 1; i < 101; i++)``

○ تبدأ الحلقة من `i = 1`` وتستمر حتى `i = 100`` (بما في ذلك 100).

2. `if (i % 3 === 0 && i % 5 === 0)``

○ يتحقق إذا كانت قيمة `i`` قابلة للقسمة على 3 و 5 معًا (أي إذا كانت النتيجة 0 عند القسمة على كل من 3 و 5).

3. `else if (i % 3 === 0)``

○ إذا كانت قيمة `i`` قابلة للقسمة على 3 فقط، يتم طباعة `"Fizz"``.

4. `else if (i % 5 === 0)``

○ إذا كانت قيمة `i`` قابلة للقسمة على 5 فقط، يتم طباعة `"Buzz"``.

النتيجة عند التنفيذ:

```
3 Fizz
5 Buzz
6 Fizz
9 Fizz
10 Buzz
12 Fizz
15 FizzBuzz
18 Fizz
20 Buzz
21 Fizz
24 Fizz
25 Buzz
27 Fizz
30 FizzBuzz
33 Fizz
35 Buzz
36 Fizz
39 Fizz
```

```

40 Buzz
42 Fizz
45 FizzBuzz
48 Fizz
50 Buzz
51 Fizz
54 Fizz
55 Buzz
57 Fizz
60 FizzBuzz
63 Fizz
65 Buzz
66 Fizz
69 Fizz
70 Buzz
72 Fizz
75 FizzBuzz
78 Fizz
80 Buzz
81 Fizz
84 Fizz
85 Buzz
87 Fizz
90 FizzBuzz
93 Fizz
95 Buzz
96 Fizz
99 Fizz

```

إجابة ممتازة! إذا كنت بحاجة إلى المزيد من المساعدة أو الأسئلة، لا تتردد في سؤالني! 😊



ايوه بحاجه الى الاسئله



بالطبع! إليك بعض الأسئلة لتدريبك على استخدام `for loop` في JavaScript:

السؤال 1:

اكتب كودًا باستخدام `for loop` في JavaScript لطباعة الأرقام من 1 إلى 20، ولكن يجب أن تطبع `"Prime"` للأرقام الأولية فقط.

السؤال 2:

اكتب كودًا باستخدام `for loop` في JavaScript لطباعة الأرقام الزوجية فقط بين 10 و 100.

السؤال 3:

اكتب كودًا باستخدام `for loop` في JavaScript لحساب مجموع الأرقام من 1 إلى 50.

السؤال 4:

اكتب كودًا باستخدام `for loop` في JavaScript لطباعة الأرقام من 1 إلى 100، ولكن إذا كان الرقم قابلاً للقسمة على 7 اطبع `"Lucky"`.

السؤال 5:

اكتب كودًا باستخدام `for loop` في JavaScript لطباعة السلسلة التالية: `"1, 4, 9, 16, 25..."` (أي مربعات الأعداد) حتى الرقم 100.

انتظر إجاباتك! 😊



السؤال الأول: for (let i = 1; i < 21; i++) {
;console.log(i)
}



الكود الذي كتبته صحيح لطباعة الأرقام من 1 إلى 20، ولكن السؤال يطلب طباعة `"Prime"` فقط للأرقام الأولية. لذا يجب تعديل الكود للتحقق مما إذا كان الرقم أوليًا أم لا.

الكود المعدل:

javascript

```
for (let i = 1; i <= 20; i++) {  
  let isPrime = true;  
  if (i === 1) {  
    isPrime = false; // ليس عددًا أوليًا  
  } else {  
    for (let j = 2; j <= Math.sqrt(i); j++) {  
      if (i % j === 0) {  
        isPrime = false; // كان يقبل القسمة على أي رقم آخر  
        break;  
      }  
    }  
  }  
  if (isPrime) {  
    console.log(`${i} Prime`);  
  }  
}
```

الشرح:

1. `for (let i = 1; i <= 20; i++)`

◦ تمر الحلقة على كل رقم بين 1 و 20.

2. `let isPrime = true`

◦ نفترض أن الرقم أولي إلى أن يثبت عكس ذلك.

3. `:`if (i === 1)``

○ الرقم 1 ليس أوليًا.

4. `:`for (let j = 2; j <= Math.sqrt(i); j++)``

○ نتحقق إذا كان الرقم يقبل القسمة على أي رقم آخر غير 1 و نفسه. إذا حدث ذلك، فإنه ليس أوليًا.

5. `:`if (isPrime)``

○ إذا بقيت قيمة `isPrime`` صحيحة، يتم طباعة أن الرقم أولي.

النتيجة عند التنفيذ:

mathematica

```
2 Prime
3 Prime
5 Prime
7 Prime
11 Prime
13 Prime
17 Prime
19 Prime
```

إذا كنت بحاجة إلى المزيد من الأسئلة أو المساعدة، لا تتردد في سؤالني! 😊



السؤال الاول: `for (let i = 1; i <= 20; i++)`

`;let isPrime = true`

`} if (i === 1)`

`;isPrime = false`

`} else {`

`{ } for (let j = 2; j <= Math.sqrt(i);)`

`} if (i % j == 0)`

`;isPrime = false`

`;break`

`{`

`{`

`{`



الكود الذي كتبته يحتوي على مشكلة بسيطة في وضعية الحلقة الداخلية التي تتحقق من كون الرقم أوليًا أم لا. لتصحيح الكود، يجب التأكد من كتابة حلقة التكرار الداخلية (`for``) بشكل صحيح وتشغيل الكود بطريقة منطقية.

الإصلاح المقترح:

```
for (let i = 1; i <= 20; i++) {
  let isPrime = true;

  if (i === 1) {
    isPrime = false; // الرقم 1 ليس عددًا أوليًا
  } else {
    for (let j = 2; j <= Math.sqrt(i); j++) { // حلقة التكرار الداخلية
      if (i % j === 0) { // إذا كان الرقم قابلاً للقسمة
        isPrime = false;
        break; // إنهاء الحلقة
      }
    }
  }

  if (isPrime) {
    console.log(`${i} Prime`);
  }
}
```

الملاحظات على الكود الذي كتبته:

1. الحلقة الداخلية:

◦ لم يتم كتابة شرط صحيح أو زيادات (`++j`) في حلقة التكرار الداخلية، مما يجعلها غير مكتملة.

2. `if (i % j == 0)` :

◦ هذه العبارة يجب أن تكون داخل الحلقة الداخلية `for`.

3. `break` :

◦ يجب استخدامه داخل الحلقة عندما يتم التأكد من أن الرقم ليس أوليًا.

النتيجة عند التنفيذ (بعد الإصلاح):

mathematica

```
2 Prime
3 Prime
5 Prime
7 Prime
11 Prime
13 Prime
17 Prime
19 Prime
```

إذا كنت بحاجة إلى المزيد من المساعدة، لا تتردد في طرح الأسئلة! 😊



السؤال الاول: `for (let i = 1; i <= 20; i++)`
`let isPrime = ture`


```

    } if (i === 1)
    ;isPrime = false
    } else {
  } for (let j = 2; j <= Math.sqrt(i); j++)
    ;isFinite = false
    ;break
    {
    {

    } if (isPrime)
;console.log(`${i} Prime`)
    {
    {

```



الكود الذي كتبته يحتوي على عدة أخطاء بسيطة تحتاج إلى تصحيح. دعنا نقوم بتحليل الكود وإصلاحه خطوة بخطوة:

الأخطاء في الكود:

1. خطأ في كتابة كلمة `true`:
 ◦ تم كتابتها على شكل `ture` بدلاً من `true`.
2. خطأ في اسم المتغير `isFinite`:
 ◦ يجب أن يكون `isPrime` لأنك تستخدمه للتحقق من الأعداد الأولية.
3. خطأ في زيادة قيمة `j` داخل الحلقة الداخلية:
 ◦ يجب كتابة `j++` في نهاية تعريف الحلقة الداخلية، وليس تركها فارغة.
4. خلط بين المتغيرات داخل الحلقة:
 ◦ استخدام المتغير `i` بدلاً من `j` في الشرط داخل الحلقة الداخلية خاطئ.

الكود المصحح:

javascript

```

for (let i = 1; i <= 20; i++) {
  let isPrime = true; // تصحيح كلمة true

  if (i === 1) {
    isPrime = false; // الرقم 1 ليس عددًا أوليًا
  } else {
    for (let j = 2; j <= Math.sqrt(i); j++) { // إصلاح الحلقة الداخلية
      if (i % j === 0) { // إذا كان الرقم قابلاً للقسمة
        isPrime = false;
        break; // إنهاء الحلقة
      }
    }
  }

  if (isPrime) {

```

```

    console.log(`${i} Prime`); // طباعة الرقم إذا كان أوليًا
  }
}

```

النتيجة عند التنفيذ (بعد الإصلاح):

mathematica

```

2 Prime
3 Prime
5 Prime
7 Prime
11 Prime
13 Prime
17 Prime
19 Prime

```

شرح الكود المصحح:

1. `for (let i = 1; i <= 20; i++)``
 - تبدأ الحلقة من الرقم 1 إلى الرقم 20.
2. `let isPrime = true``
 - يتم افتراض أن الرقم أولي إلى أن يتم إثبات العكس.
3. `if (i === 1)``
 - الرقم 1 ليس أوليًا، لذا يتم تعيين `isPrime = false``.
4. `الحلقة الداخلية for (let j = 2; j <= Math.sqrt(i); j++)``
 - تتحقق من قابلية الرقم `i`` للقسمة على الأرقام الأقل منه حتى الجذر التربيعي لـ `i``.
5. `if (i % j === 0)``
 - إذا كان الرقم يقبل القسمة على رقم آخر، يتم تعيين `isPrime = false`` وإيقاف الحلقة بـ `break``.
6. `if (isPrime)``
 - يتم طباعة الرقم إذا كان أوليًا.

إذا كنت بحاجة إلى المزيد من التوضيح أو أسئلة إضافية، لا تتردد في سؤالني! 😊



```

السؤال الثاني:
for (let i = 10; i < 101; i++)
  if (i % 2 === 0)
    console.log(i)
  {
    {

```



إجابتك للسؤال الثاني صحيحة! 🎉

الشرح:

1. `:`for (let i = 10; i < 101; i++)``
 - تبدأ الحلقة من الرقم 10 وتنتهي عند الرقم 100 (بما في ذلك الرقم 100).
2. `:`if (i % 2 === 0)``
 - الشرط يتحقق إذا كان الرقم زوجيًا (أي باقي القسمة على 2 يساوي 0).
3. `:`console.log(i)``
 - إذا كان الرقم زوجيًا، يتم طباعته.

النتيجة عند التنفيذ:

```
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
66
68
70
72
74
76
78
80
82
84
86
88
90
92
94
```

96
98
100

تحسين بسيط (اختياري):

إذا كنت تريد طباعة النتيجة بشكل أكثر وضوحًا مع رسالة:

javascript

```
for (let i = 10; i <= 100; i++) {
  if (i % 2 === 0) {
    console.log(`${i} is even`);
  }
}
```

إذا كنت بحاجة إلى المزيد من الأسئلة أو توضيح آخر، لا تتردد في سؤالني! 😊



السؤال الثالث: let sum = 1;
for (let i = 1; i < 51; i++)
 sum += 1
 {
 console.log(sum)



الكود الذي كتبته يحتوي على منطق غير صحيح إذا كنت تهدف إلى حساب مجموع الأرقام من 1 إلى 50. حاليًا، الكود يضيف الرقم الثابت 1 إلى المتغير `sum` في كل دورة من الحلقات، وليس الرقم الحالي `i`.

الإصلاح:

إذا كنت تريد حساب مجموع الأرقام من 1 إلى 50، يجب إضافة قيمة `i` إلى المتغير `sum` في كل دورة من الحلقات.

الكود المصحح:

javascript

```
let sum = 0; // بدء مجموع الأرقام من صفر
for (let i = 1; i < 51; i++) {
  sum += i; // إضافة قيمة i إلى المجموع
}
console.log(sum); // طباعة المجموع
```

الشرح:

1. `let sum = 0`;`

○ نبدأ بمجموع يساوي صفر.

2. `for (let i = 1; i < 51; i++)`;`

○ حلقة تكرار تبدأ من 1 وتنتهي عند الرقم 50.

3. `sum += i`;`

○ تضيف قيمة الرقم الحالي `i`` إلى المتغير `sum`` في كل دورة.

4. `console.log(sum)`;`

○ في النهاية، يتم طباعة مجموع الأرقام.

النتيجة عند التنفيذ:

إذا قمنا بتشغيل الكود المصحح، فإن النتيجة ستكون:

yaml

1275

وهي مجموع الأرقام من 1 إلى 50.

إذا كنت بحاجة إلى المزيد من الشرح أو الأسئلة، لا تتردد في طرحها! 😊



السؤال الثالث:
`let sum = 0;`
`for (let i = 1; i < 51; i++)`
`sum += i`
`{`
`console.log(sum)`
`}`



إجابتك للسؤال الثالث صحيحة تمامًا! 🎉

الشرح:

1. `let sum = 0`;`

○ يتم تعريف متغير `sum`` بقيمة أولية `0``، ليتم استخدامه لتخزين مجموع الأرقام.

2. `for (let i = 1; i < 51; i++)`;`

- تبدأ الحلقة من الرقم `1` وتنتهي عند الرقم `50` (يشمل 50 لأن الشرط هو `i < 51`).
- 3. `sum += i`;
- في كل دورة من الحلقة، يتم إضافة قيمة المتغير `i` إلى المتغير `sum`.
- 4. `console.log(sum)`;
- بعد انتهاء الحلقة، يتم طباعة القيمة النهائية لمجموع الأرقام.

النتيجة عند التنفيذ:

الكود يقوم بحساب مجموع الأرقام من 1 إلى 50، وستكون النتيجة:

yaml

1275

توضيح العملية الحسابية:

- المجموع = $50 + \dots + 3 + 2 + 1$
- يمكن التحقق من النتيجة باستخدام الصيغة الرياضية:

$$\frac{n \times (n + 1)}{2} = \text{Sum}$$

حيث $n = 50$.

$$1275 = \frac{(1 + 50) \times 50}{2} = \text{Sum}$$

إذا أردت المزيد من الأسئلة أو الشرح، لا تتردد في سؤالني! 😊



```
السؤال الرابع: for (let i = 1; i < 101; i++)
    if (i % 7 == 0)
        console.log(i)
    {
    }
```



إجابتك للسؤال الرابع صحيحة تمامًا! 🎉

الشرح:

1. `for (let i = 1; i < 101; i++)``

○ تبدأ الحلقة من الرقم `1` وتستمر حتى الرقم `100` (يشمل 100 لأن الشرط هو `i < 101`).

2. `if (i % 7 == 0)``

○ تتحقق من كون الرقم يقبل القسمة على 7 بدون باقي (أي أنه من مضاعفات الرقم 7).

3. `console.log(i)``

○ إذا تحقق الشرط، يتم طباعة الرقم.

النتيجة عند التنفيذ:

الكود سيطبع جميع الأرقام بين 1 و 100 التي هي مضاعفات الرقم 7:

```
7
14
21
28
35
42
49
56
63
70
77
84
91
98
```

تحسين بسيط (اختياري):

إذا أردت طباعة النتيجة مع رسالة توضيحية لكل رقم:

javascript

```
for (let i = 1; i < 101; i++) {
  if (i % 7 == 0) {
    console.log(`${i} is divisible by 7`);
  }
}
```

إذا أردت المزيد من الأسئلة أو المساعدة، فلا تتردد في التواصل! 😊



اسئلته الثانيه



السؤال الثاني:

اكتب كودًا باستخدام حلقة `for` لطباعة جميع الأرقام من 1 إلى 50 مع الإشارة إلى ما إذا كان الرقم "زوجي" (Even) أو "فردى" (Odd).

مثال للإجابة:

javascript

```
for (let i = 1; i <= 50; i++) {
  if (i % 2 === 0) {
    console.log(`${i} is Even`);
  } else {
    console.log(`${i} is Odd`);
  }
}
```

عندما تنتهي من الإجابة، أرسل الكود وسأخبرك إذا كان صحيحًا أم لا! 😊



```
for (let i = 1; i < 51; i++) {
  if (i % 2 == 0) {
    console.log(`${i} :even`);
  } else if (i % 2 == 1) {
    console.log(`${i} :odd`);
  }
}
```



إجابتك صحيحة تمامًا! 🎉

الشرح:

1. `for (let i = 1; i < 51; i++)`

○ تبدأ الحلقة من الرقم 1 وتستمر حتى الرقم 50 (لأن الشرط هو `i < 51`).

2. `if (i % 2 == 0)`

○ الشرط يتحقق إذا كان الرقم زوجيًا (باقي القسمة على 2 يساوي 0)، وفي هذه الحالة يتم طباعة `even: {i}`.

3. `else if (i % 2 == 1)`

○ الشرط يتحقق إذا كان الرقم فرديًا (باقي القسمة على 2 يساوي 1)، وفي هذه الحالة يتم طباعة `odd: {i}`.

4. `(...) console.log`

○ تطبع النتيجة مع تحديد إذا كان الرقم "even" أو "odd".

النتيجة عند التنفيذ:

الكود سيطبع:

ruby

```
1 :odd
2 :even
3 :odd
4 :even
5 :odd
6 :even
...
50 :even
```

ملاحظات إضافية:

- يمكن حذف شرط ``else if (i % 2 == 1)`` والاكتفاء بـ ``else``، لأن أي رقم غير زوجي سيكون فرديًا بالضرورة:

javascript

```
for (let i = 1; i < 51; i++) {
  if (i % 2 == 0) {
    console.log(`${i} :even`);
  } else {
    console.log(`${i} :odd`);
  }
}
```

إذا أردت المزيد من الأسئلة أو الأفكار، فلا تتردد في سؤالني! 😊



امثله على نظام ده في 1-loop- اكتب دالة تسمى countMultiples لحساب عدد الأرقام بين 1 و 20 وهي مضاعفات لرقم معين 5.

الإخراج:

الإخراج: 4

(شرح: مضاعفات العدد 5 بين 1 و 20 هي 5، 10، 15، 20).



إليك طريقة كتابة الدالة `countMultiples`` لحساب عدد الأرقام بين 1 و 20 التي هي مضاعفات لرقم معين:

javascript

```
function countMultiples(n) {
  let count = 0; // متغير لتخزين عدد المضاعفات

  for (let i = 1; i <= 20; i++) {
    if (i % n === 0) { // إذا كان i مضاعفًا للرقم n
      count++; // قم بزيادة العداد
    }
  }

  return count; // إعادة العدد النهائي
}

// استدعاء الدالة واختبارها
console.log(countMultiples(5)); // الإخراج: 4
```

شرح الكود:

1. `function countMultiples(n)`

○ دالة تستقبل رقمًا (`n`) كمعامل لتحديد المضاعفات المطلوبة.

2. `let count = 0`

○ متغير لعدد المضاعفات، يبدأ من الصفر.

3. `for (let i = 1; i <= 20; i++)`

○ حلقة تبدأ من الرقم 1 وتنتهي عند الرقم 20.

4. `if (i % n === 0)`

○ يتحقق إذا كان الرقم الحالي (`i`) مضاعفًا لـ `n` (أي باقي القسمة يساوي 0).

5. `count++`

○ إذا تحقق الشرط، يتم زيادة قيمة المتغير `count` بمقدار 1.

6. `return count`

○ تعيد الدالة عدد المضاعفات عند الانتهاء.

7. `console.log(countMultiples(5))`

○ يتم استدعاء الدالة مع الرقم 5، وطباعة الناتج.

الإخراج:

اختبار بأرقام أخرى:

اختبار الرقم 3:

javascript

```
console.log(countMultiples(3)); // الإخراج: 6
```

(المضاعفات: 3، 6، 9، 12، 15، 18)

اختبار الرقم 7:

javascript

```
console.log(countMultiples(7)); // الإخراج: 2
```

(المضاعفات: 7، 14)

إذا كنت تريد تعديل الكود أو إضافة تحسينات، فأخبرني! 😊



أبعتلك السؤال مع اجابته واريد منك ان تقول اجابه صحيحه ام خطأ



بالطبع، يمكنك إرسال السؤال مع الإجابة، وسأقوم بتقييم الإجابة لك! 😊



السؤال:3-اكتب دالة تسمى oddSummation لحساب مجموع الأعداد الفردية من 1 إلى 12 باستخدام الحلقة.



أرسل الإجابة الخاصة بك على هذا السؤال، وسأقوم بتقييم صحتها!



```
for (let i = 1; i < 13; i++) {  
  if (i % 2 == 1) {  
    console.log(`${i} :odd`);  
  }  
}
```

```
}
}
```



إجابتك صحيحة من حيث تحديد الأعداد الفردية (التي تحقق شرط القسمة على 2 بدون باقي) وطباعة هذه الأعداد، لكنك لم تحسب المجموع كما هو مطلوب في السؤال. السؤال يطلب حساب مجموع الأعداد الفردية، وليس فقط طباعة الأعداد الفردية.

الكود المصحح لحساب المجموع:

javascript

```
function oddSummation() {
  let sum = 0; // متغير لتخزين المجموع
  for (let i = 1; i <= 12; i++) {
    if (i % 2 == 1) { // إذا كان الرقم فرديًا
      sum += i; // إضافة الرقم إلى المجموع
    }
  }
  return sum; // إعادة المجموع النهائي
}

console.log(oddSummation()); // طباعة المجموع
```

الشرح:

- `let sum = 0` : تم تعريف متغير `sum` لتخزين المجموع.
- `if (i % 2 == 1)` : يتحقق إذا كان الرقم فرديًا.
- `sum += i` : إذا كان الرقم فرديًا، يتم إضافته إلى المجموع.
- `return sum` : في النهاية، يتم إرجاع المجموع.

الإخراج عند التنفيذ:

35

(لأن مجموع الأعداد الفردية بين 1 و 12 هو: $1 + 3 + 5 + 7 + 9 + 11 = 35$)

إذا كنت بحاجة إلى المزيد من المساعدة أو التوضيح، فلا تتردد في سؤالني! 😊



اسلئه عن while



بالطبع! إليك بعض الأسئلة باستخدام حلقة `while`، مع أمثلة على الإجابات إذا كنت بحاجة إلى المساعدة لاحقًا:

السؤال الأول:

اكتب كودًا باستخدام حلقة `while` لطباعة جميع الأرقام من 1 إلى 10.

السؤال الثاني:

اكتب كودًا باستخدام حلقة `while` لحساب مجموع الأرقام من 1 إلى 20.

السؤال الثالث:

اكتب كودًا باستخدام حلقة `while` لطباعة جميع الأرقام الزوجية بين 1 و 50.

السؤال الرابع:

اكتب كودًا باستخدام حلقة `while` لطباعة جدول الضرب للرقم 5 (أي 5×1 ، 5×2 ، ... حتى 5×10).

السؤال الخامس:

اكتب كودًا باستخدام حلقة `while` لطباعة الأعداد من 10 إلى 1 (تنازليًا).

اختر سؤالًا وأجب عليه، وسأقيّم إجابتك! 😊



```
السؤال الاول:
let i = 0;
while (i < 11) {
  console.log(i);
  ++i;
}
```



إجابتك صحيحة تمامًا! 🎉

شرح الكود:

1. `let i = 0;`

◦ يتم تعريف المتغير `i` بقيمة أولية `0`.

2. `while (i < 11)`

◦ حلقة تستمر في التنفيذ طالما أن قيمة `i` أقل من `11`.

3. `console.log(i)`؛

○ تطبع قيمة `i` الحالية في كل دورة.

4. `++i`؛

○ تزيد قيمة `i` بمقدار 1 في كل دورة، حتى تصل إلى 11، مما يؤدي إلى خروج الحلقة.

النتيجة عند التنفيذ:

الكود سيطبع الأرقام من 0 إلى 10:

```
0
1
2
3
4
5
6
7
8
9
10
```

ملاحظات:

- الكود مكتوب بطريقة صحيحة ويؤدي الغرض المطلوب.
- إذا أردت البدء من الرقم 1 بدلاً من 0، يمكن تعديل الكود إلى:

javascript

```
let i = 1;
while (i <= 10) {
  console.log(i);
  i++;
}
```

إذا أردت المزيد من الأسئلة أو شرح إضافي، فلا تتردد في طلب المساعدة! 😊



السؤال الثاني: `let sum = 1`؛

`let i = 0`؛

```
} while (i < 21)
;console.log(i)
;sum += i
```

```

    ;++i
    {

```



الكود الخاص بك يعمل بشكل صحيح إلى حد ما، لكنه يبدأ المتغير `sum` بقيمة 1 وليس 0، مما يجعل مجموع الأرقام بين 1 و 20 غير دقيق.

التصحيح:

1. قيمة `sum` الأولية:

○ يجب أن تبدأ بقيمة 0 لأنك تجمع الأرقام من البداية.

2. طباعة `i` و `sum`:

○ في حال أردت طباعة المجموع النهائي فقط (وليس طباعة كل قيمة لـ `i` في كل دورة)، يمكنك استخدام `console.log` بعد الحلقة.

الكود المصحح:

javascript

```

let sum = 0; // مجموع الأرقام يبدأ من 0
let i = 1;   // ابدأ من الرقم 1

while (i <= 20) { // استمر حتى الرقم 20
    sum += i;     // أضف القيمة الحالية لـ sum
    i++;         // بمقدار 1 زيادة قيمة i
}

console.log(sum); // طباعة المجموع النهائي

```

الإخراج:

210

(لأن مجموع الأرقام من 1 إلى 20 هو:
 $210 = 20 + \dots + 3 + 2 + 1$)

النسخة المعدلة من الكود إذا أردت طباعة كل خطوة:

```
let sum = 0;
let i = 1;

while (i <= 20) {
  console.log(`Adding ${i} to sum.`);
  sum += i;
  i++;
}

console.log(`Total sum: ${sum}`);
```

ملاحظات إضافية:

- إذا بدأت `i` من 0 كما في الكود الأول، ستضيف الصفر إلى المجموع، ولن يغير من الناتج لأنه لا يؤثر في الجمع.
- الكود المصحح يضمن البدء من 1 إذا كنت تريد جمع الأرقام الطبيعية.

هل تحتاج إلى تعديل إضافي أو تفسير آخر؟ 😊



```
السؤال الثالث: let i = 0;
    } while (i < 51)
    ;console.log(i)
    ;++i
    } if (i % 2 == 0)
    ;console.log(i)
    ;break
    {
    {
```



الكود الذي كتبته فيه بعض النقاط التي يمكن تحسينها أو تصحيحها لتكون الإجابة صحيحة وذات معنى واضح. دعني أشرح الكود خطوة بخطوة وأوضح المشكلة:

الكود كما هو مكتوب:

```
let i = 0;
while (i < 51) {
  console.log(i);
  i++;
  if (i % 2 == 0) {
    console.log(i);
    break;
  }
}
```



```
}
}
```

تحليل الكود:

1. `let i = 0` :
 ○ تعريف المتغير `i` وتهيئته بقيمة ابتدائية `0`.
2. `while (i < 51)` :
 ○ شرط الحلقة يستمر طالما أن `i` أقل من 51.
3. `console.log(i)` :
 ○ طباعة قيمة `i` الحالية.
4. `++i` :
 ○ زيادة قيمة `i` بمقدار 1.
5. `if (i % 2 == 0)` :
 ○ يتحقق إذا كانت قيمة `i` زوجية.
 ○ إذا تحقق الشرط، تتم طباعة `i` مرة أخرى وتُكسر الحلقة باستخدام `break`.

المشكلة:

- الحلقة ستتوقف فورًا عندما تجد أول رقم زوجي بعد القيمة الابتدائية.
- **نتيجة الكود:** دائمًا ستطبع الرقمين `0` و `2` فقط لأن أول رقم زوجي يتحقق عند `i = 2`، ثم تكسر الحلقة باستخدام `break`.

الإخراج عند التنفيذ:

```
0
2
```

ما الذي يمكن تحسينه؟

1. إذا كان الهدف طباعة جميع الأرقام الزوجية بين 0 و 50، لا حاجة إلى استخدام `break`.
2. الكود الحالي يكسر الحلقة بمجرد أن يجد أول رقم زوجي.

الكود المصحح لطباعة جميع الأرقام الزوجية بين 0 و50:

javascript

```
let i = 0;
while (i <= 50) { // استمر حتى 50
  if (i % 2 == 0) { // تحقق إذا كان الرقم زوجيًا
    console.log(i); // طباعة الرقم الزوجي
  }
  i++; // زيادة المتغير i
}
```

الإخراج بعد التصحيح:

python

```
0
2
4
6
8
...
50
```

ملاحظات إضافية:

- إذا كان المطلوب إبقاء `break` ضمن الكود، يمكن تعديل المنطق ليكون أكثر وضوحًا حسب الغرض المطلوب.
- إذا كان لديك استفسار إضافي أو هدف مختلف من الكود، يمكنك توضيحه وسأساعدك في تحقيقه! 😊



السؤال الثالث: let i = 0;

```

  } while (i < 51)
;console.log(i)
  ;++i
  } if (i % 2 == 0)
;console.log(i)

{
{
```



تحليل الكود الخاص بك:

الكود يقوم بطباعة القيم من `i` داخل حلقة `while`، مع شرط إضافي داخل الحلقة يتحقق مما إذا كانت القيمة الحالية لـ `i` زوجية. عند تحقق هذا الشرط، تتم طباعة الرقم مرة أخرى.

شرح الكود خطوة بخطوة:

1. `let i = 0`؛

◦ تعريف المتغير `i` وتهيئته بالقيمة الابتدائية `0`.

2. `while (i < 51)`؛

◦ حلقة تستمر طالما أن `i` أقل من 51.

3. `console.log(i)`؛

◦ طباعة قيمة `i` الحالية.

4. `++i`؛

◦ زيادة قيمة `i` بمقدار 1 بعد طباعة القيمة الحالية.

5. `if (i % 2 == 0)`؛

◦ التحقق مما إذا كانت قيمة `i` زوجية.

◦ إذا كانت القيمة زوجية، يتم طباعة `i` مرة أخرى.

الإخراج المتوقع:

سيطبع الكود كل قيمة من `i` مرتين إذا كانت زوجية (مرة في البداية، ومرة داخل الشرط). مثال للإخراج:

python

```
0
1
2
2
3
4
4
...
50
50
```

ملاحظات:

- الكود يعمل، ولكنه لا يبدو منطقيًا إذا كنت تريد طباعة الأرقام بشكل منظم.
- إذا كنت تريد طباعة جميع الأرقام الزوجية فقط بين 0 و50، يمكن تعديل الكود ليكون أكثر وضوحًا.

الكود المُحسَّن لطباعة الأرقام الزوجية فقط:

```
let i = 0;
while (i <= 50) {
  if (i % 2 == 0) { // تحقق إذا كان الرقم زوجيًا
    console.log(i); // طباعة الرقم الزوجي
  }
  i++; // زيادة قيمة i
}
```

الإخراج:

python

```
0
2
4
6
8
...
50
```

إذا كنت تقصد شيئاً آخر بالكود، يمكنك توضيحه وسأساعدك! 😊



السؤال الثالث: let i = 0;
while (i < 51)

```
} if (i % 2 == 0)
;console.log(i)
; ++i
{
{
```



الكود الذي كتبته يحتوي على خطأ بسيط:

في الكود الحالي، يتم زيادة قيمة `i` فقط داخل شرط `if`، مما يؤدي إلى مشكلة **حلقة لانهاية** عندما يكون الرقم `i` فرديًا. والسبب أن `i` لن يتغير إذا لم يكن الشرط (`i % 2 == 0`) محققًا، مما يجعل الحلقة تستمر إلى الأبد.

التصحيح:

يجب زيادة قيمة `i` خارج شرط `if` لضمان أن الحلقة تستمر بشكل طبيعي حتى تحقق الشرط بالكامل.

javascript

```
let i = 0;
while (i < 51) {
  if (i % 2 == 0) {
    console.log(i); // طباعة الأرقام الزوجية فقط
  }
  i++; // في كل دورة i زيادة قيمة
}
```

شرح الكود المصحح:

1. ``;let i = 0``

◦ يتم تعريف المتغير ``i`` بقيمة ابتدائية ``0``.

2. ``while (i < 51)``

◦ تستمر الحلقة طالما أن ``i`` أقل من 51.

3. ``if (i % 2 == 0)``

◦ يتم التحقق إذا كانت قيمة ``i`` زوجية (أي تقبل القسمة على 2 بدون باقي).

4. ``;console.log(i)``

◦ إذا تحقق الشرط، تتم طباعة قيمة ``i``.

5. ``;++i``

◦ يتم زيادة قيمة ``i`` بعد كل دورة، سواء تحقق شرط الزوجية أم لا، لضمان تقدم الحلقة.

الإخراج المتوقع:

python

```
0
2
4
6
8
10
...
50
```

ملاحظات إضافية:

- إذا كنت تريد طباعة الأرقام الزوجية فقط من 1 إلى 50، يمكنك بدء ``i`` من 1 بدلاً من 0.

- إذا كنت بحاجة إلى شرح إضافي أو تحسينات، أخبرني! 😊



حل السؤال الثالث



حل السؤال الثالث:

الكود المطلوب طباعة الأرقام الزوجية من 0 إلى 50 باستخدام حلقة `while`. الكود الذي قدمته به خطأ لأنه يزيد قيمة `i` فقط داخل شرط `if`، مما يؤدي إلى حلقة لانهائية إذا لم يتحقق الشرط.

الكود المصحح:

javascript

```
let i = 0; // قيمة ابتدائية i تعريف
while (i < 51) { // الحلقة تعمل طالما i أقل من 51
  if (i % 2 == 0) { // التحقق إذا كان زوجيًا
    console.log(i); // طباعة الأرقام الزوجية
  }
  i++; // بعد كل دورة i زيادة
}
```

شرح الكود:

1. `let i = 0;` تعريف المتغير `i` وتهيئته بالقيمة 0 كبداءة.
2. `while (i < 51)` حلقة تستمر طالما أن قيمة `i` أقل من 51.
3. `if (i % 2 == 0)` التحقق مما إذا كانت قيمة `i` تقبل القسمة على 2 بدون باقي (أي زوجية).
4. `console.log(i)` إذا كان الشرط صحيحًا، تتم طباعة الرقم الزوجي.
5. `i++` زيادة قيمة `i` بمقدار 1 في كل دورة، لضمان تقدم الحلقة.

الإخراج المتوقع:

```

0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50

```

ملحوظة:

- زيادة المتغير `i` يجب أن تكون خارج شرط `if` لضمان أن الحلقة تتقدم بشكل صحيح.
- إذا أردت تنفيذ أي تعديل أو تحتاج إلى شرح إضافي، لا تتردد في طلب المساعدة! 😊



```

السؤال الخامس: let i = 10;
                    } while (i >= 1)
                    ;console.log(i)
                    ;--i
                    {

```



تحليل الكود:

الكود المكتوب:

javascript

```

let i = 10; // تعريف المتغير i بقيمة 10
while (i >= 1) { // الحلقة تعمل طالما أن i أكبر من أو يساوي 1
  console.log(i); // طباعة قيمة i الحالية
}

```

```
i--; // تقليل قيمة i
}
```

شرح الكود خطوة بخطوة:

1. `let i = 10;`

◦ يتم تعريف المتغير `i` بقيمة ابتدائية 10.

2. `while (i >= 1)`

◦ تستمر الحلقة طالما أن قيمة `i` أكبر من أو تساوي 1.

3. `console.log(i);`

◦ يتم طباعة قيمة `i` الحالية في كل دورة.

4. `i--;`

◦ يتم تقليل قيمة `i` بمقدار 1 بعد كل دورة.

الإخراج المتوقع:

الكود يقوم بطباعة الأرقام من 10 إلى 1 تنازليًا:

```
10
9
8
7
6
5
4
3
2
1
```

التأكد من صحة الكود:

✓ الكود صحيح ويعمل بشكل مثالي لعرض الأرقام من 10 إلى 1 باستخدام حلقة `while`.

ملاحظات إضافية:

- إذا أردت عرض الأرقام تصاعديًا من 1 إلى 10 بدلًا من تنازليًا، يمكن تغيير بداية المتغير `i` إلى 1 وتغيير شرط الحلقة وزيادة `i` في كل دورة.





أسئلة إضافية عن استخدام `while` loop:

السؤال الأول:

اكتب كود باستخدام حلقة `while` لطباعة الأرقام من 1 إلى 20، مع كتابة "even" بجانب الأرقام الزوجية و"odd" بجانب الأرقام الفردية.

الإجابة المتوقعة:

javascript

```
let i = 1;
while (i <= 20) {
  if (i % 2 === 0) {
    console.log(`${i} even`);
  } else {
    console.log(`${i} odd`);
  }
  i++;
}
```

السؤال الثاني:

اكتب كودًا باستخدام `while` لحساب مجموع الأعداد من 1 إلى 100.

الإجابة المتوقعة:

javascript

```
let i = 1;
let sum = 0;
while (i <= 100) {
  sum += i;
  i++;
}
console.log(`The sum is: ${sum}`);
```

السؤال الثالث:

اكتب كودًا باستخدام `while` لطباعة مضاعفات الرقم 3 بين 1 و30.

javascript

```
let i = 1;
while (i <= 30) {
  if (i % 3 === 0) {
    console.log(i);
  }
  i++;
}
```

السؤال الرابع:

اكتب كودًا باستخدام `while` لطباعة الأرقام من 20 إلى 1 تنازليًا.

الإجابة المتوقعة:

javascript

```
let i = 20;
while (i >= 1) {
  console.log(i);
  i--;
}
```

السؤال الخامس:

اكتب كودًا باستخدام `while` لطباعة جميع الأرقام بين 1 و50 التي تكون مضاعفات لكل من 3 و5.

الإجابة المتوقعة:

javascript

```
let i = 1;
while (i <= 50) {
  if (i % 3 === 0 && i % 5 === 0) {
    console.log(i);
  }
  i++;
}
```

إذا أردت المزيد من الأسئلة أو الحلول، لا تتردد في طلبها! 😊



```

السؤال الاول: let i = 0;
                } while (i < 21)
                } if (i % 2 == 1)
;console.log(`${i} :even`)
                {
                ;++i
                {

```



مراجعة الكود:

الكود المكتوب:

javascript

```

let i = 0;
while (i < 21) {
  if (i % 2 == 1) {
    console.log(`${i} :even`);
  }
  i++;
}

```

التحليل:

الكود يحتوي على **خطأ منطقي** في النص الذي يتم طباعته. الشرط الذي يتحقق منه هو إذا كانت القيمة **فردية** ($i \% 2 == 1$)، ولكن النص المطبوع يشير إلى أنها **زوجية** (``even``).

الإصلاح:

لتكون الطباعة صحيحة، يجب تغيير النص الذي يتم طباعته إلى ``odd`` بدلاً من ``even``. الكود المصحح:

javascript

```

let i = 0;
while (i < 21) {
  if (i % 2 == 1) {
    console.log(`${i} :odd`);
  }
  i++;
}

```

الإخراج المتوقع بعد التصحيح:

سيتم طباعة الأرقام الفردية بين 0 و20 مع النص "odd"، مثل:

ruby

```

1 :odd
3 :odd
5 :odd
7 :odd
9 :odd
11 :odd
13 :odd
15 :odd
17 :odd
19 :odd

```

ملحوظة:

إذا أردت طباعة الأرقام الزوجية بدلاً من الفردية، يمكنك تعديل الشرط ليكون:

javascript

```

if (i % 2 == 0) {
  console.log(`${i} :even`);
}

```

هل لديك أسئلة إضافية أو تعديلات أخرى؟ 😊



السؤال الثالث: let sum = 0;

```

;let i = 1
} while (i < 101)
;console.log(i)
;sum += i
; ++i
{

```



تحليل الكود:

الكود المكتوب:

javascript

```

let sum = 0;
let i = 1;
while (i < 101) {
  console.log(i); // طباعة قيمة i
  sum += i; // إضافة قيمة i إلى المتغير sum
}

```

```
i++; // زيادة i بمقدار 1
}
```

شرح الكود:

```
`;let sum = 0`.1
```

○ تعريف المتغير `sum` وتعيين قيمته الابتدائية 0، لتخزين مجموع الأرقام.

```
`;let i = 1`.2
```

○ تعريف المتغير `i` وتهيئته بالقيمة 1 كبدائية.

```
`while (i < 101)`.3
```

○ حلقة `while` تستمر طالما أن قيمة `i` أقل من 101.

```
`;console.log(i)`.4
```

○ في كل دورة من الحلقة، يتم طباعة قيمة `i` الحالية.

```
`;sum += i`.5
```

○ إضافة قيمة `i` إلى المتغير `sum` في كل مرة.

```
`;++i`.6
```

○ زيادة قيمة `i` بمقدار 1 في كل دورة.

الإخراج المتوقع:

سيتم طباعة الأرقام من 1 إلى 100 في كل دورة، بينما سيتم إضافة كل قيمة من هذه الأرقام إلى المتغير `sum`.
الإخراج المبدئي (بدائية):

python

```
1
2
3
...
100
```

وبالنهاية، سيتم حساب مجموع الأرقام من 1 إلى 100.

كيفية طباعة المجموع النهائي:

إذا كنت ترغب في طباعة المجموع النهائي للأرقام، يجب إضافة طباعة `sum` بعد إتمام الحلقة:

```
console.log(`The sum is: ${sum}`);
```

الإخراج النهائي:

python

```
1
2
3
...
100
The sum is: 5050
```

التفسير:

- المجموع النهائي هو مجموع الأرقام من 1 إلى 100، والذي يساوي 5050.

هل تحتاج إلى مزيد من الشرح أو تعديل؟ 😊



كتابه السؤال السؤال



بالطبع! إليك السؤال مع الكود الخاص به:

السؤال الثالث:

اكتب كودًا باستخدام حلقة `while` لحساب مجموع الأرقام من 1 إلى 100، وطباعة كل رقم أثناء حسابه.

الإجابة:

javascript

```
let sum = 0;
let i = 1;
while (i < 101) {
  console.log(i); // في كل دورة i طباعة قيمة
  sum += i;       // إلى المتغير i إضافة قيمة
  i++;           // بمقدار 1 زيادة
}
console.log(`The sum is: ${sum}`); // طباعة المجموع النهائي
```

python

```
1
2
3
...
100
The sum is: 5050
```

إذا كنت بحاجة إلى أي مساعدة إضافية أو تعديل آخر، لا تتردد في سؤالني! 😊



السؤال ثم استنى لما ابعتلك نتجيه



تمام! إليك السؤال:

السؤال:

اكتب كودًا باستخدام حلقة `while` لحساب مجموع الأرقام من 1 إلى 100، وطباعة كل رقم أثناء حسابه.

انتظر نتيجتك الآن! 😊



```
let sum = 0;
let i = 1;
while (i < 101) {
  console.log(i);
  sum += i;
  i++;
  console.log(`the is sum:${sum}`);
}
```



تحليل الكود:

الكود الذي قدمته:

```
let sum = 0;
let i = 1;
while (i < 101) {
  console.log(i); // في كل دورة i طباعة قيمة
  sum += i; // إضافة قيمة i إلى المتغير sum
  i++; // بمقدار 1 زيادة i
  console.log(`the is sum:${sum}`); // طباعة المجموع بعد إضافة كل قيمة
}
```

الملاحظات:

- الكود يعمل بشكل صحيح، ولكنك تطبع المجموع بعد إضافة كل قيمة في نفس الدورة. لذا ستطبع المجموع بعد كل إضافة، وسيكون من الأفضل طباعة المجموع بعد إتمام الحلقة بالكامل.

الإصلاح المقترح:

لإظهار المجموع بعد الانتهاء من جميع العمليات:

```
let sum = 0;
let i = 1;
while (i < 101) {
  console.log(i); // في كل دورة i طباعة قيمة
  sum += i; // إضافة قيمة i إلى المتغير sum
  i++; // بمقدار 1 زيادة i
}
console.log(`The final sum is: ${sum}`); // طباعة المجموع النهائي بعد اكتمال الحلقة
```

الإخراج المتوقع بعد التصحيح:

```
1
2
3
...
100
The final sum is: 5050
```

ملاحظات:

- إذا كنت ترغب في طباعة المجموع بشكل تدريجي داخل الحلقة، فإن كودك سيعمل بشكل صحيح، لكن عادةً ما يفضل طباعة النتيجة النهائية بعد اكتمال الحسابات.



```
let sum = 0;
let i = 1;
while (i < 101) {
  console.log(i);
  sum += i;
  i++;
}
```

كدا الافضل}



نعم، هذا الكود أفضل! لأنك الآن تقوم بحساب المجموع بشكل تدريجي، ولكنك فقط تطبع الأرقام دون طباعة المجموع في كل دورة، مما يجعل الكود أكثر وضوحًا.

الكود:

javascript

```
let sum = 0;
let i = 1;
while (i < 101) {
  console.log(i); // في كل دورة i طباعة قيمة
  sum += i;       // sum إلى المتغير i إضافة قيمة
  i++;           // بمقدار 1 زيادة
}
console.log(`The final sum is: ${sum}`); // طباعة المجموع النهائي بعد اكتمال الحلقة
```

الإخراج المتوقع:

سيتم طباعة الأرقام من 1 إلى 100، وفي النهاية سيعرض المجموع النهائي:

kotlin

```
1
2
3
...
100
The final sum is: 5050
```

ملاحظات:

- الكود الآن واضح، ويطبع الأرقام من 1 إلى 100.
- المجموع النهائي يتم طباعته بعد الانتهاء من جميع الحسابات.

إذا كنت بحاجة لمساعدة إضافية، لا تتردد في سؤالني! 😊



السؤال الثاني



تمام! إليك السؤال الثاني:

السؤال الثاني:

اكتب كودًا باستخدام حلقة `while` لطباعة الأرقام من 10 إلى 1 تنازليًا.

انتظر نتيجتك! 😊



```
let i = 10;
while (i >= 1) {
  console.log(i);
  i++;
}
```



الكود الذي كتبته يحتوي على **خطأ صغير** في كيفية زيادة قيمة `i`. بما أنك تريد طباعة الأرقام من 10 إلى 1 تنازليًا، يجب أن يتم تقليل قيمة `i` وليس زيادتها.

الخطأ:

javascript

```
i++; // هذا سيزيد قيمة i بدلاً من تقليلها
```

الإصلاح:

يجب عليك تغيير `++i` إلى `--i` لتقليل قيمة `i` في كل دورة:

الكود المصحح:

javascript

```
let i = 10;
while (i >= 1) {
```

```
console.log(i);
i--; // تقليل قيمة i
}
```

الإخراج المتوقع:

سيتم طباعة الأرقام من 10 إلى 1:

```
10
9
8
7
6
5
4
3
2
1
```

هل تحتاج إلى مزيد من التوضيح أو التعديلات؟ 😊



السؤال الثاني



السؤال الثاني:

اكتب كودًا باستخدام حلقة `while` لحساب مجموع الأرقام الزوجية من 1 إلى 50، ثم قم بطباعة الناتج النهائي فقط.

انتظر إجابتك! 😊



```
let sum = 0;
let i = 1;
while (i < 51) {
  if (i % 2 == 0) {
    console.log(i);
    sum += i;
  }
  i++;
}
console.log(`this is sum ${sum}`);
```



الكود الذي كتبتة صحيح تمامًا! 🏆

تحليل الكود:

javascript

```
let sum = 0; // تعريف متغير لحفظ المجموع
let i = 1; // بداية العد من 1
while (i < 51) { // استمرار الحلقة حتى يصل i إلى 50
  if (i % 2 == 0) { // التحقق إذا كان الرقم زوجيًا
    console.log(i); // طباعة الرقم الزوجي
    sum += i; // إضافة الرقم الزوجي إلى المجموع
  }
  i++; // في كل دورة i زيادة قيمة
}
console.log(`this is sum ${sum}`); // طباعة المجموع النهائي
```

الإخراج المتوقع:

1. سيتم طباعة جميع الأرقام الزوجية من 1 إلى 50.
2. سيتم طباعة المجموع النهائي لهذه الأرقام.

الإخراج:

kotlin

```
2
4
6
8
...
50
this is sum 650
```

تعليق:

- الكود مكتوب بطريقة صحيحة ومنظمة.
- إذا كنت تريد تحسين الكود أو لديك أي استفسار إضافي، أنا هنا للمساعدة! 😊



السؤال الثاني



السؤال الثاني:

اكتب كودًا باستخدام حلقة `while` لطباعة جميع الأرقام التي تقبل القسمة على 3 بين 1 و30، ثم قم بحساب مجموع هذه الأرقام وطباعته بعد انتهاء الحلقة.

انتظر إجابتك! 😊



```
let sum = 0;
let i = 1;
while (i < 31) {
  if (i % 3 == 1) {
    console.log(i);
    sum += i;
  }
  i++;
}
console.log(`this is sum${i}`);
```



الكود يحتوي على خطأين صغيرين:

1. شرط القسمة:

javascript

```
if (i % 3 == 1)
```

هذا الشرط غير صحيح. للتحقق من أن الرقم يقبل القسمة على 3، يجب أن يكون باقي القسمة مساوياً لـ 0 وليس 1. الإصحاح:

javascript

```
if (i % 3 == 0)
```

2. طباعة المجموع النهائي: في السطر التالي:

javascript

```
console.log(`this is sum${i}`);
```

يتم طباعة قيمة `i` بدلاً من `sum`. يجب طباعة `sum`، لأنه يحتوي على المجموع النهائي.

javascript

```
console.log(`this is sum ${sum}`);
```

الكود المصحح:

javascript

```
let sum = 0;
let i = 1;
while (i < 31) {
  if (i % 3 == 0) { // تحقق إذا كان الرقم يقبل القسمة على 3
    console.log(i); // طباعة الرقم
    sum += i; // إضافة الرقم إلى المجموع
  }
  i++;
}
console.log(`this is sum ${sum}`); // طباعة المجموع النهائي
```

الإخراج المتوقع:

سيتم طباعة جميع الأرقام التي تقبل القسمة على 3 بين 1 و30، بالإضافة إلى المجموع النهائي:

kotlin

```
3
6
9
12
15
18
21
24
27
30
this is sum 165
```

هل تحتاج إلى أي تعديلات إضافية أو شرح مفصل أكثر؟ 😊



السؤال الثاني



السؤال الثاني:

اكتب كودًا باستخدام حلقة `while` لطباعة جميع الأرقام بين 1 و50 التي تقبل القسمة على 5، ثم قم بحساب مجموع هذه الأرقام وطباعته بعد انتهاء الحلقة.

انتظر إجابتك! 😊



```
let sum = 0;
let i = 1;
while (i < 51) {
  if (i % 5 == 0) {
    console.log(i);
    sum += i;
  }
  i++;
}
console.log(`this is sum:${sum}`);
```



الكود صحيح تمامًا! ✓

تحليل الكود:

javascript

```
let sum = 0; // تعريف المتغير لتخزين المجموع
let i = 1; // بدء العد من الرقم 1
while (i < 51) { // استمرار الحلقة حتى يصل إلى 50
  if (i % 5 == 0) { // التحقق إذا كان الرقم يقبل القسمة على 5
    console.log(i); // طباعة الرقم
    sum += i; // إضافة الرقم إلى المجموع
  }
  i++; // في كل دورة i زيادة
}
console.log(`this is sum:${sum}`); // طباعة المجموع النهائي
```

الإخراج المتوقع:

سيتم طباعة جميع الأرقام التي تقبل القسمة على 5 بين 1 و50، بالإضافة إلى المجموع النهائي:

kotlin

```
5
10
15
20
```

```
25
30
35
40
45
50
this is sum:275
```

تعليق:

- الكود مكتوب بشكل صحيح ومنظم.
- إذا كنت بحاجة إلى أي مساعدة إضافية أو لديك أسئلة أخرى، فأنا هنا لمساعدتك! 😊



السؤال الثاني



السؤال الثاني:

اكتب كودًا باستخدام حلقة `while` لطباعة جميع الأرقام الفردية من 1 إلى 30، ثم قم بحساب مجموع هذه الأرقام وطباعته بعد انتهاء الحلقة.

انتظر إجابتك! 😊



```
let sum = 0;
let i = 1;
while (i < 31) {
  if (i % 2 == 1) {
    console.log(i);
    sum += i;
  }
  i++;
}
console.log(`this is sum ${sum}`);
```



الكود الذي كتبته صحيح تمامًا! ✓

تحليل الكود:

1. التأكد من فردية الرقم:

javascript

```
if (i % 2 == 1)
```

هذا الشرط يتحقق إذا كان الرقم فرديًا.
2. إضافة الرقم الفردي إلى المجموع:

javascript

```
sum += i;
```

3. زيادة قيمة i:

javascript

```
i++;
```

هذه الزيادة تضمن تقدم الحلقة حتى الوصول إلى الرقم 30.
4. طباعة المجموع النهائي:

javascript

```
console.log(`this is sum ${sum}`);
```

الكود الكامل:

javascript

```
let sum = 0;
let i = 1;
while (i < 31) {
  if (i % 2 == 1) { // تحقق إذا كان الرقم فرديًا
    console.log(i); // طباعة الرقم الفردي
    sum += i; // إضافته إلى المجموع
  }
  i++; // زيادة قيمة i
}
console.log(`this is sum ${sum}`); // طباعة المجموع النهائي
```

الإخراج المتوقع:

سيتم طباعة جميع الأرقام الفردية من 1 إلى 30، يليها المجموع النهائي:

```
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
this is sum 225
```

تعليق:

- الكود مكتوب بشكل صحيح ومنظم.
- إذا كنت بحاجة إلى أي استفسار إضافي أو تحسينات، فأنا هنا لمساعدتك! 😊



السؤال الثاني



السؤال الثاني:

اكتب كودًا باستخدام حلقة `while` لطباعة جميع الأرقام التي تقبل القسمة على 4 بين 1 و40، ثم قم بحساب مجموع هذه الأرقام وطباعته بعد انتهاء الحلقة.

انتظر إجابتك! 😊



```
let sum = 0;
let i = 1;
while (i < 41) {
  if (i % 4 == 0) {
    console.log(i);
    sum += i;
  }
  i++;
}
console.log(`this is sum ${sum}`);
```



الكود صحيح تمامًا! ✓

تحليل الكود:

1. شرط القسمة على 4:

```
javascript

if (i % 4 == 0)
```

يتحقق هذا الشرط إذا كان الرقم يقبل القسمة على 4 بدون باقي.

2. إضافة الرقم إلى المجموع:

```
javascript

sum += i;
```

يتم إضافة الأرقام التي تحقق الشرط إلى المتغير `sum`.

3. زيادة قيمة `i`:

```
javascript

i++;
```

هذا يضمن تقدم الحلقة حتى الوصول إلى الرقم 40.

4. طباعة المجموع النهائي:

```
javascript

console.log(`this is sum ${sum}`);
```

الكود الكامل:

```
javascript

let sum = 0;
let i = 1;
while (i < 41) {
  if (i % 4 == 0) { // تحقق إذا كان الرقم يقبل القسمة على 4
    console.log(i); // طباعة الرقم
    sum += i; // إضافته إلى المجموع
  }
  i++; // زيادة قيمة i
}
```

```
}
console.log(`this is sum ${sum}`); // طباعة المجموع النهائي
```

الإخراج المتوقع:

سيتم طباعة جميع الأرقام التي تقبل القسمة على 4 بين 1 و40، يليها المجموع النهائي:

kotlin

```
4
8
12
16
20
24
28
32
36
40
this is sum 220
```

تعليق:

- الكود مكتوب بشكل دقيق وممتاز.
- إذا كنت بحاجة إلى استفسارات إضافية أو أفكار حول حلقات أخرى، فأنا هنا لمساعدتك! 😊



السؤال الثاني



السؤال الثاني:

اكتب كودًا باستخدام حلقة `while` لحساب مجموع الأرقام الفردية بين 1 و50، ثم قم بطباعة الناتج النهائي.

انتظر إجابتك! 😊



عايزه ااكار ضامه for while



أفكار لتمارين تجمع بين for و while:

1. حساب مجموع الأعداد بين 1 و100 باستخدام `for` ثم إيقاف العملية باستخدام `while`

- في البداية، استخدم حلقة `for` لحساب المجموع.
- ثم، استخدم `while` للتحقق إذا كان المجموع أكبر من 500، وإذا كان كذلك، قم بإيقاف الحلقة.

2. إيجاد الأرقام الأولية بين 1 و100 باستخدام `for` ثم استخدام `while` للتحقق من قابلية القسمة

- استخدم حلقة `for` للتكرار على الأعداد من 1 إلى 100.
- داخل الحلقة، استخدم حلقة `while` للتحقق إذا كان العدد يقبل القسمة على أي عدد من 2 إلى الجذر التربيعي للعدد.
- إذا لم يقبل القسمة على أي من هذه الأعداد، فهذا العدد أولي.

3. طباعة تسلسل فيبوناتشي باستخدام `for` وحساب المجموع باستخدام `while`

- في البداية، استخدم `for` لطباعة تسلسل فيبوناتشي.
- ثم استخدم `while` لحساب مجموع هذه الأرقام.

4. إيجاد الأرقام التي تقبل القسمة على 3 و5 باستخدام `for` مع جمعهم في `while`

- استخدم حلقة `for` لطباعة الأرقام من 1 إلى 100.
- استخدم `while` لجمع الأرقام التي تقبل القسمة على 3 و5.

5. عدد تكرار حرف معين في نص باستخدام `for` ثم `while` للتأكد من الكمية

- استخدم حلقة `for` للتكرار على النص.
- استخدم حلقة `while` للتحقق إذا كانت الكمية التي تم العثور عليها صحيحة.

6. طباعة الأعداد الفردية والزوجية من 1 إلى 50 باستخدام `for` مع استخدام `while` للتحقق من الشروط

- استخدم `for` لطباعة الأعداد.
- استخدم `while` للتحقق إذا كان العدد فردياً أم زوجياً.

7. إنشاء جدول ضرب باستخدام `for` و `while`

- استخدم `for` لتحديد الرقم الأساسي (مثلاً، الرقم 7).
- استخدم `while` لحساب نواتج ضرب هذا الرقم مع الأرقام من 1 إلى 10.

8. حساب المتوسط باستخدام `for` و `while`

- استخدم `for` لحساب مجموع الأرقام.
- استخدم `while` لحساب عدد الأرقام التي تم إضافتها للحصول على المتوسط.

9. التحقق من تكرار أرقام باستخدام `for` و `while`

- استخدم حلقة `for` لإدخال الأرقام في مصفوفة.
- استخدم `while` للتحقق إذا كانت الأرقام متكررة.

10. برنامج لحساب القوى باستخدام `for` ثم `while` للتحقق من النتيجة

- استخدم `for` لحساب القوى من 1 إلى 10 لعدد معين.
- ثم استخدم `while` للتحقق إذا كانت النتيجة تتطابق مع حسابك اليدوي.

تعليق:

تجمع هذه الأفكار بين الحلقات `for` و `while` وتسمح لك بالتدرب على كيفية استخدام كل منهما في السياقات المختلفة. يمكنك تعديل هذه الأفكار وتوسيعها حسب الحاجة! هل تحتاج لمساعدة إضافية في شرح أي من هذه الأفكار؟ 😊



```
السؤال الاول: let i = 1
for (i < 101; i++; )
  console.log(i)
  sum += i
  ++i
console.log(`this is sum ${sum}`){
  } while (i < 501)
  console.log(i)
  sum+=i
  ++i
console.log(`this is sum:${sum}`){
```



الكود الذي كتبته يحتوي على بعض الأخطاء التي تحتاج إلى تصحيح. سأوضح لك هذه الأخطاء وأقدم لك الكود الصحيح:

الأخطاء:

1. في حلقة `for`، لا توجد شرط البداية بشكل صحيح. يجب أن تبدأ الحلقة بـ `let i = 1` ثم تضع شرط التكرار، وأخيرًا وضع التزايد.

2. في `for`، يوجد `++i` داخل الحلقة، مما يتسبب في زيادة `i` مرتين (مرة في التزايد الخاص بـ `for` ومرة أخرى داخل الكود). هذا غير مطلوب.
3. في حلقة `while`، بدأ المتغير `i` من 101 بشكل خاطئ. يجب أن يكون هو نفس المتغير `i` الذي استخدمته في حلقة `for`.

الكود المصحح:

javascript

```
let sum = 0;
let i = 1;

// لحساب المجموع من 1 إلى 100 حلقة for
for (i = 1; i <= 100; i++) {
  console.log(i); // طباعة الرقم
  sum += i; // إضافة الرقم إلى المجموع
}
console.log(`this is sum: ${sum}`); // طباعة المجموع

// لحساب المجموع من 101 إلى 500 حلقة while
while (i <= 500) {
  console.log(i); // طباعة الرقم
  sum += i; // إضافة الرقم إلى المجموع
  i++; // زيادة المتغير i
}

console.log(`this is sum: ${sum}`); // طباعة المجموع النهائي
```

شرح الكود:

1. حلقة `for`:

- تبدأ الحلقة من 1 وتستمر حتى 100.
- في كل دورة، يتم طباعة الرقم وإضافته إلى المتغير `sum`.

2. حلقة `while`:

- بعد انتهاء حلقة `for`، يبدأ `i` من 101 وتستمر الحلقة حتى 500.
- يتم طباعة الرقم وإضافته إلى المتغير `sum` في كل دورة.

الإخراج المتوقع:

- ستتم طباعة الأرقام من 1 إلى 100 أولاً.
- ثم ستتم طباعة الأرقام من 101 إلى 500.
- في النهاية، سيتم طباعة مجموع الأرقام بين 1 و500.

هل هناك أي تفاصيل أخرى تحتاج إلى شرحها أو تعديلها؟ 😊



```

السؤال الاول: let sum = 0
let i = 1
for (i = 1; i <= 101; i++; )
  console.log(i)
  sum += i

console.log(`this is sum ${sum}`){
  while (i <= 501)
    console.log(i)
    sum+=i
    ++i
  console.log(`this is sum:${sum}`){

```



الكود الذي كتبته يحتوي على بعض الأخطاء في بناء حلقة `for` وحلقة `while`. سأوضح لك الأخطاء وأقدم لك الكود الصحيح:

الأخطاء:

1. في حلقة `for`، عند كتابة `i <+ 101` يجب أن تكون `i <= 101` (العلامة الصحيحة هي `<=>`).
2. في حلقة `for`، يوجد `++i` مرتين (مرة في التزايد الخاص بـ `for` ومرة أخرى داخل الكود). يجب إزالة `++i` الداخلية.
3. في حلقة `while`، القيمة الأولية لـ `i` يجب أن تكون `101` بعد انتهاء حلقة `for`، ولكنك تكرر `i = 1` داخل حلقة `while`.

الكود المصحح:

javascript

```

let sum = 0;
let i = 1;

// حساب المجموع من 1 إلى 100 حلقة for
for (i = 1; i <= 100; i++) {
  console.log(i); // طباعة الرقم
  sum += i; // إضافة الرقم إلى المجموع
}

console.log(`this is sum: ${sum}`); // طباعة المجموع بعد حلقة for

// حساب المجموع من 101 إلى 500 حلقة while
while (i <= 500) {
  console.log(i); // طباعة الرقم
  sum += i; // إضافة الرقم إلى المجموع
  i++; // زيادة المتغير i
}

console.log(`this is sum: ${sum}`); // طباعة المجموع النهائي بعد حلقة while

```

التوضيح:

1. حلقة `for`:

- تبدأ من 1 وتستمر حتى 100.
- يتم طباعة الرقم في كل تكرار وإضافته إلى المجموع.
- بعد الانتهاء من حلقة `for`، المجموع يتم طباعته.

2. حلقة `while`:

- تبدأ من الرقم `101` (بعد الانتهاء من حلقة `for`).
- تستمر حتى 500.
- يتم طباعة الرقم في كل تكرار وإضافته إلى المجموع.
- بعد الانتهاء من حلقة `while`، يتم طباعة المجموع النهائي.

الإخراج المتوقع:

- سيتم طباعة الأرقام من 1 إلى 100 أولاً.
- ثم سيتم طباعة الأرقام من 101 إلى 500.
- في النهاية، سيتم طباعة مجموع الأرقام بين 1 و500.

هل تحتاج إلى أي تعديلات أخرى أو توضيحات إضافية؟ 😊



```
let sum = 0
let i = 1
for (i = 1; i <+ 100; i++){
  console.log(i)
  sum += i
} console.log(`this is sum ${sum}`)
while (i <= 500) {
  console.log(i)
  sum+=i
  i++
}console.log(`this is sum:${sum}`)
```



الكود يحتوي على بعض الأخطاء التي تحتاج إلى تصحيح. سأوضح لك الأخطاء وأقدم لك الكود المصحح:

الأخطاء في الكود:

1. في شرط الحلقة `for`، لا يجب استخدام `i <+ 100`، الصيغة الصحيحة هي `i <= 100`، حيث أن `<+` هي العملية الصحيحة.

2. في حلقة `for`، قمت باستخدام `++i` داخل الحلقة وهذا يتسبب في زيادة `i` مرتين (مرة في التزايد الخاص بـ `for` ومرة أخرى في الكود). يجب إزالة `++i` الداخلية.
3. لا تحتاج إلى إعادة تعيين `i = 1` داخل حلقة `while`، لأنك قد بدأت حلقة `for` باستخدام `i = 1` بالفعل.

الكود المصحح:

javascript

```
let sum = 0;
let i = 1;

// حساب المجموع من 1 إلى 100 حلقة for
for (i = 1; i <= 100; i++) {
  console.log(i); // طباعة الرقم
  sum += i; // إضافة الرقم إلى المجموع
}

console.log(`this is sum: ${sum}`); // طباعة المجموع بعد حلقة for

// حساب المجموع من 101 إلى 500 حلقة while
while (i <= 500) {
  console.log(i); // طباعة الرقم
  sum += i; // إضافة الرقم إلى المجموع
  i++; // زيادة المتغير i
}

console.log(`this is sum: ${sum}`); // طباعة المجموع النهائي بعد حلقة while
```

شرح الكود:

1. حلقة `for`:

- تبدأ من 1 وتستمر حتى 100 (`i <= 100`).
- في كل مرة يتم طباعة قيمة `i` وإضافتها إلى المتغير `sum`.

2. حلقة `while`:

- بعد اكتمال حلقة `for`، تبدأ قيمة `i` من 101.
- يتم إضافة الأرقام من 101 إلى 500 إلى `sum` وطباعة كل رقم في كل دورة.

الإخراج المتوقع:

- ستم طباعة الأرقام من 1 إلى 100 أولاً.
- ثم ستم طباعة الأرقام من 101 إلى 500.
- سيتم في النهاية طباعة مجموع الأرقام بين 1 و500.

هل هناك أي شيء آخر تريد توضيحه أو تعديله؟ 😊