1. ما هو الفرق بين var و let و const في JavaScript؟

الإجابة:

var: كانت تستخدم في JavaScript القديمة. متغيرة النطاق (scope) هي الدالة أو الكود الذي يحتوي عليها.

let: يمكن إعادة تعيينها، ومجالها محلي (block-scoped).

const: لا يمكن إعادة تعيينها بعد تعيين القيمة، ومجالها محلي (block-scoped).

2. ما هو الـ Hoisting في JavaScript؟

الإجابة:
الـ Hoisting هو ميزة في JavaScript حيث يتم رفع تعريف المتغيرات والدوال إلى أعلى النطاق الذي توجد فيه. ومع ذلك، يتم رفع المتغيرات التي تم تعريفها باستخدام var فقط، بينما لا يتم رفع المتغيرات التي تم تعريفها باستخدام let و const.

3. ما هو الفرق بين null و undefined؟

الإجابة:

null: قيمة تم تعيينها بشكل صريح لتشير إلى عدم وجود قيمة.

undefined: تعني أن المتغير تم تعريفه ولكن لم يتم تعيين قيمة له بعد.

4. ما هو الـ Closure في JavaScript؟

الإجابة:
الـ Closure هو وظيفة يمكنها الوصول إلى المتغيرات في نطاقها المحلي حتى بعد أن تنتهي الدالة الأصلية من التنفيذ. بمعنى آخر، يمكن للدالة الداخلية الوصول إلى المتغيرات التي كانت موجودة في النطاق الخارجي.

5. ما هو الـ Callback Function؟

الإجابة:
الـ Callback Function هي دالة تُمرر إلى دالة أخرى كوسيلة لتنفيذها بعد أن تنتهي الدالة الأصلية من تنفيذها. تُستخدم كثيرًا في التعامل مع العمليات غير المتزامنة مثل الـ setTimeout أو الـ fetch.

6. ما هو الفرق بين == و ===؟

الإجابة:

==: يقوم بمقارنة القيم بعد تحويل الأنواع إذا لزم الأمر.

===: يقوم بمقارنة القيم والنوع بدون تحويل الأنواع.

7. ما هو الـ Event Loop في JavaScript؟

الإجابة:
الـ Event Loop هو آلية في JavaScript تتحكم في تنفيذ العمليات غير المتزامنة. يقوم بالتحقق من الـ Call Stack وإذا كان فارغًا، يقوم بتنفيذ الـ Callback Functions التي تم وضعها في الـ Event Queue.

8. ما هو الـ Promise في JavaScript؟

الإجابة:
الـ Promise هو كائن يُستخدم لإدارة العمليات غير المتزامنة. يمكن أن يكون في أحد الحالات التالية: pending (قيد الانتظار)، resolved (تمت الموافقة)، أو rejected (تم الرفض).

9. ما هي الـ Arrow Functions في JavaScript؟

الإجابة:
الـ Arrow Functions هي دوال قصيرة في JavaScript يتم تعريفها باستخدام =>. وهي لا تمتلك خاصية this الخاصة بها، بل تستخدم this من السياق الذي تم تعريفها فيه.

10. ما هو الـ Destructuring في JavaScript؟

الإجابة:
الـ Destructuring هو طريقة لاستخراج القيم من المصفوفات أو الكائنات وتعيينها إلى متغيرات. على سبيل المثال:

```
;{ const person = { name: "John", age: 30
;const { name, age } = person
"console.log(name); // "John
console.log(age); // 30
```

11. ما هو الـ this في JavaScript؟

الإجابة:
this هو مرجع إلى الكائن الذي يستدعي الدالة. في السياقات المختلفة، يمكن أن يشير this إلى كائنات مختلفة. في الدوال العادية، يشير إلى الكائن الذي يحتوي على الدالة، بينما في دوال الـ Arrow Functions، يشير إلى الكائن الذي تم تعريف الدالة فيه.

12. ما هي الـ async/await في JavaScript؟

الإجابة:
async و await هما طريقتان للتعامل مع الـ Promises بطريقة أكثر وضوحًا. async يتم إضافتها قبل الدالة لجعلها ترجع Promise، و await يتم استخدامها داخل دالة async لإيقاف التنفيذ حتى يتم حل الـ Promise.

13. كيف يمكن تحويل الكائن إلى JSON في JavaScript؟

الإجابة:
يمكنك تحويل الكائن إلى JSON باستخدام JSON.stringify():

```
{ const obj = { name: "John", age: 30 };
const json = JSON.stringify(obj);
console.log(json); // '{"name":"John","age":30}'
```

14. ما هو الـ setTimeout و setInterval في JavaScript؟

الإجابة:

setTimeout: يقوم بتحديد دالة ليتم تنفيذها بعد فترة زمنية معينة (بالملي ثانية).

setInterval: يقوم بتحديد دالة ليتم تنفيذها بشكل متكرر بعد فترة زمنية معينة.

15. ما هو الفرق بين الـ map و forEach في JavaScript؟

الإجابة:

map: يقوم بإنشاء مصفوفة جديدة بناءً على النتائج التي تم إرجاعها من الدالة المرسلة.

forEach: يقوم بتكرار العناصر في المصفوفة ولكن لا يُرجع قيمة جديدة.

16. ما هو الفرق بين function و () => {} (Arrow Function) في JavaScript؟

الإجابة:

function: يستخدم this الخاص بالدالة.

() => {}: لا يحتوي على this الخاص به، ويستخدم this من السياق الخارجي.

17. كيف يمكن تعريف دالة في JavaScript؟

الإجابة:

باستخدام الكلمة المفتاحية function:

```
function myFunction() { }
```

باستخدام الـ Arrow Function:

```
const myFunction = () => { }
```

18. ما هو الـ bind في JavaScript؟

الإجابة:
bind هي دالة تُستخدم لإنشاء دالة جديدة مع سياق (this) معين. يتم استخدامها عندما تحتاج إلى تخصيص this لدالة معينة:

```
const obj = { name: "John" };
const greet = function() {
  console.log(this.name);
};
const greetObj = greet.bind(obj);
greetObj(); // "John"
```

19. ما هو الفرق بين call و apply؟

الإجابة:

call: يستخدم لاستدعاء دالة مع this خاص بمحدد، حيث يتم تمرير المتغيرات كوسائط منفصلة.

apply: مثل call ولكن يتم تمرير المتغيرات كمصفوفة.

20. متى يتم استخدام الـ setInterval و clearInterval؟

الإجابة:

setInterval: يستخدم لتنفيذ دالة بشكل متكرر بعد فترة زمنية معينة.

clearInterval: يستخدم لإيقاف التنفيذ المتكرر الذي تم تحديده بواسطة setInterval.

21. ما هو الـ spread operator؟

الإجابة:

الـ Spread Operator (...) يستخدم لتفكيك العناصر في المصفوفات أو الكائنات. يمكن استخدامه لدمج المصفوفات أو تكرار الكائنات:

```
;[const arr = [1, 2, 3
;[const arr2 = [...arr, 4, 5
[console.log(arr2); // [1, 2, 3, 4, 5
```

22. ما هي الـ Rest Parameters؟

الإجابة:
الـ Rest Parameters (...) يسمح بتمرير عدد غير محدد من الوسائط إلى دالة على شكل مصفوفة:

```
} (function sum(...numbers
 ;(return numbers.reduce((a, b) => a + b, 0
{
console.log(sum(1, 2, 3, 4)); // 10
```

23. ما هو الـ default parameters في JavaScript؟

الإجابة:
يتم تحديد قيم افتراضية للوسائط في حال لم يتم تمريرها:

```
} ("function greet(name = "Guest
 ;(`{console.log(`Hello, ${name
{
"greet(); // "Hello, Guest
"greet("John"); // "Hello, John
```

24. ما هو الـ prototype في JavaScript؟

الإجابة:
الـ prototype هو كائن يتم ربطه بكل دالة في JavaScript. يمكن استخدامه لإضافة خصائص وطرق إلى الدوال والكائنات.

```
} (function Person(name
    ;this.name = name
{
} ()Person.prototype.greet = function
 ;(`{console.log(`Hello, ${this.name
;{
;("const john = new Person("John
"john.greet(); // "Hello, John
```

25. كيف يمكن التحقق من نوع البيانات في JavaScript؟

الإجابة:
يمكنك استخدام typeof و instanceof للتحقق من نوع البيانات:

"typeof "hello"; // "string
"typeof 42; // "number
"typeof true; // "boolean
"typeof {} // "object
"typeof [] // "object" (لكن المصفوفات هي نوع خاص من الكائنات)

26. ما هو Object.freeze؟

الإجابة:
Object.freeze هو طريقة تجعل الكائن غير قابل للتعديل بعد إنشائه. لا يمكن إضافة أو حذف أو تعديل خصائصه:

;{ "const person = { name: "John
;(Object.freeze(person
"person.name = "Jane"; // لا يحدث شيء

27. ما هو الـ JSON.parse() و JSON.stringify()؟

الإجابة:

JSON.stringify(): يحول الكائنات إلى JSON String.

JSON.parse(): يحول JSON String إلى كائن(objecet
JavaScript).

28. ما هو الـ new في JavaScript؟

الإجابة:
new يستخدم لإنشاء كائنات جديدة من دالة منشئ (Constructor Function). عندما يتم استخدام new، يتم إنشاء كائن جديد ويتم ربطه بالدالة المنشئة.

29. ما هو الفرق بين الـ set و map في JavaScript؟

الإجابة:

set: هي مجموعة من القيم الفريدة.

map: هي مجموعة من الأزواج (المفتاح: القيمة)، حيث يمكن أن تكون المفاتيح من أي نوع.

30. ما هو الفرق بين الـ for...in و for...of؟

الإجابة:

for...in: يستخدم للتكرار عبر خصائص الكائن.

for...of: يستخدم للتكرار عبر القيم في المصفوفات أو الكائنات القابلة للتكرار.

31. ما هي المصفوفات المرتبطة بـ Promise؟

الإجابة:
المصفوفات المرتبطة بـ Promise هي:

Promise.all(): ينتظر تنفيذ جميع الـ Promises.

Promise.race(): ينتظر تنفيذ أول Promise.

Promise.allSettled(): ينتظر تنفيذ جميع الـ Promises بغض النظر عن النتيجة.

Promise.any(): ينتظر تنفيذ أول Promise بنجاح.

32. ما هو الـ setTimeout؟ وكيف يمكن استخدامه؟

الإجابة:
setTimeout يستخدم لتأخير تنفيذ دالة لمدة زمنية محددة:

```
setTimeout(() => { console.log("Hello!"); }, 1000); // سيتم تنفيذها بعد 1 ثانية
```

33. ما هي الدوال ذاتية الاستدعاء (IIFE)؟

الإجابة:
الدوال ذاتية الاستدعاء (Immediately Invoked Function Expression) هي دوال تُنفذ مباشرة بعد تعريفها:

```
(function() {
  console.log("Hello, World!");
})();
```

34. ما هو التابع Math.random()؟

الإجابة:
Math.random() يُرجع رقم عشري عشوائي بين 0 (شاملة) و 1 (غير شاملة).

Math.random(); // قيمة بين 0 و 1

35. ما هو الفرق بين call و apply؟

الإجابة:

call: يسمح بتمرير معطيات فردية.

apply: يسمح بتمرير المعطيات كمصفوفة.

1. ما هو الـ this في JavaScript؟

الإجابة:
الـ this يشير إلى الكائن الذي يتم استدعاء الدالة منه. في الدوال العادية، يشير إلى الكائن الذي يحتوي على الدالة. في الدوال داخل الكائنات، يشير إلى الكائن نفسه.

2. ما الفرق بين null و undefined؟

الإجابة:

null: قيمة تم تعيينها يدويًا تعني "لا شيء".

undefined: قيمة يتم تعيينها تلقائيًا عندما لا يتم تعريف متغير.

3. ما هو الـ closure في JavaScript؟

الإجابة:
الـ closure هو دالة تحتوي على مرجع للمتغيرات من البيئة التي تم إنشاؤها فيها، حتى لو تم استدعاؤها في بيئة مختلفة.

4. ما هي الـ callback function؟

الإجابة:
دالة يتم تمريرها كوسيط لدالة أخرى وتنفذ في وقت لاحق، عادة بعد الانتهاء من عملية غير متزامنة.

5. ما هو الـ Promise في JavaScript؟

الإجابة:
الـ Promise هو كائن يمثل العملية غير المتزامنة التي قد تكتمل أو تفشل في المستقبل.

6. ما الفرق بين == و === في JavaScript؟

الإجابة:

==: يقارن القيم بعد تحويلها إلى نفس النوع.

===: يقارن القيم والنوع دون تحويل.

7. ما هو الـ setTimeout؟

الإجابة:
setTimeout يستخدم لتأخير تنفيذ دالة لمدة معينة من الوقت (بالملي ثانية).

8. ما هو الـ setInterval؟

الإجابة:
setInterval يستخدم لتنفيذ دالة بشكل متكرر بعد فترة زمنية محددة.

9. ما هو الـ typeof؟

الإجابة:
typeof هو عامل يُستخدم للتحقق من نوع البيانات (مثل string, number, boolean, إلخ).

10. ما هو الـ spread operator؟

الإجابة:
الـ spread operator (...) يستخدم لتفكيك العناصر في المصفوفات أو الكائنات.

11. ما هو الـ rest parameter؟

الإجابة:
الـ rest parameter (...) يسمح بتمرير عدد غير محدد من الوسائط إلى دالة على شكل مصفوفة.

12. كيف يمكن تحويل كائن إلى JSON؟

الإجابة:
باستخدام JSON.stringify():

```
const obj = { name: "John" };
const json = JSON.stringify(obj);
```

13. كيف يمكن تحويل JSON إلى كائن؟

الإجابة:
باستخدام JSON.parse():

```
const json = '{"name":"John"}';
const obj = JSON.parse(json);
```

14. ما هو الـ bind؟

الإجابة:
bind هو طريقة تستخدم لإنشاء دالة جديدة مع سياق (this) محدد.

15. ما هو الفرق بين call و apply؟

الإجابة:

call: يستخدم لتمرير المعطيات بشكل فردي.

apply: يستخدم لتمرير المعطيات كمصفوفة.

16. ما هو الفرق بين for...in و for...of؟

الإجابة:

for...in: يستخدم للتكرار عبر خصائص الكائن.

for...of: يستخدم للتكرار عبر القيم في المصفوفات أو الكائنات القابلة للتكرار.

17. ما هو الـ prototype؟

الإجابة:
الـ prototype هو كائن يتم ربطه بكل دالة في JavaScript ويمكن استخدامه لإضافة خصائص وطرق إلى الكائنات.

18. ما هو الـ new؟

الإجابة:
new يستخدم لإنشاء كائن جديد باستخدام دالة منشئة (constructor function).

19. ما هو الـ Object.freeze()؟

الإجابة:
Object.freeze() يستخدم لجعل الكائن غير قابل للتعديل بعد إنشائه.

20. ما هو الـ async/await؟

الإجابة:
async/await هو بناء جديد يسمح بكتابة الكود غير المتزامن بطريقة أكثر وضوحًا وسهولة في القراءة.

21. ما هو الـ Math.random()؟

الإجابة:
Math.random() يُرجع قيمة عشوائية بين 0 و 1.

22. كيف يتم استخدام الـ try/catch؟

الإجابة:
يتم استخدام try/catch لمعالجة الأخطاء. إذا حدث خطأ داخل الكود في try، يتم الانتقال إلى الكود في catch:

```
} try {
  // كود قد يسبب خطأ
} (catch (error {
  (console.log(error);
{
```

23. ما هو الفرق بين الـ localStorage و sessionStorage؟

الإجابة:

localStorage: يحتفظ بالبيانات حتى بعد إغلاق المتصفح.

sessionStorage: يحتفظ بالبيانات فقط خلال جلسة المتصفح الحالية.

24. ما هو الـ constructor؟

الإجابة:
constructor هو دالة خاصة تستخدم لإنشاء كائنات جديدة من دالة منشئة.

25. ما هو الـ event delegation؟

الإجابة:

event delegation هو نمط يستخدم لتعيين حدث واحد للأب بدلاً من تعيينه لكل عنصر فرعي.

26. ما هو الـ module في JavaScript؟

الإجابة:
module هو وحدة من الكود يمكن استيرادها أو تصديرها بين الملفات في JavaScript.

27. ما هو الـ import/export؟

الإجابة:

import: يستخدم لاستيراد الكود من ملف آخر.

export: يستخدم لتصدير الكود من ملف ليتم استخدامه في ملفات أخرى.

28. ما هو الـ symbol في JavaScript؟

الإجابة:
symbol هو نوع بيانات فريد وغير قابل للتكرار يستخدم كـ مفتاح خاص في الكائنات.

29. ما هو الـ set و map؟

الإجابة:

set: مجموعة من القيم الفريدة.

map: مجموعة من الأزواج (المفتاح: القيمة)، حيث يمكن أن تكون المفاتيح من أي نوع.

30. ما هو الـ event loop في JavaScript؟

الإجابة:
الـ event loop هو آلية في JavaScript لإدارة الأحداث غير المتزامنة وضمان تنفيذ الكود في ترتيب صحيح.

31. ما هو الفرق بين الـ var و let و const؟

الإجابة:

var: يمكن إعادة تعيينه ويملك نطاقًا دالياً.

let: يمكن إعادة تعيينه ويملك نطاقًا بلوكيًا.

const: لا يمكن إعادة تعيينه ويملك نطاقًا بلوكيًا.

32. ما هو الـ array destructuring؟

الإجابة:
الـ array destructuring يسمح بتفكيك المصفوفات إلى متغيرات منفصلة:

```
;[const arr = [1, 2, 3
;const [a, b, c] = arr
```

33. ما هو الـ object destructuring؟

الإجابة:
الـ object destructuring يسمح بتفكيك الكائنات إلى متغيرات منفصلة:

```
;{ const obj = { name: "John", age: 30
;const { name, age } = obj
```

34. ما هو الـ new Promise()؟

الإجابة:
new Promise() هو كائن يُستخدم لتمثيل عملية غير متزامنة قد تكتمل في المستقبل.

35. ما هو الـ event bubbling؟

الإجابة:
event bubbling هو آلية تحدث عندما يتم تنفيذ حدث على عنصر فرعي ثم ينتقل إلى العناصر الأب.

36. ما هو الـ event capturing؟

الإجابة:
event capturing هو عكس الـ event bubbling حيث يتم تنفيذ الحدث من العنصر الأب إلى العنصر الفرعي.

37. ما هو الـ debounce؟

الإجابة:
debounce هو نمط يستخدم لتأخير تنفيذ دالة حتى ينقضي وقت معين بعد آخر حدث.

38. ما هو الـ throttle؟

الإجابة:
throttle هو نمط يستخدم للحد من عدد المرات التي يتم فيها تنفيذ دالة خلال فترة زمنية معينة.

39. ما هو الـ destructuring في JavaScript؟

الإجابة:
destructuring هو نمط يسمح بتفكيك المصفوفات والكائنات بسهولة للحصول على القيم داخلها.

40. ما هو الـ setTimeout و clearTimeout؟

الإجابة:

setTimeout: يستخدم لتأخير تنفيذ دالة.

clearTimeout: يستخدم لإلغاء تنفيذ الدالة التي تم تأجيلها.

41. ما هو الفرق بين forEach و map؟

الإجابة:

forEach: يستخدم للتكرار عبر المصفوفات ولكن لا يُرجع قيمة.

map: يستخدم للتكرار عبر المصفوفات ويُرجع مصفوفة جديدة.

42. ما هو الفرق بين slice و splice؟

الإجابة:

slice: يستخدم لاستخراج جزء من المصفوفة دون تعديل المصفوفة الأصلية.

splice: يستخدم لإضافة أو إزالة عناصر من المصفوفة.

43. ما هو الـ Promise.all()؟

الإجابة:
Promise.all() ينتظر تنفيذ جميع الـ Promises ويفشل إذا فشل أي منها.

44. ما هو الـ Promise.race()؟

الإجابة:
Promise.race() ينتظر تنفيذ أول Promise فقط.

45. ما هو الـ Object.assign()؟

الإجابة:
Object.assign() يُستخدم لنسخ خصائص من كائن إلى كائن آخر.

46. ما هو الـ Array.prototype.filter()؟

الإجابة:
filter() يُستخدم لإنشاء مصفوفة جديدة تحتوي على العناصر التي تطابق شرطًا معين
ًا.

47. ما هو الـ Array.prototype.reduce()؟

الإجابة:
reduce() يُستخدم لتطبيق دالة على كل عنصر في المصفوفة وتجميع النتائج في قيمة واحدة .

48. ما هو الـ Array.prototype.map()؟

الإجابة:
map() يُستخدم لتحويل كل عنصر في المصفوفة إلى قيمة جديدة بناءً على دالة معينة.

49. ما هو الـ Array.prototype.find()؟

الإجابة:
find() يُستخدم لإيجاد أول عنصر في المصفوفة يطابق شرطًا معينًا.

50. ما هو الـ Array.prototype.some()؟

الإجابة:
some() يُستخدم للتحقق إذا كان هناك أي عنصر في المصفوفة يطابق شرطًا معينًا.

51. ما هو الـ Array.prototype.every()؟

الإجابة:
every() يُستخدم للتحقق إذا كان جميع العناصر في المصفوفة تطابق شرطًا معينًا.

52. ما هو الـ Array.prototype.indexOf()؟

الإجابة:
indexOf() يُستخدم للبحث عن عنصر في المصفوفة وإرجاع موقعه.

53. ما هو الـ Object.keys()؟

الإجابة:
Object.keys() يُستخدم للحصول على مصفوفة تحتوي على جميع المفاتيح في الكائن.

54. ما هو الـ Object.values()؟

الإجابة:
Object.values() يُستخدم للحصول على مصفوفة تحتوي على جميع القيم في الكائن.

55. ما هو الـ Object.entries()؟

الإجابة:
Object.entries() يُستخدم للحصول على مصفوفة تحتوي على جميع الأزواج (المفتاح: القيمة) في الكائن.

56. ما هو الـ Array.prototype.concat()؟

الإجابة:
concat() يُستخدم لدمج مصفوفات متعددة في مصفوفة واحدة.

57. ما هو الـ Array.prototype.join()؟

الإجابة:
join() يُستخدم لتحويل المصفوفة إلى سلسلة نصية مفصولة بفاصل معين.

58. ما هو الـ Array.prototype.reverse()؟

الإجابة:
reverse() يُستخدم لعكس ترتيب العناصر في المصفوفة.

59. ما هو الـ Array.prototype.sort()؟

الإجابة:
sort() يُستخدم لترتيب العناصر في المصفوفة وفقًا لمعيار معين.

60. ما هو الـ Array.prototype.slice()؟

الإجابة:
slice() يُستخدم لاستخراج جزء من المصفوفة دون تعديل المصفوفة الأصلية.

61. ما هو الـ Array.prototype.unshift()؟

الإجابة:
unshift() يُستخدم لإضافة عنصر في بداية المصفوفة.

62. ما هو الـ Array.prototype.push()؟

الإجابة:
push() يُستخدم لإضافة عنصر في نهاية المصفوفة.

63. ما هو الـ Array.prototype.pop()؟

الإجابة:
pop() يُستخدم لإزالة العنصر الأخير من المصفوفة.

64. ما هو الـ Array.prototype.shift()؟

الإجابة:
shift() يُستخدم لإزالة العنصر الأول من المصفوفة.

65. ما هو الـ JSON.stringify()؟

الإجابة:
JSON.stringify() يُستخدم لتحويل الكائنات إلى تنسيق JSON.

66. ما هو الـ JSON.parse()؟

الإجابة:
JSON.parse() يُستخدم لتحويل JSON إلى كائن JavaScript.

67. ما هو الـ bind()؟

الإجابة:
bind() يُستخدم لإنشاء دالة جديدة مع سياق (this) محدد.

68. ما هو الـ call()؟

الإجابة:
call() يُستخدم لاستدعاء دالة مع سياق (this) محدد وتمرير المعطيات.

69. ما هو الـ apply()؟

الإجابة:
apply() يُستخدم لاستدعاء دالة مع سياق (this) محدد وتمرير المعطيات كمصفوفة.

70. ما هو الـ Object.create()؟

الإجابة:
Object.create() يُستخدم لإنشاء كائن جديد مع تحديد الكائن الذي سيكون الكائن الأب له.

71. ما هو الـ Object.defineProperty()؟

الإجابة:
Object.defineProperty() يُستخدم لتعريف خاصية جديدة على كائن.

72. ما هو الـ Object.getOwnPropertyDescriptor()؟

الإجابة:
Object.getOwnPropertyDescriptor() يُستخدم للحصول على وصف الخاصية من الكائن.

73. ما هو الـ for...in؟

الإجابة:
for...in يُستخدم للتكرار عبر خصائص الكائن.

74. ما هو الـ for...of؟

الإجابة:
for...of يُستخدم للتكرار عبر القيم في المصفوفات أو الكائنات القابلة للتكرار.

75. ما هو الـ Map؟

الإجابة:
Map هو هيكل بيانات يسمح بتخزين الأزواج (المفتاح: القيمة) حيث يمكن أن تكون المفاتيح من أي نوع.

76. ما هو الـ Set؟

الإجابة:
Set هو هيكل بيانات يسمح بتخزين القيم الفريدة فقط.

77. ما هو الـ WeakMap؟

الإجابة:
WeakMap هو نسخة من Map حيث تكون المفاتيح من نوع كائن فقط ولا يتم الاحتفاظ بها في الذاكرة بعد إزالة جميع المراجع إليها.

78. ما هو الـ WeakSet؟

الإجابة:
WeakSet هو نسخة من Set حيث يمكن تخزين الكائنات فقط ولا يتم الاحتفاظ بها في الذاكرة بعد إزالة جميع المراجع إليها.

79. ما هو الـ Promise.allSettled()؟

الإجابة:
Promise.allSettled() ينتظر تنفيذ جميع الـ Promises سواء كانت قد اكتملت أو فشلت.

80. ما هو الـ Promise.any()؟

الإجابة:
Promise.any() ينتظر تنفيذ أول Promise ناجح.

81. ما هو الـ BigInt؟

الإجابة:
BigInt هو نوع بيانات جديد في JavaScript لتمثيل الأعداد الكبيرة التي تتجاوز حدود الـ Number.

82. ما هو الـ template literals؟

الإجابة:
template literals هي طريقة لكتابة السلاسل النصية مع دعم التعابير داخل السلسلة باستخدام ${}$.

83. ما هو الـ async function؟

الإجابة:
async function هو دالة تُستخدم لكتابة كود غير متزامن باستخدام await داخلها.

84. ما هو الـ await؟

الإجابة:
await يُستخدم في دالة async لانتظار تنفيذ Promise.

85. ما هو الـ finally في الـ try/catch؟

الإجابة:
finally هو كتلة تُنفذ بعد تنفيذ try أو catch بغض النظر عن حدوث خطأ أم لا.

86. ما هو الـ symbol؟

الإجابة:
symbol هو نوع بيانات فريد وغير قابل للتكرار يستخدم كمفتاح خاص في الكائنات.

87. ما هو الـ map في الـ Map؟

الإجابة:
map هو هيكل بيانات يُستخدم لتخزين الأزواج (المفتاح: القيمة) حيث يمكن أن تكون المفاتيح من أي نوع.

88. ما هو الـ reduce()؟

الإجابة:
reduce() يُستخدم لتجميع قيم المصفوفة في قيمة واحدة باستخدام دالة تجميع.

89. ما هو الـ split()؟

الإجابة:
split() يُستخدم لتقسيم سلسلة نصية إلى مصفوفة بناءً على فاصل معين.

90. ما هو الـ join()؟

الإجابة:
join() يُستخدم لتحويل المصفوفة إلى سلسلة نصية مفصولة بفاصل معين.

91. ما هو الـ Object.assign()؟

الإجابة:
Object.assign() يُستخدم لنسخ خصائص من كائن إلى كائن آخر.

92. ما هو الـ Object.create()؟

الإجابة:
Object.create() يُستخدم لإنشاء كائن جديد مع تحديد الكائن الذي سيكون الكائن الأب له.

93. ما هو الـ String.prototype.includes()؟

الإجابة:
includes() يُستخدم للتحقق إذا كانت السلسلة تحتوي على قيمة معينة.

94. ما هو الـ String.prototype.indexOf()؟

الإجابة:
indexOf() يُستخدم للبحث عن موقع أول ظهور لقيمة معينة في السلسلة.

95. ما هو الـ String.prototype.slice()؟

الإجابة:
slice() يُستخدم لاستخراج جزء من السلسلة النصية.

96. ما هو الـ String.prototype.replace()؟

الإجابة:
replace() يُستخدم لاستبدال جزء من السلسلة النصية بقيمة جديدة.

97. ما هو الـ String.prototype.trim()؟

الإجابة:
trim() يُستخدم لإزالة المسافات البيضاء من بداية ونهاية السلسلة النصية.

98. ما هو الـ String.prototype.split()؟

الإجابة:
split() يُستخدم لتقسيم السلسلة النصية إلى مصفوفة بناءً على فاصل معين.

99. ما هو الـ String.prototype.toLowerCase()؟

الإجابة:
toLowerCase() يُستخدم لتحويل جميع الحروف في السلسلة النصية إلى حروف صغيرة.

100. ما هو الـ String.prototype.toUpperCase()؟

الإجابة:
toUpperCase() يُستخدم لتحويل جميع الحروف في السلسلة النصية إلى حروف كبيرة.

101. ما هو الـ Array.prototype.fill()؟

الإجابة:
fill() يُستخدم لملء جميع العناصر في المصفوفة بقيمة معينة من بداية المصفوفة إلى نهايتها.

102. ما هو الـ Array.prototype.splice()؟

الإجابة:
splice() يُستخدم-فوفة في مكان معين.

103. ما هو الـ Array.prototype.flat()؟

الإجابة:
flat() يُستخدم لتسطيح المصفوفة بحيث يتم دمج المصفوفات الفرعية في مصفوفة واحدة.

104. ما هو الـ Array.prototype.flatMap()؟

الإجابة:
flatMap() يُستخدم لتطبيق دالة على كل عنصر في المصفوفة ثم تسطيح النتيجة إلى مصفوفة واحدة.

105. ما هو الـ Array.prototype.sort()؟

الإجابة:
sort() يُستخدم لترتيب العناصر في المصفوفة وفقًا لمعيار معين.

106. ما هو الـ String.prototype.charAt()؟

الإجابة:
charAt() يُستخدم لاسترجاع الحرف في موضع معين من السلسلة النصية.

107. ما هو الـ String.prototype.concat()؟

الإجابة:
concat() يُستخدم لدمج سلسلتين نصيتين أو أكثر في سلسلة واحدة.

108. ما هو الـ String.prototype.match()؟

الإجابة:
match() يُستخدم للبحث عن تطابقات مع تعبير عادي في السلسلة النصية.

109. ما هو الـ String.prototype.search()؟

الإجابة:
search() يُستخدم للبحث عن تطابق مع تعبير عادي في السلسلة النصية وإرجاع موقعه.

110. ما هو الـ String.prototype.split()؟

الإجابة:
split() يُستخدم لتقسيم السلسلة النصية إلى مصفوفة بناءً على فاصل معين.

111. ما هو الـ String.prototype.startsWith()؟

الإجابة:
startsWith() يُستخدم للتحقق إذا كانت السلسلة النصية تبدأ بقيمة معينة.

112. ما هو الـ String.prototype.endsWith()؟

الإجابة:
endsWith() يُستخدم للتحقق إذا كانت السلسلة النصية تنتهي بقيمة معينة.

113. ما هو الـ String.prototype.repeat()؟

الإجابة:
repeat() يُستخدم لتكرار السلسلة النصية عدد معين من المرات.

114. ما هو الـ String.prototype.includes()؟

الإجابة:
includes() يُستخدم للتحقق إذا كانت السلسلة النصية تحتوي على قيمة معينة.

115. ما هو الـ String.prototype.indexOf()؟

الإجابة:
indexOf() يُستخدم للبحث عن موقع أول ظهور لقيمة معينة في السلسلة النصية.

116. ما هو الـ String.prototype.slice()؟

الإجابة:
slice() يُستخدم لاستخراج جزء من السلسلة النصية.

117. ما هو الـ String.prototype.replace()؟

الإجابة:
replace() يُستخدم لاستبدال جزء من السلسلة النصية بقيمة جديدة.

118. ما هو الـ String.prototype.trim()؟

الإجابة:
trim() يُستخدم لإزالة المسافات البيضاء من بداية ونهاية السلسلة النصية.

119. ما هو الـ String.prototype.toLowerCase()؟

الإجابة:
toLowerCase() يُستخدم لتحويل جميع الحروف في السلسلة النصية إلى حروف صغيرة.

120. ما هو الـ String.prototype.toUpperCase()؟

الإجابة:
toUpperCase() يُستخدم لتحويل جميع الحروف في السلسلة النصية إلى حروف كبيرة.

121. ما هو الـ String.prototype.padStart()؟

الإجابة:
padStart() يُستخدم لإضافة حروف أو رموز إلى بداية السلسلة النصية حتى يصل طولها إلى قيمة معينة.

122. ما هو الـ String.prototype.padEnd()؟

الإجابة:
padEnd() يُستخدم لإضافة حروف أو رموز إلى نهاية السلسلة النصية حتى يصل طولها إلى قيمة معينة.

123. ما هو الـ Set.prototype.add()؟

الإجابة:
add() يُستخدم لإضافة عنصر إلى مجموعة Set.

124. ما هو الـ Set.prototype.has()؟

الإجابة:
has() يُستخدم للتحقق إذا كانت مجموعة Set تحتوي على عنصر معين.

125. ما هو الـ Set.prototype.delete()؟

الإجابة:
delete() يُستخدم لإزالة عنصر من مجموعة Set.

126. ما هو الـ Set.prototype.clear()؟

الإجابة:
clear() يُستخدم لإزالة جميع العناصر من مجموعة Set.

127. ما هو الـ Map.prototype.set()؟

الإجابة:
set() يُستخدم لإضافة أو تحديث عنصر في Map.

128. ما هو الـ Map.prototype.get()؟

الإجابة:
get() يُستخدم لاسترجاع قيمة معينة من Map باستخدام المفتاح.

129. ما هو الـ Map.prototype.has()؟

الإجابة:
has() يُستخدم للتحقق إذا كانت Map تحتوي على مفتاح معين.

130. ما هو الـ Map.prototype.delete()؟

الإجابة:
delete() يُستخدم لإزالة عنصر من Map باستخدام المفتاح.

131. ما هو الـ Map.prototype.clear()؟

الإجابة:
clear() يُستخدم لإزالة جميع العناصر من Map.

132. ما هو الـ WeakMap.prototype.set()؟

الإجابة:
set() يُستخدم لإضافة عنصر إلى WeakMap.

133. ما هو الـ WeakMap.prototype.get()؟

الإجابة:
get() يُستخدم لاسترجاع قيمة معينة من WeakMap باستخدام المفتاح.

134. ما هو الـ WeakMap.prototype.has()؟

الإجابة:
has() يُستخدم للتحقق إذا كانت WeakMap تحتوي على مفتاح معين.

135. ما هو الـ WeakMap.prototype.delete()؟

الإجابة:
delete() يُستخدم لإزالة عنصر من WeakMap باستخدام المفتاح.

136. ما هو الـ WeakSet.prototype.add()؟

الإجابة:
add() يُستخدم لإضافة عنصر إلى WeakSet.

137. ما هو الـ WeakSet.prototype.has()؟

الإجابة:
has() يُستخدم للتحقق إذا كانت WeakSet تحتوي على عنصر معين.

138. ما هو الـ WeakSet.prototype.delete()؟

الإجابة:
delete() يُستخدم لإزالة عنصر من WeakSet.

139. ما هو الـ Object.freeze()؟

الإجابة:
Object.freeze() يُستخدم لمنع تعديل الكائن.

140. ما هو الـ Object.seal()؟

الإجابة:
Object.seal() يُستخدم لمنع إضافة أو حذف خصائص من الكائن، ولكن يمكن تعديل القيم.

141. ما هو الـ Object.preventExtensions()؟

الإجابة:
Object.preventExtensions() يُستخدم لمنع إضافة خصائص جديدة إلى الكائن.

142. ما هو الـ Object.is()؟

الإجابة:
Object.is() يُستخدم للتحقق إذا كان كائنين متساويين.

143. ما هو الـ Object.getPrototypeOf()؟

الإجابة:
Object.getPrototypeOf() يُستخدم للحصول على الكائن الأب للكائن الحالي.

144. ما هو الـ Object.setPrototypeOf()؟

الإجابة:
Object.setPrototypeOf() يُستخدم لتعيين الكائن الأب لكائن معين.

145. ما هو الـ Math.random()؟

الإجابة:
Math.random() يُستخدم لتوليد عدد عشري عشوائي بين 0 و 1.

146. ما هو الـ Math.floor()؟

الإجابة:
Math.floor() يُستخدم لتقريب العدد إلى أقرب عدد صحيح أقل من أو يساوي الرقم.

147. ما هو الـ Math.ceil()؟

الإجابة:
Math.ceil() يُستخدم لتقريب العدد إلى أقرب عدد صحيح أكبر من أو يساوي الرقم.

148. ما هو الـ Math.round()؟

الإجابة:
Math.round() يُستخدم لتقريب العدد إلى أقرب عدد صحيح.

149. ما هو الـ Math.max()؟

الإجابة:
Math.max() يُستخدم لإيجاد أكبر قيمة بين مجموعة من القيم.

150. ما هو الـ Math.min()؟

الإجابة:
Math.min() يُستخدم لإيجاد أصغر قيمة بين مجموعة من القيم.

151. ما هو الـ Math.pow()؟

الإجابة:
Math.pow() يُستخدم لحساب القوة (الأس) لعدد معين.

152. ما هو الـ Math.sqrt()؟

الإجابة:
Math.sqrt() يُستخدم لحساب الجذر التربيعي لعدد معين.

153. ما هو الـ Math.abs()؟

الإجابة:
Math.abs() يُستخدم لإرجاع القيمة المطلقة لعدد معين.

154. ما هو الـ Math.sin()؟

الإجابة:
Math.sin() يُستخدم لحساب الجيب لزاوية معينة.

155. ما هو الـ Math.cos()؟

الإجابة:
Math.cos() يُستخدم لحساب جيب التمام لزاوية معينة.

156. ما هو الـ Math.tan()؟

الإجابة:
Math.tan() يُستخدم لحساب الظل لزاوية معينة.

157. ما هو الـ Math.log()؟

الإجابة:
Math.log() يُستخدم لحساب اللوغاريتم الطبيعي لعدد معين.

158. ما هو الـ Math.exp()؟

الإجابة:
Math.exp() يُستخدم لحساب الأساس e لعدد معين.

159. ما هو الـ Math.random()؟

الإجابة:
Math.random() يُستخدم لتوليد عدد عشري عشوائي بين 0 و 1.

160. ما هو الـ console.log()؟

الإجابة:
console.log() يُستخدم لطباعة الرسائل أو القيم إلى وحدة التحكم (console).

201. ما هو الفرق بين null و undefined؟

الإجابة:
null هو قيمة فارغة يتم تعيينها يدويًا، بينما undefined تعني أن المتغير تم الإعلان عنه ولكن لم يتم تعيين قيمة له.

202. ما هو الـ for...in؟

الإجابة:
for...in يُستخدم للتكرار عبر جميع خصائص الكائن (الـ properties) في JavaScript.

203. ما هو الـ for...of؟

الإجابة:

for...of يُستخدم للتكرار عبر العناصر في المصفوفات أو الكائنات القابلة للتكرار (iterable objects) مثل المصفوفات والسلاسل النصية.

204. ما هو الـ async؟

الإجابة:
async هو مفتاح يُستخدم لجعل الدالة غير متزامنة (asynchronous) بحيث تُعيد Promise.

205. ما هو الـ await؟

الإجابة:
await يُستخدم داخل دالة غير متزامنة (async function) لانتظار Promise حتى يتم حلها أو رفضها.

206. ما هي الـ closure في JavaScript؟

الإجابة:
closure هو عندما تحتفظ الدالة بالوصول إلى المتغيرات من نطاقها الخارجي حتى بعد انتهاء تنفيذ الدالة.

207. ما هو الـ callback في JavaScript؟

الإجابة:
callback هو دالة تُمرر كمعامل إلى دالة أخرى ويتم تنفيذها بعد أن يتم تنفيذ الدالة الأصلية.

208. ما هو الـ debouncing؟

الإجابة:
debouncing هو تقنية تُستخدم لتأخير تنفيذ الدالة حتى يتم التوقف عن استدعائها لفترة معينة.

209. ما هو الـ throttling؟

الإجابة:
throttling هو تقنية تُستخدم للحد من عدد مرات تنفيذ الدالة في فترة زمنية معينة.

210. ما هو الـ setImmediate()؟

الإجابة:
setImmediate() يُستخدم لتنفيذ دالة بعد اكتمال الدورة الحالية من الأحداث في JavaScript.

211. ما هو الـ requestAnimationFrame()؟

الإجابة:
requestAnimationFrame() يُستخدم لطلب تحديث الرسوم المتحركة في الإطار التالي.

212. ما هو الـ module في JavaScript؟

الإجابة:
module هو كود JavaScript مُقسم إلى وحدات يمكن استيرادها واستخدامها في ملفات أخرى.

213. ما هو الـ export في JavaScript؟

الإجابة:
export يُستخدم لتصدير الكائنات أو الدوال أو المتغيرات من وحدة (module) ليتم استخدامها في وحدات أخرى.

214. ما هو الـ import في JavaScript؟

الإجابة:
import يُستخدم لاستيراد الكائنات أو الدوال أو المتغيرات من وحدات أخرى.

215. ما هو الـ default export في JavaScript؟

الإجابة:
default export يُستخدم لتصدير قيمة واحدة فقط من الوحدة.

216. ما هو الـ named export في JavaScript؟

الإجابة:
named export يُستخدم لتصدير عدة قيم من الوحدة مع أسمائها.

217. ما هو الـ Promise.all()؟

الإجابة:
Promise.all() يُستخدم لتنفيذ مجموعة من الـ Promises بالتوازي، ويُعيد Promise جديدًا يتم حله عندما يتم حل جميع الـ Promises.

218. ما هو الـ Promise.race()؟

الإجابة:
Promise.race() يُستخدم لتنفيذ مجموعة من الـ Promises ويُعيد Promise جديدًا يتم حله عندما يتم حل أول Promise في المجموعة.

219. ما هو الـ try...catch؟

الإجابة:
try...catch يُستخدم لمعالجة الأخطاء في JavaScript. يتم تنفيذ الكود داخل try وإذا حدث خطأ يتم تنفيذه داخل catch.

220. ما هو الـ finally في JavaScript؟

الإجابة:
finally يُستخدم لتنفيذ الكود بعد try...catch بغض النظر عن حدوث الخطأ أو لا.

221. ما هو الـ Symbol في JavaScript؟

الإجابة:
Symbol هو نوع بيانات يُستخدم لإنشاء قيم فريدة وغير قابلة للتكرار.

222. ما هو الـ Map في JavaScript؟

الإجابة:
Map هو كائن يُستخدم لتخزين أزواج من المفاتيح والقيم حيث يمكن أن تكون المفاتيح من أي نوع.

223. ما هو الـ Set في JavaScript؟

الإجابة:
Set هو كائن يُستخدم لتخزين القيم الفريدة فقط.

224. ما هو الـ WeakMap؟

الإجابة:
WeakMap هو نوع من الـ Map حيث المفاتيح تكون كائنات فقط ولا يتم الاحتفاظ بها في الذاكرة عند عدم استخدامها.

225. ما هو الـ WeakSet؟

الإجابة:
WeakSet هو نوع من الـ Set حيث القيم تكون كائنات فقط ولا يتم الاحتفاظ بها في الذاكرة عند عدم استخدامها.

226. ما هو الـ Array.isArray()؟

الإجابة:
Array.isArray() يُستخدم للتحقق مما إذا كان الكائن هو مصفوفة.

227. ما هو الـ Object.keys()؟

الإجابة:
Object.keys() يُستخدم للحصول على مصفوفة تحتوي على أسماء جميع خصائص الكائن.

228. ما هو الـ Object.values()؟

الإجابة:
Object.values() يُستخدم للحصول على مصفوفة تحتوي على قيم جميع خصائص الكائن.

229. ما هو الـ Object.entries()؟

الإجابة:
Object.entries() يُستخدم للحصول على مصفوفة تحتوي على أزواج من المفاتيح والقيم لجميع خصائص الكائن.

230. ما هو الـ spread operator؟

الإجابة:
spread operator يُستخدم لتفكيك المصفوفات أو الكائنات إلى عناصر منفصلة.

231. ما هو الـ rest parameter؟

الإجابة:
rest parameter يُستخدم لجمع عدد غير محدود من المعاملات في مصفوفة.

232. ما هو الـ destructuring في JavaScript؟

الإجابة:
destructuring هو طريقة لاستخراج القيم من المصفوفات أو الكائنات وتعيينها إلى متغيرات.

233. ما هو الـ set في CSS؟

الإجابة:
set في CSS ليس مصطلحًا معترفًا به، ولكن ربما يقصد به استخدام set في JavaScript مع CSS، مثل element.style.property = value.

234. ما هو الفرق بين let و var؟

الإجابة:
let يمتلك نطاق كتلة (block scope) بينما var يمتلك نطاق دالة (function scope).

235. ما هو الـ const؟

الإجابة:
const يُستخدم لتعريف المتغيرات التي لا يمكن إعادة تعيين قيمتها بعد الإعلان عنها.

236. ما هو الـ arguments في JavaScript؟

الإجابة:
arguments هو كائن يشبه المصفوفة يحتوي على جميع المعاملات التي تم تمريرها إلى دالة.

237. ما هو الـ eval() في JavaScript؟

الإجابة:
eval() يُستخدم لتنفيذ كود JavaScript مُخزن في سلسلة نصية.

238. ما هو الـ encodeURIComponent()؟

الإجابة:
encodeURIComponent() يُستخدم لتحويل سلسلة نصية إلى صيغة يمكن استخدامها في URL.

239. ما هو الـ decodeURIComponent()؟

الإجابة:
decodeURIComponent() يُستخدم لتحويل سلسلة نصية مشفرة في URL إلى النص الأصلي.

240. ما هو الـ encodeURI()؟

الإجابة:
encodeURI() يُستخدم لتحويل URI إلى صيغة صالحة للاستخدام في URL.

241. ما هو الـ decodeURI()؟

الإجابة:
decodeURI() يُستخدم لتحويل URI مشفر إلى النص الأصلي.

242. ما هو الـ setTimeout() في JavaScript؟

الإجابة:
setTimeout() يُستخدم لتنفيذ دالة بعد فترة زمنية محددة.

243. ما هو الـ setInterval() في JavaScript؟

الإابة:
setInterval() يُستخدم لتنفيذ دالة بشكل دوري بعد فترة زمنية محددة.

244. ما هو الـ clearTimeout()؟

الإجابة:
clearTimeout() يُستخدم لإلغاء دالة تم تحديدها باستخدام setTimeout().

245. ما هو الـ clearInterval()؟

الإجابة:
clearInterval() يُستخدم لإلغاء دالة تم تحديدها باستخدام setInterval().

246. ما هو الـ this في JavaScript؟

الإجابة:
this يشير إلى الكائن الذي تم استدعاء الدالة من خلاله.

247. ما هو الـ new في JavaScript؟

الإجابة:
new يُستخدم لإنشاء كائن جديد من دالة منشئ (constructor function).

248. ما هو الـ call() في JavaScript؟

الإجابة:
call() يُستخدم لاستدعاء دالة مع تحديد قيمة لـ this وتمرير المعاملات.

249. ما هو الـ apply() في JavaScript؟

الإجابة:
apply() يشبه call() ولكن يتم تمرير المعاملات كـ مصفوفة.

250. ما هو الـ bind() في JavaScript؟

الإجابة:
bind() يُستخدم لإنشاء دالة جديدة مع تعيين قيمة لـ this والمعاملات.

251. ما هو الـ constructor في JavaScript؟

الإجابة:
constructor هو دالة خاصة تُستخدم لإنشاء كائنات جديدة باستخدام دالة منشئ.

252. ما هو الـ instanceof في JavaScript؟

الإجابة:
instanceof يُستخدم للتحقق مما إذا كان الكائن هو من نوع معين.

253. ما هو الـ typeof في JavaScript؟

الإجابة:
typeof يُستخدم للحصول على نوع البيانات الخاص بالقيمة.

254. ما هو الـ toString() في JavaScript؟

الإجابة:
toString() يُستخدم لتحويل الكائن إلى سلسلة نصية.

255. ما هو الـ JSON.stringify()؟

الإجابة:
JSON.stringify() يُستخدم لتحويل الكائنات إلى سلسلة JSON.

256. ما هو الـ JSON.parse()؟

الإجابة:
JSON.parse() يُستخدم لتحويل سلسلة JSON إلى كائن JavaScript.

257. ما هو الـ localStorage؟

الإجابة:
localStorage يُستخدم لتخزين البيانات بشكل دائم في المتصفح.

258. ما هو الـ sessionStorage؟

الإجابة:
sessionStorage يُستخدم لتخزين البيانات فقط خلال جلسة التصفح الحالية.

259. ما هو الـ cookie؟

الإجابة:
cookie يُستخدم لتخزين البيانات على جهاز المستخدم.

260. ما هو الـ window في JavaScript؟

الإجابة:
window هو الكائن الذي يمثل نافذة المتصفح ويحتوي على العديد من الوظائف والخصائص.

261. ما هو الـ document في JavaScript؟

الإجابة:
document هو الكائن الذي يمثل محتوى صفحة الويب ويمكن التفاعل معه باستخدام JavaScript.

262. ما هو الـ event في JavaScript؟

الإجابة:
event هو كائن يُمثل حدثًا معينًا يحدث في المتصفح، مثل النقر أو التمرير.

263. ما هو الـ addEventListener()؟

الإجابة:
addEventListener() يُستخدم لإضافة مستمع (listener) لحدث معين.

264. ما هو الـ removeEventListener()؟

الإجابة:
removeEventListener() يُستخدم لإزالة مستمع لحدث معين.

265. ما هو الـ event.target؟

الإجابة:
event.target يُستخدم للحصول على العنصر الذي تم استدعاء الحدث عليه.

266. ما هو الـ event.preventDefault()؟

الإجابة:
event.preventDefault() يُستخدم لإلغاء السلوك الافتراضي للحدث.

267. ما هو الـ event.stopPropagation()؟

الإجابة:
event.stopPropagation() يُستخدم لإيقاف انتشار الحدث إلى العناصر الأب.

268. ما هو الـ querySelector()؟

الإجابة:
querySelector() يُستخدم لاختيار أول عنصر يتطابق مع المُحدد (selector).

269. ما هو الـ querySelectorAll()؟

الإجابة:
querySelectorAll() يُستخدم لاختيار جميع العناصر التي تتطابق مع المُحدد.

270. ما هو الـ classList؟

الإجابة:
classList هو كائن يحتوي على قائمة من الفئات (classes) الخاصة بالعنصر.

271. ما هو الـ getElementById()؟

الإجابة:
getElementById() يُستخدم لاختيار العنصر الذي يحمل معرف (ID) معين.

272. ما هو الـ getElementsByClassName()؟

الإجابة:
getElementsByClassName() يُستخدم لاختيار جميع العناصر التي تحمل فئة معينة.

273. ما هو الـ getElementsByTagName()؟

الإجابة:
getElementsByTagName() يُستخدم لاختيار جميع العناصر التي تحمل علامة معينة.

274. ما هو الـ createElement()؟

الإجابة:
createElement() يُستخدم لإنشاء عنصر جديد في DOM.

275. ما هو الـ appendChild()؟

الإجابة:
appendChild() يُستخدم لإضافة عنصر جديد كطفل للعنصر الحالي.

276. ما هو الـ removeChild()؟

الإجابة:
removeChild() يُستخدم لإزالة عنصر طفل من العنصر الحالي.

277. ما هو الـ insertBefore()؟

الإجابة:
insertBefore() يُستخدم لإضافة عنصر قبل عنصر آخر في DOM.

278. ما هو الـ cloneNode()؟

الإجابة:
cloneNode() يُستخدم لإنشاء نسخة طبق الأصل من عنصر DOM.

279. ما هو الـ innerHTML؟

الإجابة:
innerHTML يُستخدم للحصول على أو تعيين محتوى HTML داخل عنصر.

280. ما هو الـ outerHTML؟

الإجابة:
outerHTML يُستخدم للحصول على أو تعيين محتوى HTML بما في ذلك العنصر نفسه.

281. ما هو الـ textContent؟

الإجابة:
textContent يُستخدم للحصول على أو تعيين النص داخل عنصر، دون التأثير على الـ HTML.

282. ما هو الـ value في JavaScript؟

الإجابة:
value يُستخدم للحصول على أو تعيين قيمة عنصر النموذج مثل الحقول النصية أو القوائم المنسدلة.

283. ما هو الـ disabled في JavaScript؟

الإجابة:
disabled يُستخدم لتعطيل عنصر النموذج مثل الأزرار أو الحقول النصية.

284. ما هو الـ checked في JavaScript؟

الإجابة:
checked يُستخدم للتحقق مما إذا كان مربع الاختيار (checkbox) مُحددًا أم لا.

285. ما هو الـ selected في JavaScript؟

الإجابة:
selected يُستخدم للتحقق مما إذا كانت خيار معين في قائمة منسدلة (select) مُحددًا أم لا.

286. ما هو الـ focus() في JavaScript؟

الإجابة:
focus() يُستخدم لجعل عنصر معين في الصفحة يحصل على التركيز.

287. ما هو الـ blur() في JavaScript؟

الإجابة:
blur() يُستخدم لإزالة التركيز من عنصر معين.

288. ما هو الـ scrollIntoView()؟

الإجابة:
scrollIntoView() يُستخدم لتحريك الصفحة بحيث يظهر العنصر في العرض المرئي.

289. ما هو الـ match() في JavaScript؟

الإجابة:
match() يُستخدم للبحث عن تطابقات في سلسلة نصية باستخدام تعبير منتظم.

290. ما هو الـ replace() في JavaScript؟

الإجابة:
replace() يُستخدم لاستبدال جزء من السلسلة النصية بقيمة جديدة.

291. ما هو الـ split() في JavaScript؟

الإجابة:
split() يُستخدم لتقسيم السلسلة النصية إلى مصفوفة بناءً على فاصل معين.

292. ما هو الـ toLowerCase() في JavaScript؟

الإجابة:
toLowerCase() يُستخدم لتحويل جميع الأحرف في السلسلة النصية إلى حروف صغيرة.

293. ما هو الـ toUpperCase() في JavaScript؟

الإجابة:
toUpperCase() يُستخدم لتحويل جميع الأحرف في السلسلة النصية إلى حروف كبيرة.

294. ما هو الـ trim() في JavaScript؟

الإجابة:
trim() يُستخدم لإزالة المسافات الزائدة من بداية ونهاية السلسلة النصية.

295. ما هو الـ charAt() في JavaScript؟

الإجابة:
charAt() يُستخدم للحصول على الحرف الموجود في موضع معين داخل السلسلة النصية.

296. ما هو الـ concat() في JavaScript؟

الإجابة:
concat() يُستخدم لدمج سلسلتين نصيتين أو أكثر معًا.

297. ما هو الـ indexOf() في JavaScript؟

الإجابة:
indexOf() يُستخدم للبحث عن موضع أول تطابق في السلسلة النصية.

298. ما هو الـ includes() في JavaScript؟

الإجابة:
includes() يُستخدم للتحقق مما إذا كانت السلسلة النصية تحتوي على جزء معين.

299. ما هو الـ substring() في JavaScript؟

الإجابة:
substring() يُستخدم لاستخراج جزء من السلسلة النصية بين موضعين معينين.

300. ما هو الـ slice() في JavaScript؟

الإجابة:
slice() يُستخدم لاستخراج جزء من السلسلة النصية أو المصفوفة بناءً على الفهارس المحددة.

301. ما هو الـ map() في JavaScript؟

الإجابة:
map() يُستخدم لإنشاء مصفوفة جديدة تحتوي على نتائج تطبيق دالة معينة على كل عنصر في المصفوفة الأصلية.

302. ما هو الـ filter() في JavaScript؟

الإجابة:
filter() يُستخدم لإنشاء مصفوفة جديدة تحتوي على العناصر التي تستوفي شرطًا معينًا.

303. ما هو الـ reduce() في JavaScript؟

الإجابة:
reduce() يُستخدم لتقليص المصفوفة إلى قيمة واحدة بناءً على دالة معينة.

304. ما هو الـ forEach() في JavaScript؟

الإجابة:
forEach() يُستخدم لتنفيذ دالة معينة على كل عنصر في المصفوفة، ولكنه لا يُعيد قيمة.

305. ما هو الـ some() في JavaScript؟

الإجابة:
some() يُستخدم للتحقق مما إذا كان هناك عنصر واحد على الأقل في المصفوفة يستوفي شرطًا معينًا.

306. ما هو الـ every() في JavaScript؟

الإجابة:
every() يُستخدم للتحقق مما إذا كان جميع العناصر في المصفوفة تستوفي شرطًا معينًا.

307. ما هو الـ find() في JavaScript؟

الإجابة:
find() يُستخدم للعثور على أول عنصر في المصفوفة يستوفي شرطًا معينًا.

308. ما هو الـ findIndex() في JavaScript؟

الإجابة:
findIndex() يُستخدم للعثور على فهرس أول عنصر في المصفوفة يستوفي شرطًا معينًا.

309. ما هو الـ includes() في JavaScript؟

الإجابة:
includes() يُستخدم للتحقق إذا كانت المصفوفة تحتوي على عنصر معين.

310. ما هو الـ sort() في JavaScript؟

الإجابة:
sort() يُستخدم لترتيب العناصر في المصفوفة بترتيب تصاعدي أو تنازلي.

311. ما هو الـ reverse() في JavaScript؟

الإجابة:
reverse() يُستخدم لعكس ترتيب العناصر في المصفوفة.

312. ما هو الـ splice() في JavaScript؟

الإجابة:
splice() يُستخدم لإضافة أو إزالة عناصر من المصفوفة.

313. ما هو الـ slice() في JavaScript؟

الإجابة:
slice() يُستخدم لاستخراج جزء من المصفوفة دون تعديل المصفوفة الأصلية.

314. ما هو الـ concat() في JavaScript؟

الإجابة:
concat() يُستخدم لدمج مصفوفات متعددة في مصفوفة واحدة.

315. ما هو الـ join() في JavaScript؟

الإجابة:
join() يُستخدم لتحويل المصفوفة إلى سلسلة نصية، مع تحديد فاصل بين العناصر.

316. ما هو الـ push() في JavaScript؟

الإجابة:
push() يُستخدم لإضافة عنصر أو أكثر إلى نهاية المصفوفة.

317. ما هو الـ pop() في JavaScript؟

الإجابة:
pop() يُستخدم لإزالة العنصر الأخير من المصفوفة وإرجاعه.

318. ما هو الـ shift() في JavaScript؟

الإجابة:
shift() يُستخدم لإزالة العنصر الأول من المصفوفة وإرجاعه.

319. ما هو الـ unshift() في JavaScript؟

الإجابة:
unshift() يُستخدم لإضافة عنصر أو أكثر إلى بداية المصفوفة.

320. ما هو الـ for...in في JavaScript؟

الإجابة:
for...in يُستخدم للتكرار عبر خصائص الكائن.

321. ما هو الـ for...of في JavaScript؟

الإجابة:
for...of يُستخدم للتكرار عبر عناصر المصفوفة أو الكائنات القابلة للتكرار.

322. ما هو الـ try...catch في JavaScript؟

الإجابة:
try...catch يُستخدم للتعامل مع الأخطاء التي قد تحدث أثناء تنفيذ الكود.

323. ما هو الـ throw في JavaScript؟

الإجابة:
throw يُستخدم لرمي استثناء (استثناء أو خطأ) داخل الكود.

324. ما هو الـ finally في JavaScript؟

الإجابة:
finally يُستخدم لتنفيذ كود بعد try و catch، سواء حدث خطأ أم لا.

325. ما هو الـ Promise.all() في JavaScript؟

الإجابة:
Promise.all() يُستخدم لتنفيذ مجموعة من الـ Promises في نفس الوقت والانتظار حتى يتم الانتهاء من جميع الـ Promises.

326. ما هو الـ Promise.race() في JavaScript؟

الإجابة:
Promise.race() يُستخدم للانتظار حتى يتم الانتهاء من أول Promise في مجموعة من الـ Promises.

327. ما هو الـ setImmediate() في JavaScript؟

الإجابة:
setImmediate() يُستخدم لتنفيذ دالة بعد اكتمال الدورة الحالية للـ event loop.

328. ما هو الـ nextTick() في JavaScript؟

الإجابة:
nextTick() يُستخدم لتنفيذ دالة في الدورة القادمة للـ event loop، قبل أي عملية أخرى.

329. ما هو الـ Function.prototype.call() في JavaScript؟

الإجابة:
call() يُستخدم لاستدعاء دالة مع تحديد قيمة this وتمرير المعاملات بشكل فردي.

330. ما هو الـ Function.prototype.apply() في JavaScript؟

الإجابة:
apply() يُستخدم لاستدعاء دالة مع تحديد قيمة this وتمرير المعاملات كمصفوفة.

331. ما هو الـ Function.prototype.bind() في JavaScript؟

الإجابة:
bind() يُستخدم لإنشاء دالة جديدة مع تحديد قيمة this والمعاملات.

332. ما هو الـ eval() في JavaScript؟

الإجابة:
eval() يُستخدم لتنفيذ الكود النصي المرسل كـ سلسلة نصية.

333. ما هو الـ with() في JavaScript؟

الإجابة:
with() يُستخدم لتوسيع نطاق الكود ليشمل كائنًا معينًا، ولكن يُنصح بتجنبه بسبب قلة الأمان.

334. ما هو الـ window.location في JavaScript؟

الإجابة:
window.location يُستخدم للحصول على أو تعيين عنوان URL الحالي للصفحة.

335. ما هو الـ window.history في JavaScript؟

الإجابة:
window.history يُستخدم للتفاعل مع سجل التصفح (مثل العودة إلى الصفحة السابقة أو الانتقال إلى الصفحة التالية).

336. ما هو الـ window.navigator في JavaScript؟

الإجابة:
window.navigator يُستخدم للحصول على معلومات حول المتصفح الذي يستخدمه المستخدم.

337. ما هو الـ window.alert() في JavaScript؟

الإجابة:
window.alert() يُستخدم لعرض رسالة منبثقة تحتوي على نص معين.

338. ما هو الـ window.confirm() في JavaScript؟

الإجابة:
window.confirm() يُستخدم لعرض مربع حوار يحتوي على رسالة وأزرار "موافق" و "إلغاء".

339. ما هو الـ window.prompt() في JavaScript؟

الإجابة:
window.prompt() يُستخدم لعرض مربع حوار يحتوي على حقل نصي لادخال قيمة من المستخدم.

340. ما هو الـ document.write() في JavaScript؟

الإجابة:
document.write() يُستخدم لكتابة محتوى مباشرة في مستند HTML أثناء تحميل الصفحة.

341. ما هو الـ document.createElement() في JavaScript؟

الإجابة:
document.createElement() يُستخدم لإنشاء عنصر HTML جديد.

342. ما هو الـ document.getElementById() في JavaScript؟

الإجابة:
document.getElementById() يُستخدم للحصول على عنصر HTML باستخدام معرفه (ID).

343. ما هو الـ document.getElementsByClassName() في JavaScript؟

الإجابة:
document.getElementsByClassName() يُستخدم للحصول على جميع العناصر التي تحتوي على فئة معينة.

344. ما هو الـ document.getElementsByTagName() في JavaScript؟

الإجابة:
document.getElementsByTagName() يُستخدم للحصول على جميع العناصر التي تحمل علامة معينة.

345. ما هو الـ document.querySelector() في JavaScript؟

الإجابة:
document.querySelector() يُستخدم للحصول على أول عنصر يتطابق مع المُحدد.