React (1 - 34)

1. **React ما هو؟**
   مكتبة JavaScript لبناء واجهات المستخدم باستخدام components.
2. **JSX ما هو؟**
   امتداد لـ JavaScript بيسمحلك تكتب HTML داخل JavaScript.
3. **component شو يعني؟**
   جزء مستقل من الواجهة، ممكن تعيدي استخدامه.
4. **function component و class component شو الفرق بين؟**
   function component أبسط، class component فيها lifecycle methods.
5. **virtual DOM شو هو؟**
   نسخة خفيفة من DOM الحقيقي، React بتستخدمها للمقارنة والتحديث السريع.
6. **state شو هو؟**
   بيانات بتتغير مع الوقت داخل component.
7. **props شو هو؟**
   بيانات بنمررها للـ component من خارجها.
8. **props هل ممكن نعدل على؟**
   لا، props للقراءة فقط.
9. **props و state شو الفرق بين؟**
   props خارجية وما بتتغير، state داخلية وبتتغير.
10. **useState شو هو؟**
    hook بتستخدمه لإدارة state داخل function components.
11. **useEffect شو هو؟**
    hook بتستخدمه لتنفيذ side effects مثل جلب البيانات.
12. **useEffect كيف أستدعي مرة وحدة فقط؟**
    بتحطي مصفوفة فاضية [] كـ dependency.
13. **lifting state up شو يعني؟**
    نقل state للـ parent component عشان يتشارك بين عدة أطفال.
14. **controlled component شو هو؟**
    مثلاً state العنصر الي بياناته بتيجي من input ، مربوط بـ useState.
15. **uncontrolled component شو هو؟**
    العنصر الي بياناته بتيجي من DOM مباشرة.
16. **useRef شو هو؟**
    hook بيخزن قيمة ما بتسبب إعادة render لما تتغير.
17. **useRef و useState شو الفرق بين؟**
    useState بيسبب render، useRef ما بيسبب.
18. **React Fragment شو هو؟**
    عنصر وهمي بستخدمه عشان ألف أكثر من عنصر بدون إضافة عنصر حقيقي للـ DOM.
19. **lists بالـ key شو هو؟**
    معرف فريد بيساعد React يميز العناصر لما تتغير.
20. **keys ليش لازم نستخدم؟**
    لتحسين الأداء ومنع أخطاء عند التحديث.
21. **context شو هو؟**
    طريقة لتمرير بيانات بين components بدون استخدام props.
22. **useContext شو هو؟**
    hook بيسمحلك تستخدم context داخل component.

23. **شو هو custom hook؟**
    function بتبدأ بـ use وبتحتوي logic قابل لإعادة الاستخدام.
24. **شو هو React Router؟**
    مكتبة لإدارة التنقل بين الصفحات في تطبيق React.
25. **شو الفرق بين BrowserRouter و HashRouter؟**
    BrowserRouter يستخدم URL الحقيقي، HashRouter يستخدم الـ #.
26. **شو هو Link component؟**
    زي `<a>` بس من React Router بغير الصفحة بدون إعادة تحميل.
27. **شو هو lazy loading؟**
    تحميل components عند الحاجة فقط لتحسين الأداء.
28. **كيف أعمل lazy loading؟**
    باستخدام `React.lazy()` و`Suspense`.
29. **شو الفرق بين useMemo و memo؟**
    `useMemo` لتقليل إعادة حساب قيم، `memo` لتقليل render للـ component.
30. **شو هو useCallback؟**
    hook بيمنع إعادة إنشاء function في كل render.
31. **شو هو HOC (Higher Order Component)؟**
    function بتاخد component وبترجع واحد جديد مع ميزات إضافية.
32. **كيف نحدث state داخل object أو array؟**
    باستخدام spread operator أو methods بتعطي نسخة جديدة.
33. **شو هو reconciliation؟**
    عملية مقارنة virtual DOM مع DOM الحقيقي وتحديثه.
34. **شو هو StrictMode؟**
    أداة من React بتساعدك تكتشف المشاكل المحتملة بالتطبيق.

◆ Next.js (35 - 67)

35. **شو هو Next.js؟**
    Framework مبني على React بيدعم SSR (Server Side Rendering) و SSG (Static Site Generation).
36. **شو الفرق بين SSG و SSR؟**
    - SSR: الصفحة بتتولد عند كل طلب.
    - SSG: الصفحة بتتولد وقت بناء التطبيق.
37. **شو هي file-based routing؟**
    كل ملف داخل مجلد `pages/` بصير route تلقائي.
38. **شو هو dynamic routing؟**
    صفحات فيها متغيرات زي `[id].js`.
39. **شو هو API route؟**
    ملفات داخل `pages/api/` بتمثل endpoints للسيرفر.
40. **شو الفرق بين getServerSideProps و getStaticProps؟**

- `getStaticProps`: لـ SSG.
- `getServerSideProps`: لـ SSR.

41. **شو هو getStaticPaths؟**
    function بتحدد القيم المحتملة للديناميك routes وقت البناء.
42. **شو هو middleware؟**
    كود بيتنفذ بين الطلب والاستجابة، ممكن تستخدمه للحماية أو إعادة التوجيه.
43. **شو الفرق بين _app.js و _document.js؟**
    - `_app.js`: لتعديل التطبيق ككل، مثلاً إضافة Layout.
    - `_document.js`: لتعديل HTML الأساسي مثل `<html>` و `<body>`.
44. **شو هي image optimization في Next.js؟**
    تحسين صور تلقائي باستخدام `<Image />` component.
45. **ليش نستخدم `<Image>` بدل `<img>`؟**
    لأنه بيدعم lazy loading وتحسين الأداء.
46. **شو الفرق بين Link و useRouter؟**
    - `Link`: للتنقل بين الصفحات.
    - `useRouter`: hook بتعطيك معلومات عن المسار الحالي.
47. **كيف أعمل redirect؟**
    باستخدام `next.config.js` أو `router.push()`.
48. **شو الفرق بين static file في Next.js و public folder؟**
    ملفات `public/` بتكون متاحة مباشرة للمستخدم بدون import.
49. **هل ممكن أستخدم React hooks عادي داخل Next.js؟**
    نعم، لأنه Next.js مبني على React.
50. **شو هو ISR (Incremental Static Regeneration)؟**
    تحديث الصفحات الثابتة بعد النشر بدون إعادة بناء التطبيق بالكامل.
51. **شو هو app router الجديد في Next.js؟**
    نظام توجيه جديد بيستخدم مجلد `app/` بدلاً من `pages/`.
52. **شو الفرق بين app router و pages router؟**
    - app router بيدعم layouts ،server components ،loading ،error.
    - pages router هو النظام القديم المعتمد على `pages/`.
53. **شو هي server components؟**
    بتشتغل بالسيرفر، بتقلل حجم الجافاسكربت على المتصفح components.
54. **شو هي client components؟**
    بالأعلى `use client` بتشتغل بالمتصفح، لازم نكتب components.
55. **كيف أعمل layout دائم لكل الصفحات؟**
    باستخدام ملفات layout.js في مجلد `app/`.
56. **كيف أضيف metadata للصفحات؟**
    باستخدام `metadata` داخل page component أو ملفات `head.js`.
57. **شو الفرق بين error.js و loading.js؟**
    - loading.js: صفحة تحميل مؤقتة.
    - error.js: صفحة تظهر عند حدوث خطأ.
58. **كيف أعمل API requests داخل Next.js؟**
    باستخدام fetch أو Axios داخل server functions أو client components.
59. **هل لازم أستخدم Express مع Next.js؟**
    لا، Next.js بيحتوي على سيرفر خاص فيه.
60. **شو هو next.config.js؟**
    ملف إعدادات مخصص لتعديل إعدادات المشروع.

61. **كيف أضيف Tailwind CSS لمشروع Next.js؟**
    عن طريق تثبيت Tailwind وتعديل `postcss.config.js` و`globals.css`.
62. **هل Next.js مناسب لتطبيقات كبيرة؟**
    نعم، بيدعم الأداء العالي والتنظيم.
63. **شو هو usePathname؟**
    hook من app router بتعطيك اسم المسار الحالي.
64. **كيف أستخدم params داخل dynamic routes في app router؟**
    باستخدام `generateStaticParams` و`params` كـ props للصفحة.
65. **هل بقدر أستخدم MongoDB أو أي Database مع Next.js؟**
    أكيد، من خلال API routes أو server actions.
66. **كيف أستخدم environment variables؟**
    بضيفها في ملف `.env.local` وبستخدم `process.env.VARIABLE_NAME`.
67. **هل ممكن أعمل authentication داخل Next.js؟**
    نعم، من خلال NextAuth أو Auth0 أو حل مخصص.

◆ Node.js (68 - 100)

68. **شو هو Node.js؟**
    بيئة تشغيل JavaScript مبنية على السيرفر على V8 engine.
69. **ليش نستخدم Node.js؟**
    لأنه سريع، خفيف، وبيشتغل Asynchronously.
70. **شو الفرق بين asynchronous و synchronous؟**
    - synchronous: الأوامر بتنفذ بالتسلسل.
    - asynchronous: الأوامر ممكن تنفذ بنفس الوقت بدون توقف.
71. **شو هو npm؟**
    مدير الحزم الرسمي لـ Node.js، بنستخدمه لتثبيت البكجات.
72. **شو هو package.json؟**
    ملف يحتوي معلومات المشروع والبكجات المستخدمة فيه.
73. **شو الفرق بين dependencies و devDependencies؟**
    - dependencies: بتحتاجها وقت التشغيل.
    - devDependencies: بتحتاجها وقت التطوير فقط.
74. **شو هي modules في Node.js؟**
    ملفات فيها كود قابل لإعادة الاستخدام.
75. **كيف أستورد module؟**
    `const x = require('module')` أو باستخدام `import`.
76. **شو هو built-in module؟**
    مدمج بـ Node.js مثل fs، path، http.
77. **شو هو fs module؟**
    للتعامل مع الملفات (قراءة، كتابة، حذف…) module.

78. **كيف أعمل سيرفر باستخدام Node.js؟**
باستخدام http module:

```
js
CopyEdit
const http = require('http');
const server = http.createServer((req, res) => {
  res.end('Hello');
});
server.listen(3000);
```

79. **شو الفرق بين import و require؟**
    - require: syntax قديم (CommonJS).
    - import: syntax حديث (ES Modules).
80. **شو هو Express.js؟**
framework خفيف مبني على Node.js لتسهيل بناء APIs.
81. **ليش نستخدم Express؟**
لأنه بسيط، سريع، وسهل التنظيم.
82. **كيف أعمل API endpoint باستخدام Express؟**

```
js
CopyEdit
app.get('/api', (req, res) => {
  res.json({ message: 'Hello' });
});
```

83. **شو هو middleware في Express؟**
function بتتنفذ بين الطلب والاستجابة.
84. **شو هو body-parser؟**
middleware بيساعدنا نقرأ البيانات من body في الطلبات.
85. **شو هو req و res؟**
    - req: الطلب القادم من المستخدم.
    - res: الاستجابة اللي بنرسلها للمستخدم.
86. **كيف أتعامل مع JSON في Express؟**
باستخدام`express.json()` كـ middleware.
87. **شو هو route parameter؟**
متغير بالمسار مثل`/user/:id`.
88. **كيف أتعامل مع route params؟**
باستخدام`req.params`.
89. **كيف أتعامل مع query params؟**
باستخدام`req.query`.
90. **شو هو CORS؟**
إعداد أمني بمنع أو السماح بالوصول من دومينات مختلفة.
91. **شو هي البيئة (environment)؟**
مجموعة متغيرات بتحدد إعدادات التشغيل (مثل port ،mode...).
92. **كيف أستخدم .env file؟**
بضيف القيم داخله وبقرأهم باستخدام`process.env`.

93. **شو هو MongoDB؟**
    قاعدة بيانات NoSQL مرنة بتخزن البيانات كـ JSON-like documents.
94. **شو هو Mongoose؟**
    مكتبة بربط Node.js مع MongoDB بطريقة منظمة.
95. **شو هو schema في Mongoose؟**
    تعريف لشكل البيانات داخل document.
96. **كيف أعمل اتصال مع MongoDB؟**

```
js
CopyEdit
mongoose.connect(process.env.MONGO_URL)
```

97. **شو هو async/await؟**
    بيساعدك تكتب كود asynchronous بشكل مرتب وسهل القراءة syntax.
98. **شو الفرق بين PUT و PATCH؟**
    - PUT: تعديل كل الكائن.
    - PATCH: تعديل جزء من الكائن.
99. **كيف أتعامل مع الأخطاء؟**
    خاص للأخطاء middleware أو try/catch باستخدام.
100. **كيف أعمل API كاملة باستخدام Node و Express؟**
    بعمل routes ،middlewaresمثل وربطها مع قاعدة البيانات MongoDB. ،