מבחן בריאקט

1.מהירות טעינה תלויה בקלינט ובסרוור

בקלינט הסיבות יכולות להיות:

\*טעינה מיותרת

\*טעינה של חומרים שהאתר מציג מאוחר יותר

\*קריאות שרת מיותרות שימוש לא נכון בcechin

\*שרת קטן מרוחק חלש

DOM\* עמוס מאוד

בסרוור הסיבות יכולות להיות

\*פעולות שאינם קריטיות לקריאה (העלולות להפיל גם את הSetTimeOut)

\*שימוש בתור כגון RabbitMQ יכול ליעל את הקריאות

2. הערך לא משתנה ולכן צריך לעשות useState והנתונים לא נשלפו שוב כי אף אחד לא דיווח על שינוי בנתונים

3. Class ופונקציה

4.הרחבת תחביר עבורSJ ומייצר אלמנטים של ריאקט

5. Virtual DOM?! הוא בעצם ייצוג בזיכרון, אובייקט JS קל משקל שהוא העתק של הReal DOM

6. **קוד פתוח**- React הינו פרויקט העובד במתכונת קוד פתוח מה שלמעשה אומר כי כל מפתח מרחבי העולם יכול לשפר ולהרחיב את היכולות של השפה בחינם, כמו כן ישנו רישיון המעניק הגנה למפתחים – מה שנותן ביטחון עבור לא מעט מפתחים שונים בפלטפורמה.

**קהילה גדולה**- כאשר אנחנו מדברים על React אנו רואים קהילה גדולה ומגוונת אשר פועלת ללא לאות על מנת לעזור אחד לשני תוך כדי ערבות הדדית והמון רצון טוב.

**בשימוש נרחב אצל חברות רבות**-  React פותחה על ידי פייסבוק בשנת 2013 והפכה כבר מזמן לשם ודבר בקרב חברות גדולות אחרות בעולם לדוגמת אינסטגרם, טסלה, סקייפ ושאר חברות אחרות. הפרויקט נחשב לאחד הפרויקטים המוצלחים ביותר של השנים האחרונות וכמעט ולא ניתן לראות חברות הייטק, בתי תוכנה ו/או סטרטאפים בהתהוות אשר לא עובדים עם שפת התכנות הפופולארית.

מהירות- בדיקות שנעשו הוכיחו כי React מציגה ביצועים טובים ואיכותיים הרבה יותר משפות תכנות וסביבות עבודה שונות.

עבודה עם JS כשפת תכנות עיקרית-React נכתבה ועובדת יחד עם אחת מן שפות התכנות הפופולאריות ביותר בעולם הלוא היא Java Script. כל אלו הופכים אותה לבחירה הטבעית של מיליוני אנשי פיתוח ברחבי העולם.

7.ישנם פונקציות כמו onClick onChange

8. פונקצית map וחשוב לזכור לעשות key

9. ראוטינג הוא ניווט בין "עמודים" בתוך האפליקציה שלנו, כמובן שהם לא ״עמודים״ אמיתיים, מכיוון שמדובר ב- SPAאפליקציה שהיא בעצם דף אחד, אבל מבחינת המשתמש זה נראה שכן. מכיוון שאיננו טוענים דפים חדשים מהשרת, איננו יכולים להשתמש בניווט הדפדפן הרגיל אלא צריכים לספק בעצמנו את ה״דפים״

ריאקט לא מגיעה עם ראוטר מובנה, אך אנו יכולים להשיג ראוטינג עם הספריה

 React-router-dom

10. \*בכל שינוי בטופס הסטייט מתעדכן והvalue שווה לסטייט

\*בחירה

11. \*זיכרון הפנימי של הקומפוננטה הסטייט מסמן לריאקט מתי לרנדר את הקומפוננטה

לדוגמא const [i, setI] = useState(0) כדי לשנות את i נעשה setI(5)

\* כדי להשתמש בסטייט בקלאס נגדיר בקונסטרקטור:

this.state = { greeting: `Hello ${props.name}!` }

כדי לשנות נעשה

this.setState({ greeting: `Hello ${props.name}!` })

12. ע"י propse או ע"י קונטקסט שהוא שיתוף מידע גלובלי

13. בפונקציות ע"י useEffect - useEffect (()=>{});

בקומפוננטה מסוג קלאס- ע"י הפונקציה componentDidUpdate שזה ארוע מחזור חיים של הקומפוננטה הפונקציה מופעלת ברגע שה-props מתעדכנים או ה- setState מופעל.

14. ע"י קונטקסט

15. הוקים הם פונקציות שניתן להשתמש בהם בקומפוננטות פונקציונליות בלבד. יש להקפיד ליישם את ההוקים בחלק העליון של הקומפוננטה. הוקים שנמצאים בתוך תנאים או פונקציות פנימיות עלולים לגרום לבאגים, ולחוסר סדר והגיון בקוד. או שהם לא יתקיימו בכלל. סוגי הוקים לדוגמא: useState , useEffect

16.ג'אווהסקריפט יריץ קודם את לולאת הפור וידפיס: current number is +המספר הנוכחי מיליון פעמים עד שיגמור. ואח"כ יריץ את ה setTimeOut ויכתוב: in time out. והסיבה היא שהמנגנון שמטפל בפעולות האסינכרוניות בג'אווהסקריפט גומר לבצע את הפעולות לפי הסדר, ואז מריץ פקודות אסינכרוניות כמו setTimeOut

17.מנגנון של ג'אווהסקריפט שמטפל בסדר הרצת הפקודות בקוד. כל הפקודות הסינכרונית נכנסות ל Stack – מחסנית ומורצות לפי הסדר בו הם כתובות. פקודות אסינכרוניות, לעומת זאת, נכנסות ל - Queueתור. ברגע שמתרוקנת המחסנית – המנוע גמר להריץ את כל הפקודות הסינכרוניות לפי הסדר, מועברות הפקודות האסינכרוניות לפי הסדר מהתור אל המחסנית, ומתבצעות לפי הסדר בו נכנסו לתור. ה Event-Loop בודק כל הזמן האם המחסנית ריקה, ואם כן מכניס אליה את האוביקטים הממתינים בתור.

18. ישנן שלש אפשרויות להגדרת משתנים בג'אווהסקריפט let, const, ו var:

Const: זהו משתנה קבוע שניתן לאתחל אותו רק בעת ההצהרה עליו.

Let: משתנה רגיל

Let וconst מוכרים רק ב block scope בו הם מוגדרים וא"א להגדיר 2 משתנים באותו שם.

שתיהם נוספו בעדכון של es6

Var: אינו נתמך ב block scope מוכר בכל הפרויקט וניתן לדרוס אותו ע"י הגדרת משתנה נוסף באותו שם

19 . בחירה!

20. promise הוא אובייקט של JavaScript  שבתוכו אנחנו מכניסים קוד, שמחזיר אחת משתי תוצאות, הצלחה או כישלון. הרבה פעמים משתמשים ב- promise עבור קוד א-סינכרוני.

התחביר של Promise :

new Promise((resolve, reject) => {});

21. לבנות פיז'נטור שיציג כמות של נתונים בדף ואז לטעון רק את הנתונים שמוצגים בדף ולא לשלוף את הנתונים שאין בהם עוד צורך ושיעבור לדף הבא יטען את הנתונים שנצרך להם.

22.

a. true

b. false

c. '22'

d. '22'

e. 4