Please answer the questions below.

Where we ask for you to provide code, this code can just be ‘sudo code’ and does not have to be executable.

The idea of these questions are basically just to get an idea of your understanding. You are allowed to skip questions if you cannot answer but of course the more you can answer the better.

Completing all the tasks should not take more than an hour to an hour and a half. There are 20 questions in total, most of which only require textual answers.

Please create a repo on github to host your answers and test code. You can either do this on your own public repo or share ‘kkroese’ and ‘devblazer’ on your private github repo. Please do not include any OS specific files like microsoft word documents. Also do not commit binary files like zip, tar or gz files.

Please send the link to your repository in the chat on Upworks.

Javascript:

1. What is your favourite new javascript feature and why?

Ans.  
Array functions  
Array functions like map, filter, forEach make things simpler and easy to iterate through objects / arrays and makes searching very easy.

1. Explain an interesting way in which you have used this javascript feature.

Ans. Filter function used to look for duplicates in an array, returning duplicates based on a specific key.

1. Is there any difference between regular function syntax and the shorter arrow function syntax? (Write the answer in your own words)  
   Ans.
   1. Syntax of arrow function is different
   2. the keyword `this`: Arrow functions do not have their own `this`
   3. `Arguments` object: not available in arrow functions, but available in regular functions
2. What is the difference between ‘myFunctionCall(++foo)’ and ‘myFunctionCall(foo++)’

Ans. ++foo returns the incremented value however foo++ returns the same value

1. In your own words, explain what a javascript ‘class’ is and how it differs from a function.  
   Ans. A function consists of a piece of code so that it can be reused, while a class is a “blueprint” for an object, something that contains the definition of what the object can do.

Css:

1. In your own words, explain css specificity.

Ans: CSS specificity is the measure of how specific the selector of an HTML element is. The selector with the highest specificity is applied to the HTML element over other less specific ones and when multiple declarations have equal specificity, the last declaration found in CSS is applied to the element. But when an important rule(!important) is used on a style declaration, this declaration overrides any other declarations, however, is bad practice and should be avoided.

One most basic example would be:

#highest { background-color: red; }

.higher { background-color: gray; }

p { background-color: blue; }

1. In your own words, explain, what is ‘!important’ in css. Also how does it work? Are there any special circumstances when using it, where it’s behaviour might not be what you expect?

Ans: !important keyword is used after value to a property and is used to override all the existing values given to single property. As defined in point 1 !important at first breaks the specificity of an HTML element but if each declaration consist of !important value of the same property then again it works as per the specificity defined. Like for an instance:

Considering the example:

#highest { background-color: red; }

.higher { background-color: gray; }

p { background-color: blue !important; }

- It will make the same element of background color blue but if we change this to

#highest { background-color: red !important; }

.higher { background-color: gray !important; }

p { background-color: blue !important; }

then it will again make the same element of background color red.

1. What is your prefered layout system: inline-block, floating + clearing, flex, grid, other? And why?

Ans: Flex, personally I noticed that line of codes while writing flex based CSS is comparatively less than what we write in any other layouts.

Also one major advantage is ability to fill extra space without the need to use Javascript.

For an instance, if I have to align an item vertically and horizontally center to its parent then I have to simple write 3 line of codes ie. 'display: flex; justify-content: center; align-items: center' but if same thing I have to do in block based CSS then

1. Are negative margins legal and what do they do (margin: -20px)?

Ans: Negative margins are considered valid but a bad practise as it makes the code complex and confusing as to identify certain parameters of an element like if overlaped one element on another by using negative margin then on inspecting the same it will take some time to identify the placement of div which is now behind the current one.

margin: -20px will basically affect the model box. The margin box lost -20px on the top, right, bottom and left however this does not affect the content & padding boxes. Because your element is absolutely positioned from the relative document, moving the element -20px from all sides create UI disturbance.

1. If a <div/> has no margin or other styling and a <p/> tag inside of it has a margin top of some kind, the margin from the <p/> tag will show up on the div instead (the margin will show above the div not inside of it), why is this? What are the different things that can be done to prevent it?

Ans: In box model, if we combine two margins of different elements then combinely they make a single margin known as collapsed margin. It happens with block level elements. So inorder to prevent it following ways can be used:

1) Make p element inline-block (`as inline-block level elements do not face collapsed margin problem`)

2) Give the parent div 'float: left; width: 50%;' as it clarifies parent div its original box model

3) Instead of margin-top we can give padding-top to p tag (`just a backup option not to be used ideally`)

Unit tests:

1. What technologies do you use to unit test your react components?  
   Ans: Jest, enzyme & React-testing-library for unit test cases & also jest puppeteer for end to end test cases.
2. Are there any pitfalls associated with this technology that have caused you difficulty in the past?

Ans: Yes so when earlier I started working on React version 15 it was all es5 with React.createClass and now since it has evolved, it's difficult to work on existing project as all associated modules & packages gets updated. Similar issue I faced with hooks & context api's when I started using it on my project in prod mode when they were in beta release. Rest the current version of React 16.13.0 works perfect and robust.

1. How do you test in your unit tests to see if the correct properties are being passed to child components.

Ans: In unit tests cases we can check if the correct properties are being passed to child components with toHaveProperty method.

Syntax: expect(`some component`).toHaveProperty('disabled', false); Similarly we can check toHaveProperty to check props beings passed along with using toBe to check change in value of properties on action events.

React:

1. React test step1:  
     
   Create a react component that has a <div/> with a border.  
   Inside this <div/> should be a <span/> that displays the ‘live’ width of the browser window at all times. Keep in mind that the size of the window could easily be changed by the user and you should reflect this.

Ans: """

import React, { useEffect } from 'react';

const BrowserSize = () => {

const [width, setWidth] = React.useState(window.innerWidth);

const updateWidth = () => {

setWidth(window.innerWidth);

};

useEffect(() => {

window.addEventListener("resize", updateWidth);

return () => window.removeEventListener("resize", updateWidth);

});

return (

<div>

<span>{`Live width: ${width}`}</span>

</div>

);

}

export default BrowserSize;

"""

1. React test step2:  
     
   Inside the <div/> you created in the previous step, add a text input that, as a number is entered into it, uses that number to set the height of the div itself in pixels, live as you update the text field (keypress not change event).

Ans:

'''

import React, { useEffect, useState } from 'react';

const BrowserSize = () => {

const [width, setWidth] = useState(window.innerWidth);

const [divHeight, setDivHeight] = useState(40);

const updateWidth = () => {

setWidth(window.innerWidth);

};

const handleChange = (e) => {

e.preventDefault();

let value = e.target.value;

setDivHeight(value)

}

useEffect(() => {

window.addEventListener("resize", updateWidth);

return () => window.removeEventListener("resize", updateWidth);

});

return (

<div className='background' style={{'height': divHeight+'px'}}>

<div>

Height of div:&nbsp;

<input type="number" value={divHeight} onChange={(e) => handleChange(e)}/>

</div>

<span>{`Live width: ${width}`}</span>

</div>

);

}

export default BrowserSize;

'''

1. React test step3:  
     
   Add the following code to your project root (same project as in step 2, but add the code in the global / window space):   
     
    Let divHeight;  
    window.setDivHeight = (height) => divHeight = height;  
     
   Add a HOC for your div component that allows you to set the height of your <div/> component from the previous steps by calling that external function.  
     
   If you do not know what a HOC is or how to create one, that is also fine, just mention that in your answer and instead create a parent component that can still do this (allow you to call that function ‘setDivHeight’ in order to set the height of the div manually.  
     
   Bare in mind that when the height of the div is forcefully set like this, the text fields value should also update to reflect this and should still carry on working as normal (user can continue to modify its value).

Ans.

'''''''''''''''''''''

import HandleDivHeightHOC from './handleDivHeightHOC';

import BrowserSize from './browserSize';

const DynamicBrowserSize = () => HandleDivHeightHOC(BrowserSize);

export default DynamicBrowserSize;

===================================

import React, { useState } from 'react';

const HandleDivHeightHOC = (Component) => {

const [dynamicDivHeight, setDivHeight] = useState(40);

const handleChangeDiv = (value) => {

setDivHeight(value)

}

return (

<>

<Component handleChangeDiv={handleChangeDiv} dynamicDivHeight={dynamicDivHeight}/>

</>

);

}

export default HandleDivHeightHOC;

======================================

import React, { useEffect, useState } from 'react';

const BrowserSize = (props) => {

const { handleChangeDiv, dynamicDivHeight } = props;

const [width, setWidth] = useState(window.innerWidth);

const [divHeight, setDivHeight] = useState(40);

const updateWidth = () => {

setWidth(window.innerWidth);

};

const handleChange = (e) => {

e.preventDefault();

let value = e.target.value;

if(handleChangeDiv){

handleChangeDiv(value);

}else{

setDivHeight(value);

}

}

useEffect(() => {

window.addEventListener("resize", updateWidth);

return () => window.removeEventListener("resize", updateWidth);

});

const height = dynamicDivHeight ? dynamicDivHeight : divHeight;

return (

<div className='background mb-20' style={{'height': height+'px'}}>

<div>

Height of div {dynamicDivHeight ? 'with HOC' : ''}:&nbsp;

<input type="number" value={height} onChange={(e) => handleChange(e)}/>

</div>

<span>{`Live width: ${width}`}</span>

</div>

);

}

export default BrowserSize;

''''''''''''''''''''''''''''

RXjs:

1. What are the differences between Subject, BehaviorSubject and ReplaySubject? And in what situation would you use each of these (please provide example scenarios)?
   1. A subject is a special object that allows you to emit data to the stream and control it. Values emitted before subscriptions are “lost”. Used when you do not want to retain the emitted value like a field to be displayed in a specific case on a page.
   2. The BehaviorSubject allows you to provide a default value that will be emitted to every subscriber if no other value has been emitted so far. Otherwise subscribers receive the last emitted value. Used when the last emitted value needs to be used across the application at multiple places, like using application settings throughout the application.
   3. The ReplaySubject stores all emitted values up to a certain number, time or infinitely. All new subscribers will then get all stored values. Used when you want to store the history of actions taken by a user in that particular session, this will store all actions for example last 5 actions of a user.
2. If you have an array of values in a stream and you wish to pipe it such that it will emit the arrays values individually, one by one and wait for them all to be completed before processing another array, how would you do this? Please provide a code example.  
   E.g.  
   [1,2] -> [3,4] -> [5,6,7]  
   Should produce:  
   1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7
3. If you have a stream that receives individual values and would like to pipe it such that it builds an array out of these values, emitting the updated array each time a new value is added to it, how would you do this? Please provide a code example.  
   E.g.  
   1 -> 2 -> 3 -> 4  
   Should produce:  
   [1] -> [1,2] -> [1,2,3] -> [1,2,3,4]

Twilio:

1. Explain which of the Twilio Api’s you have used. Also explain how and in what scenarios you have used them.
   1. Send SMS / Video Call API: Used it in a education chat application where tutors connect to live sessions with their students for a one to one chat
   2. Autopilot: Used to create a chatbot