CSS Area of Exploration

1. CSS Position: Position property specifies the positioning method used in CSS. There are four methods of positioning. Static, Relative, Fixed, Absolute, Sticky. Elements are positioned using the top, bottom, left, and right properties after the position property are set.

   Static: By default, HTML elements are positioned static.

   Relative: Position relative indicates the relative position to its normal position, according to the top, right, bottom, left properties.

   Absolute: The element is removed from its original position and that position is filled up by its relative positioning element. It is positioned relative to its closest positioned ancestor. Its final position is determined by the top, right, bottom, left properties.

   Fixed: The element is removed from its normal flow document and positioned according to the top, left, bottom, right, properties.

2. Z-Index: Z-index sets the stack order of the element. This means which elements are being shown in front or behind the others, when two positioned overlapped each other without z-index specified the last HTML code shown top always. The value of z-index placed positive or negative value. Z-index only works in the positioned elements.

3. CSS Overflow: The overflow property indicates the clip or content add or remove the scroll bars according to the body content. The overflow property has four values. This is visible, hidden, scroll, auto.

   visible: The default behavior of the element in HTML.
   hidden: the overflow is clipped and the rest of the content is invisible.
   scroll: The overflow is clipped and a scrollbar is added to see the rest of the content.
   auto: Similar to scroll but it only adds a scrollbar when necessary.

4. CSS combinators: A combinator is a special thing that made the relationship among the selectors.
   The combinators are descendent selector (space), child selector (>), adjacent sibling selector (+), general sibling selector (~)

   descendent selector (space): The descendent selector matches all the elements to the descendant's element,
   child selector (>): The child selector selects all the child elements of the specified element.
   adjacent sibling selector (+): It indicates directly after to another specific element.
   general sibling selector (~): The general sibling selector selects all the elements that are the next sibling of the specified element.

5. CSS Transform: Transform property allows move, rotate, scale and skew to the CSS elements. The transform properties of CSS are translate (), rotate (), scaleX (), scaleY (), scale (), skewX (), skewY (), skew (), matrix ()

6. CSS Transition: Transition allows to change the property value smoothly in a given direction. The transition has some properties like transition, transition-delay, transition-duration, transition-timing-function. The transition effect will start when the specified CSS property changes the value.
   transition-timing-function: there are some values for the transition-timing function. These are ease (start slowly, then fast, end),
   linear (same speed from start to end), ease-in (slow-start), ease-out(slow-end), ease-in-out (slow start and end)

7. CSS Animations: An animation lets an element gradually change from one style to another. To use CSS animation, keyframes are specified for the animation. Keyframes hold the style to be indicated at certain times. Use @keyframe animation_name property to use the animation.

String Property

1. charAt (): The string objects charAt () method returns the new string according to the index of the specified string.
   ```
   Let string = “Hello World”
   console.log (string. charAt (0))
   console.log (string. charAt (1))
   console.log (string. charAt (2))
   console.log (string. charAt (3))
   console.log (string. charAt (4))

   Output: H e l l o
   ```

2. concat (): The concat method adds two or more strings or arguments together.
   ```
   Let str1 = “Hello”
   Let str2 = “world”
   Let str3 = str1.concat (str2)

   Output: Hello World
   ```

3. includes (): The includes method performs a case-sensitive search, whereas to find a string from another string returning true or false as output.
   ```
   Let string = “Hello World”
   string. includes(‘hello’)

   Output: false
   ```

4. endsWith (): The endsWith () method determines whether a string is ends with the character or not. It returns a Boolean output like true or false.

   Let string = “Hello World”
   string. endsWith(‘hello’)

   Output: false

5. indexOf (): returns the index within the calling string object of the first occurrence of the specified value. It returns -1 if the value is not found.

   Let string = “Hello World”
   string. indexOf (‘’, 0)

   Output: 0

6. lastIndexOf(): searching backward from the index. Returns -1 if the value is not found.

   Let string = “Hello World”
   string. lastIndexOf (‘a’)

   Output: -1

7. replace(): The replace method replaces the string within the given string.

   Let string = “Hello World”
   string. replace (” World”,” Bangladesh”)

   Output: Hello Bangladesh

8. slice: The slice method extracts a section of a string and returns a new string, without modifying the original string.

Let string = “Hello World”
string. slice (6)

Output: World

<u>**JavaScript Array**</u>

1. Filter: The filter method creates a new array according to some test implementation by the function.
   const words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];
   const result = words. filter (word => word. length > 6);

2. Find: The find method returns only the single array from an array or object or returns a single element from an array.
   const array1 = [5, 12, 8, 130, 44];
   const found = array1.find(element => element > 10);

3. forEach: The forEach method returns only a single array from the executed functions.
   const array1 = ['a', 'b', 'c'];
   array1.forEach(element => console.log(element));

4. shift: shift method removes the first element from an array and returns the removed element.
   const array1 = [1, 2, 3];
   const firstElement = array1.shift();
   console.log(array1);

5. unshift: The unshift method adds one or more element at the beginning of the array.
   const array1 = [1, 2, 3];

   console.log(array1.unshift(4, 5)); Output: [4,5,1,2,3]

JavaScript try-catch

JavaScript try catch uses from test block of code and detecting of error. Like the try statement gives you to test the block of code for errors and catch statement handles the error. Both try and catch statement are used together for running code detect error and also handle the error.

E.g.

```
try {
 Alert (“hello world”)
}
Catch {
Console.log (“alert is not working.”)
}
```

ES6

1. Functions with Default Parameter Values: Default function parameters allows named parameter to be initialized with default values or passing undefined or no values.

   E.g. function add (a, b = 1) {
   ```
     return a + b;
   }
   console.log (add (5, 2));
   Output: 7
   console.log (add (2));
   Output: 3
   ```

2. Destructuring: Destructuring makes it possible to unpack the values from arrays or objects. By using destructuring we can get values from array and objects property.

   E.g. let a, b, rest;
   ```
   [a, b] = [10, 20];

   console.log(a);
   Output: 10
   ```

3. Spread Operator: Spread operator just copies all the array element from the previous array and returns a new array.

   ```
   E.g. function sum (x, y, z) {
     return x + y + z;
   }
   const numbers = [1, 2, 3]'
   console.log (sum (...numbers));
   ```

1. Truthy and Falsy Value: Truthy and falsy are Boolean contexts. Truthy value is a type of value that is by default true in nature. And falsy value is the type of value that is by default false in nature. A value is truthy unless it's defined as falsy.

2. Undefined and Null: Undefined means in JavaScript that a variable is being declared with no value. In other words, without value, declaring a variable is so-called undefined.
   Null means zero. That is the absence of value and a variable. Neither variable nor value declared the JavaScript returns null.

3. "==" and "===": The "==" and "===" operator is the comparison operator. The main difference between "==" and "===" are "==" compares only two variables without checking the datatype of the variable. Whereas the "===" operator compares two variables in strict modes, such as comparing the variable with checking the datatypes.

4. Closure: A closure is a combination of functions. In JavaScript, an inner function always takes some reference of the outer function. In general, a closure makes access to an outer function from an inner function. When the function is created a closure is also being created in JavaScript. A parent function always sends the value or variable to its child function. But The parent can't access the child's element.

5. Encapsulation: The wrapping up of data and function into a single unit (called class) is known as encapsulation. The data is not accessible to the outside and those functions which are wrapped in the class can access it.

6. Event loop Stack: The Event loop has only one task that is its just monitors the call back queue and the callback stack. Generally, JavaScript programs are being executed by the google browser v8 engine or in node js. The JavaScript engine moves the code in a callback stack. Naturally, the stack is a LIFO system. All the functions and code go through to the stack. And then serially the code is being executed. But when an asynchronous function occurs like a set timeout. The JavaScript engine pops out the function and sends it to the web API for the browser or c++ API for node-js. After pop up, the stack works normally as before. Besides this, the web API also executed the function given from the callback stack. And after execution, the response is being transferred to the callback queue. And all the responses get and executed by the web API from the callback stack are being transferred to the callback queue. The callback queue stores the execution code and when the call stack is free it sends the code to the callback stack again. The monitoring between the callback queue and callback stack is done by the event loop.

7. This: This keyword refers to the object that it belongs to. In a method this refers to the owner object, in a function this refers to the global object.

8. Event Bubble and Event Delegate: Event bubbling is a method in which an event is propagated to one element inside another element and both elements is being registered and handled. The event is first captured to the inside element and then propagated to the outside element. We can terminate the event bubble by using the stopPropagation method.

9. DOM: DOM stands for Document Object Model. DOM represents the content or document visually to the web browser. It’s a programming interface for web documents. The Dom is not a part of the JavaScript language but is instead a Web API used to build the websites.

10. Window and Document: Window is a global object and contains all the variables, functions, classes and history, and location. The document is also a part of the window and as considered the part of the Window

1. PropTypes: By using props we can pass different types of information like integers, strings, arrays, etc. as props to the components. We can either create default props or pass props directly to the component. Passing props from outside of a component and receiving them inside of another component. But we never check the data types of the props of the components. It's totally upon us whether we validate props in the components or not. But It's a good practice for the large application to validate the props type of the components. This will help to debug easily and avoid bugs in the future. We can use the prototype for validating any data we are receiving from props. But before using it we must add the dependency in our project and import it. Just like default props, propTypes are also objects whereas keys are the prop names and values are their types.

2. State-Props: Props are used for passing the data from one component to another. The state is something that passed the data within the component only. Props are immutable and cannot be modified, where the state is mutable and can be updated or modified. Props can be used within the state and functional components. Both of them hold the information.

3. JSX: JSX stands for JavaScript XML. Writing the HTML code with JavaScript is known as JSX. JSX provides the syntactic sugar for the reaction. createElement (). JSX doesn't return multiple parent elements. After compilation, JSX expressions become regular JavaScript function calls and evaluate JavaScript objects. In JSX JavaScript code can be written

with curly braces. JSX always allows a single line Code for JavaScript. JSX allows declaring variables with Html.

4. Component Lifecycle: Three phases introduce the react component life cycle. The phases are Mounting, Updating, and Unmounting. Mounting means adding elements into the DOM. Four built-in methods of react have been called for Mounting. These are constructor (), getDerivedStateFromProps (), render (), componentDidMount (). The next phase in the lifecycle is updating. A react component is updated when its state or props are being updated. The last phase of react life cycle is Unmounting. Unmounting happens when a component is removed from the virtual DOM.

5. Hooks: Hooks specified the functional component to access the react components state and to allow the other features. Hooks are one kind of function that lets us hook into react state and lifecycle features. Because of hooks, there is no need to use the class components in react. Hooks generally replace the class components. Hooks are mainly used to “hook” for react features such as state and components lifecycle methods. There are a few kinds of hooks that are being introduced to the react these are useState, useEffect, useLocation, use history, useParam, etc.

6. Custom Hooks: The custom hook is created by the user. The custom hook is a JavaScript function that is created by the programmer or developer. To share logic among some components and other JavaScript functions custom hook is very useful. Custom hook allows sharing a piece of code to share different levels of several parts of the app

7. Context-API: In React sometimes the props are being sent to the 3rd,4th, and nth, components. But the intermediate component has no connection with the props. It’s called prop drilling. To solve this problem context API is introduced. Context API is a way to produce global variables effectively. This is the process of alternating prop drilling. These are also mentioned as props transferred from grandparents to children to parents and so on.

8. Virtual-DOM and diff Algorithm: DOM stands for Document Object Model. Dom is very much expensive. When to change one element into the DOM. It changes the whole structure of the model. So, DOM always supervises the changes. Virtual Dom makes a copy of real DOM and compares it. The Virtual Dom compares with the real Dom and identifies only the changes that might occur.

9. Prop Drilling: In React sometimes the props are being sent to the 3rd,4th, and nth, components. But the intermediate component has no connection with the props. It's called prop drilling. To solve this problem context API is introduced. Context API is a way to produce global variables effectively. This is the process of alternating prop drilling. These are also mentioned as props transferred from grandparents to children to parents and so on.

10. Optimize React-JS Application: There have a few techniques to optimize the React Js Application. Using immutable Data Structure revolves around a strict unidirectional data flow. Immutable data objects are simpler to create, test and use. Use React.Fragment to avoid additional Element wrapper: react. fragment allows a group of children without adding an extra node.