15 Javascript snippets

Pooja Jangid

Check if a String Contains a Substring

```
const str = "Hello, World!";
const contains = str.includes("World"); // true
```

- **Explanation:** includes() checks if a string contains a specified substring, returning true if it does and false otherwise.

Pooja Jangid

Convert String to Number

```
const str = "123";
const num = Number(str); // 123
```

- **Explanation:** Number() converts a string (or any other type) to a number. If the string isn’t a valid number, it returns NaN.

Pooja Jangid

Generate a Random Number Between Two Values

```
function getRandom(min, max)
 {
   return Math.random() * (max - min) + min;
}
```

- **Explanation:** This function generates a random floating-point number between min and max.

Pooja Jangid

Remove Duplicates from an Array

```
function getRandom(min, max)
  {
    return Math.random() * (max - min) + min;
}
```

- **Explanation:** Using Set, which only stores unique values, and the spread operator [...], you can remove duplicates from an array.

Pooja Jangid

Generate a Random Number Between Two Values

```
function getRandom(min, max)
  {
    return Math.random() * (max - min) + min;
}
```

- **Explanation:** This function generates a random floating-point number between min and max.

Pooja Jangid

Flatten a Nested Array

```
const nestedArr = [1, [2, [3, 4]], 5];
const flatArr = nestedArr.flat(Infinity); // [1, 2, 3, 4, 5]
```

- **Explanation:** flat() flattens an array to a specified depth. Using Infinity as the argument flattens it completely.

Pooja Jangid

Debounce Function

```
function debounce(func, delay) {
  let debounceTimer;
  return function(...args) {
  const context = this;
  clearTimeout(debounceTimer);
  debounceTimer = setTimeout(() => func.apply(context, args), delay);
};
}
```

- **Explanation:** Debouncing ensures a function is only called after a certain amount of time has passed since it was last called, useful in scenarios like preventing multiple submissions of a form.

Pooja Jangid

Check if an Object is Empty

```
const isEmpty = (obj) => Object.keys(obj).length === 0;
```

- **Explanation:**This function checks if an object has any properties. If Object.keys(obj).length is 0, the object is empty.

Copy to Clipboard

```
function copyToClipboard(text) {
  navigator.clipboard.writeText(text).then(() => {
    console.log("Copied to clipboard");
  });
}
```

- **Explanation:**This snippet copies a given text to the user’s clipboard using the clipboard API.

Deep Clone an Object

```
const obj = { a: 1, b: { c: 2 } };
const deepClone = JSON.parse(JSON.stringify(obj));
```

- **Explanation:**This method creates a deep clone of an object, meaning nested objects are also copied. However, this method won’t work with functions or special objects like Date.

Get the Current Date in YYYY-MM-DD Format

```
const today = new Date().toISOString().split('T')[0]; // "2024-08-21"
```

Explanation: This snippet returns the current date in the common YYYY-MM-DD format by using toISOString() and then splitting the string at T.

Find the Maximum Value in an Array

```
const arr = [10, 5, 100, 2];
const max = Math.max(...arr); // 100
```

Explanation: ThiUsing the spread operator ..., you can pass an array as individual arguments to Math.max() to find the maximum value.

Remove Falsy Values from an Array

```
const arr = [0, 1, false, 2, '', 3];
const filteredArr = arr.filter(Boolean); // [1, 2, 3]
```

Explanation: The filter() method can be used with Boolean to remove all falsy values (false, 0, "", null, undefined, NaN) from an array.

Scroll to Top of the Page

```
window.scrollTo({ top: 0, behavior: 'smooth' });
```

- **Explanation:**This snippet smoothly scrolls the page back to the top, which is commonly used in "back to top" buttons.

Pooja Jangid

Get the Query Parameters from a URL

```
const urlParams = new URLSearchParams(window.location.search);
const paramValue = urlParams.get('paramName');
```

Explanation: This code gets the value of a specific query parameter from the current URL.

Pooja Jangid

Throttle Function

```
function throttle(func, limit) {
  let inThrottle;
return function() {
   const context = this,
   args = arguments;
  if (!inThrottle)
  {
    func.apply(context, args);
    inThrottle = true;
     setTimeout(() => inThrottle = false, limit);
  }
 }
}
```

Explanation: Throttling ensures a function is only called at most once in a specified time period. This is useful for rate-limiting event handlers.

Pooja Jangid

Thank you

Pooja Jangid