TW-10 TEAM LEAD VERSION

TEAMWORK

CLARUSWAY
WAY TO REINVENT YOURSELF

Meeting Agenda

- Icebreaking
- Questions
- Interview Questions
- Coffee Break
- Coding Challenge
- Video of the week
- Retro meeting
- Case study / project

Teamwork Schedule

Ice-breaking 10m

- Personal Questions (Stay at home & Corona, Study Environment, Kids etc.)
- Any challenges (Classes, Coding, studying, etc.)
- Ask how they're studying, give personal advice.
- Remind that practice makes perfect.

Ask Questions 15m

1. How do you access a property of an object in JavaScript?

A. By using square brackets
B. By using the dot notation
C. By using parentheses
D. By using commas

Answer: B

2. How do you check if a property exists in an object in JavaScript?

A. By using the exist keyword
B. By using the contains keyword
C. By using the hasOwnProperty method
D. By using the isProperty method

Answer: C

```
const person = {
  name: "Bob"
};

console.log(person.hasOwnProperty("name")); // true - Using hasOwnProperty
console.log("age" in person); // false - Using the 'in' operator
```

3. How do you delete a property from an object in JavaScript

A. By using the `delete` keyword
B. By using the `remove` keyword
C. By setting the property value to null
D. By assigning an empty string to the property

Answer: A

```
const person = {
 name: "John",
 age: 30
};

delete person.age; // Removing the 'age' property
```

4. How do you add a new property to an existing object in JavaScript

A. By using the `add` keyword
B. By using the `insert` keyword
C. By using the `update` keyword
D. By assigning a value to a new key

Answer: D

5. What is an object in JavaScript?

A. A function
B. A data tool
C. A data structure
D. An array

Answer: C

6. How can you create an empty object in JavaScript?

A. `emptyObject = {};`
B. `emptyObject = new Empty();`
C. `emptyObject = Object.empty();`
D. `emptyObject = new Object();`

Answer: A

7. How do you clone an object in JavaScript?

A. Use the Object.clone() method
B. Use the Object.assign() method or the spread operator (...)
C. Use the Object.copy() method
D. Use the Object.duplicate() method

Answer: B

By Object.assign() :

```
const originalObject = { name: "John", age: 30 };

// Clone the original object using Object.assign()
const clonedObject = Object.assign({}, originalObject);

// Now, 'clonedObject' is a separate copy of 'originalObject'
console.log(clonedObject); // { name: 'John', age: 30 }
```

By spread operator :

```
const originalObject = { name: "John", age: 30 };

// Clone the original object using the spread operator
const clonedObject = { ...originalObject };

// Now, 'clonedObject' is a separate copy of 'originalObject'
console.log(clonedObject); // { name: 'John', age: 30 }
```

8. What is object destructuring in JavaScript?

A. A way to create objects from strings
B. A way to concatenate objects
C. A way to merge objects
D. A way to extract properties from an object and assign them to variables

Answer: D

9. How do you swap the values of two variables without using a temporary variable using array destructuring?

A. const a = b; const b = a
B. const [a, b] = [a, b];
C. const [a, b] = [b, a];
D. const [b, a] = [a, b];

Answer: C

10. What happens if you try to destructure an array with more variables than there are elements in the array?

A. Extra variables are assigned undefined
B. An error is thrown
C. The array is automatically resized
D. Only the first few variables are assigned values

Answer: A

11. What does the rest element (...) do in array destructuring?

A. It spreads elements into multiple arrays
B. It gathers remaining elements into an array
C. It removes elements from the array
D. It reverses the order of elements in the array

Answer: B

```
// Example 1: Collecting remaining elements
const numbers = [1, 2, 3, 4, 5];

// Using rest element to collect the remaining elements
const [first, second, ...rest] = numbers;

console.log(first); // 1
console.log(second); // 2
console.log(rest); // [3, 4, 5]
```

12. What is JSON (JavaScript Object Notation)?

A. A lightweight data interchange format
B. A JavaScript method for creating objects
C. A way to define variables in JavaScript
D. A JavaScript library for animations

Answer: A : JSON is a lightweight data interchange format that is often used to transmit data between a server and a web application. It is based on a subset of JavaScript object literal notation.

```
{
  "name": "John Doe",
  "age": 30,
  "city": "New York",
  "isStudent": false,
  "hobbies": ["reading", "hiking", "cooking"],
  "address": {
    "street": "123 Main St",
```

```
    "zipcode": "10001"
  }
}
```

13. Write a code for get *sum of every positive element* in given array

```
const input = [1, -4, 12, 0, -3, 30, 42, -150];

input.filter(function (num) {
    return num > 0;
  }).reduce(function (accumulator, currentValue) {
    return accumulator + currentValue;
  }, 0);

// or written with Arrow function
input
  .filter((num) => num > 0)
  .reduce((accumulator, currentValue) => accumulator + currentValue, 0);

//output: 85
```

14. Write a code for abbreviate the given name and return the name initials.

```
const input = "John Ronald Reuel Tolkien"

input.split(" ").map(function (word) {
    return word[0];
  }).join("");

// or written with Arrow function
input
  .split(" ")
  .map((word) => word[0])
  .join("");


//output: JRRT
```

15. If you want to square each element of an array and return a new array with the squared values, which method would you use?

A. `reduce`
B. `filter`
C. `map`
D. All of the above

Answer: C

```
const numbers = [1, 2, 3, 4, 5];
const squaredNumbers = numbers.map((num) => num ** 2);
```

16. If you want to find the sum of all even numbers in an array, which method would you use?

A. `map`
B. `reduce`
C. `filter`
D. `forEach`

Answer: B

```
const numbers = [1, 2, 3, 4, 5];
const sumOfEvens = numbers.reduce((acc, num) => (num % 2 === 0 ? acc + num : acc),
0);
```

17. Write a code get each array elements length to a new array with map() method.

```
const names = ["Alice", "Bob", "Charlie"];
const nameLengths = names.map((name) => name.length);

console.log(nameLengths) // [5, 3, 7]
```

18. Write a code get each array elements capitalized with map() method.

```
const words = ["apple", "banana", "cherry"];
const capitalizedWords = words.map((word) => word.toUpperCase());

console.log(capitalizedWords) //['APPLE', 'BANANA', 'CHERRY']
```

Interview Questions 15m

1. What is the difference between Object.keys(), Object.values(), and Object.entries()?

Answer : Object.keys() returns property names, Object.values() returns property values, and Object.entries() returns key-value pairs

2. What is the Object.freeze() method used for?

Answer :To make an object immutable, preventing changes to its properties

3. What is constructor functions in JavaScript?

Answer :In JavaScript, constructor functions are special functions used to create and initialize objects. They are used as templates or blueprints for creating multiple objects of the same type, each with its own unique property values but sharing common methods. Constructor functions are typically invoked with the `new` keyword.

Here's how you define and use a constructor function:

```
// Constructor function for creating Person objects
function Person(name, age) {
  this.name = name;
  this.age = age;
}

// Creating instances (objects) using the constructor function
const person1 = new Person("Alice", 25);
const person2 = new Person("Bob", 30);

console.log(person1); // { name: 'Alice', age: 25 }
console.log(person2); // { name: 'Bob', age: 30 }
```

In the example above:

- We define a constructor function `Person` that takes two parameters, `name` and `age`. Inside the constructor function, `this` refers to the newly created object.
- We create two instances (objects) of the `Person` constructor using the `new` keyword, passing specific values for the `name` and `age` properties.
- Each instance, `person1` and `person2`, has its own set of `name` and `age` properties, initialized with the values provided during object creation. Constructor functions are useful for creating multiple objects that share a common structure and behavior. They allow you to encapsulate the object's initialization logic and methods within a reusable function, promoting code organization and reducing redundancy. Additionally, constructor functions can be used in conjunction with prototypes to share methods across all instances,

which can help conserve memory and optimize performance when dealing with many objects of the same type.

4. Explain `reduce()` method in Javascript

Answer : .reduce() runs a callback for every array element just like .map() does. The only difference is that reduce() passes the result of this accumulator from one array element to the other. Some built-in reduce() functions are: Array.prototype.reduce(), and the reduceRight() method which are used to apply functions against accumulators from 1. Left to right and 2. Right to left respectively.

The accumulator either contains the initial value or the return value from the previous call. The accumulator could be any string, integer, object, etc. It is the net result of the function. The present value of the accumulator is simply the element that is being worked against.

Accumulators must be passed when .reduce() is being called.

5. What is the DOM?

Answer : The DOM is the Document Object Model, which is a tree-like structure that represents the HTML document. It is used by JavaScript to access and manipulate the document.

Coding Challenge **15m**

1. High Priced Product Categories

- You are given an array of objects representing a collection of products, each with a name, price, and category. Your task is to use map, filter, and reduce to calculate the average price of products in each category, and then return an array of objects containing only the categories that have an average price above 50.
- Sample input :

```
const products = [
  { name: "Product 1", price: 20, category: "Electronics" },
  { name: "Product 2", price: 30, category: "Clothes" },
  { name: "Product 3", price: 40, category: "Electronics" },
  { name: "Product 4", price: 50, category: "Clothes" },
  { name: "Product 5", price: 60, category: "Clothes" },
  { name: "Product 6", price: 70, category: "Electronics" },
  { name: "Product 7", price: 80, category: "Clothes" },
  { name: "Product 8", price: 90, category: "Electronics" },
];
```

- Expected outcome :

```
[
  { category: 'Clothes', average: 55 },
  { category: 'Electronics', average: 55 }
]
```

Solution :

```
const products = [
  { name: "Product 1", price: 20, category: "Electronics" },
  { name: "Product 2", price: 30, category: "Clothes" },
  { name: "Product 3", price: 40, category: "Electronics" },
  { name: "Product 4", price: 50, category: "Clothes" },
  { name: "Product 5", price: 60, category: "Clothes" },
  { name: "Product 6", price: 70, category: "Electronics" },
  { name: "Product 7", price: 80, category: "Clothes" },
  { name: "Product 8", price: 90, category: "Electronics" },
];

/* Use map to create an object with category as the key
and an array of products as the value */
const productsByCategory = products.reduce((acc, product) => {
  const category = product.category;
  if (!acc[category]) {
    acc[category] = [];
  }
  acc[category].push(product);
  return acc;
}, {});

// Use map to calculate the average price for each category
const avgPriceByCategory = Object.keys(productsByCategory).map(category => {
  const sum = productsByCategory[category].reduce((acc, product) => acc +
product.price, 0);
  return { category: category, average: sum / productsByCategory[category].length
};
});

// Use filter to only select categories with an average above a certain threshold
const highPricedCategories = avgPriceByCategory.filter(category => category.average
> 50);
  console.log(highPricedCategories)
```

2. HR VS IT Department

- **Task :** You are given an array of objects representing a collection of employees, each with a name, salary, and department. Your task is to use map, filter, and reduce to calculate the average salary for each department and then return an array of objects containing only the departments that have an average salary above 65000.
- Sample input :

```
const employees = [
  { name: "John", salary: 50000, department: "IT" },
  { name: "Jane", salary: 60000, department: "HR" },
  { name: "Bob", salary: 55000, department: "IT" },
  { name: "Sophie", salary: 75000, department: "HR" },
  { name: "Mike", salary: 65000, department: "IT" },
  { name: "Emily", salary: 80000, department: "HR" },
  { name: "David", salary: 70000, department: "IT" },
];
```

- Expected outcome :

```
[
  { department: 'HR', average: 71666 }
]
```

> Solution :

```
const employees = [
  { name: "John", salary: 50000, department: "IT" },
  { name: "Jane", salary: 60000, department: "HR" },
  { name: "Bob", salary: 55000, department: "IT" },
  { name: "Sophie", salary: 75000, department: "HR" },
  { name: "Mike", salary: 65000, department: "IT" },
  { name: "Emily", salary: 80000, department: "HR" },
  { name: "David", salary: 70000, department: "IT" },
];



/* Use reduce to create an object with department as the key
and an array of employee objects as the value */
const employeesByDepartment = employees.reduce((acc, employee) => {
  const department = employee.department;
  if (!acc[department]) {
    acc[department] = [];
  }
  acc[department].push(employee);
  return acc;
}, {});
```

```
// Use map to calculate the average salary for each department
const avgSalaryByDepartment = Object.keys(employeesByDepartment).map(department =>
{
  const sum = employeesByDepartment[department].reduce((acc, employee) => acc +
employee.salary, 0);
  return { department: department, average: sum /
employeesByDepartment[department].length };
});

// Use filter to only select departments with an average above a certain threshold
const highPaidDepartments = avgSalaryByDepartment.filter(department =>
department.average > 65000);
console.log(highPaidDepartments)
```

☕

Coffee Break 10m

☕

Video of the Week 10m

- JS DOM

Case study/Project 15m

- HC-05 iOS Calculator

Retro Meeting on a personal and team level 10m

Ask the questions below:

- What went well?
- What could be improved?
- What will we commit to do better in the next week?

Closing 5m

- Next week’s plan
- QA Session
