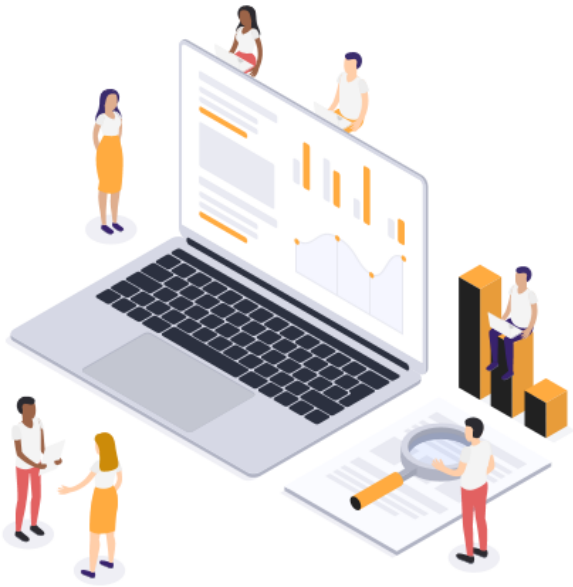
TW-11 TEAM LEAD VERSION

TEAMWORK



CLARUSWAY
WAY TO REINVENT YOURSELF

Meeting Agenda

- Icebreaking
- Workshop Activities
  - Questions
- Teamwork Activities
  - Interview Questions
  - Video of the week
  - Case study / project
- Retro meeting

Workshop Activities 90m

Ask Questions 30m

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 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 }
```

7. 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

8. 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

9. 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]
```

10. 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"
  }
}
```

11. The DOM presents an HTML document as a _____ .

A. Hash table structure
B. Dynamic structure
C. Tree-structure
D. All of these

Answer: C

12. You can find the element you want to manipulate with _____ ?

A. getElementById()
B. getElementsByTagName()
C. getElementsByClassName()
D. All of these way

Answer: D

13. The Document object is part of the Which object ?

A. Tree
B. Window
C. System
D. Anchor

Answer: B

14. Which method do you use to attach one DOM node to another?

A. JattachNode()
B. getNode()
C. querySelector()
D. appendChild()

Answer: D

15. Suppose that 'cw' is an element node. Select the expression below which can be used to select the parent node of 'cw'.

A. cw.getParent()
B. cw.parentContainer
C. cw.parentElement
D. cw.nodes()

Answer: C

16. How to remove the 'p1' class from the following element?

```
<p class="p1">A simple paragraph</p>

const pElement =  document.getElementsByClassName("p1")
```

A. pElement.classList.remove("p1")
B. pElement.className = ""
C. pElement.removeAttribute('class')
D. All of the above

Answer: D

17. You've written the event listener shown below for a form button, but each time you click the button, the page reloads. Which statement would stop this from happening?

```
button.addEventListener(
  'click',
  function (e) {
    button.className = 'clicked';
  },
  false,
);
```

A. e.blockReload();
B. button.preventDefault();
C. button.blockReload();
D. e.preventDefault();

Answer: D

18. What is the purpose of the `event.target` property in JavaScript event handling?

A. It returns the HTML element that triggered the event.
B. It returns the current time.
C. It returns the parent element of the target.
D. It returns the previous URL visited by the user.

Answer: A

19. What is wrong with this code?

```
const obj = {
  greet() {
    console.log('Hello, world!');
  },
  name: 'Ryan',
  age: 27,
};
```

A. The function greet needs to be defined as a key/value pair.
B. Trailing commas are not allowed in JavaScript.
C. Functions cannot be declared as properties of objects
D. Nothing, there are no errors.

Answer:D

20. What is the primary purpose of the DOM in web development?

A. To define the structure of an HTML document
B. To style web pages with CSS
C. To interact with and manipulate HTML elements in a web page
D. To create server-side scripts

Answer: C

21. How many event listeners can you add to a single element?

A. Two
B. As many as you want
C. Only one
D. eventlisteners count < Dom elements count

Answer: B

22. Choose the right Javascript event

A. `onmouseout`
B. `anmouseout`
C. `inmouseout`
D. `enmouseout`

Answer: A

23. Which method is used to create a new HTML element in the DOM using JavaScript?

A. `newElement`
B. `createNode`
C. `createElement`
D. `addNode`

Answer: C

24. What does the innerHTML property do in JavaScript?

A. It retrieves the CSS styles of an element.
B. It gets or sets the HTML content of an element.
C. It returns the tag name of an element.
D. It checks if an element has a specific class.

Answer: B

25. What is event propagation in JavaScript?

A. A method for registering multiple events on the same element
B. The act of stopping an event from occurring
C. A way to measure the time between events
D. The process by which events bubble up from the target element to the document

Answer: D

Teamwork Schedule

Ice-breaking 10m

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

Interview Questions 30m

1. Explain the difference between the `querySelector` and `querySelectorAll` methods for selecting elements in the DOM.

Answer :
`querySelector` returns the first element that matches the specified CSS selector, while `querySelectorAll` returns a collection of all elements that match the selector.

2. What is the purpose of the `getAttribute` and `setAttribute` methods in DOM manipulation?

Answer :
`getAttribute` is used to retrieve the value of an attribute on an element, while `setAttribute` is used to set or modify the value of an attribute on an element.

3. What is event bubbling, and how does it affect the order in which event handlers are executed?

Answer :
Event bubbling is a key concept in JavaScript's event propagation model. It describes the order in which events are handled when an event occurs on an HTML element and how the event "bubbles" up through the DOM hierarchy. Event bubbling affects the order in which event handlers are executed.

When an event (such as a click or mouseover) occurs on a DOM element, it doesn't just trigger an event handler on that specific element; it also triggers event handlers on all of the element's ancestor (parent) elements in the DOM tree, propagating from the innermost element to the outermost element.

Example :

```
<!DOCTYPE html>
<html>
<head>
  <title>Event Bubbling Example</title>
</head>
<body>
  <div id="container">
    <button id="button1">Button 1</button>
    <button id="button2">Button 2</button>
  </div>
</body>
</html>
```

```
// Get references to the container and buttons
const container = document.getElementById('container');
const button1 = document.getElementById('button1');
const button2 = document.getElementById('button2');
```

```
// Event handler for the container (ancestor)
container.addEventListener('click', function (event) {
  console.log('Container Clicked');
  console.log('Event Target:', event.target);
  console.log('Current Target:', event.currentTarget);
});

// Event handler for Button 1 (child)
button1.addEventListener('click', function (event) {
  console.log('Button 1 Clicked');
  console.log('Event Target:', event.target);
  console.log('Current Target:', event.currentTarget);
});

// Event handler for Button 2 (child)
button2.addEventListener('click', function (event) {
  console.log('Button 2 Clicked');
  console.log('Event Target:', event.target);
  console.log('Current Target:', event.currentTarget);
});
```

In this example, we have a simple HTML structure with a container div and two buttons inside it. We've attached event listeners to both the container and the buttons. The container's event listener will capture click events on any of its descendants, including the buttons.

When you click on Button 1 or Button 2, here's what happens:

The click event starts at the target element (Button 1 or Button 2) where it was initially triggered. Then, it "bubbles up" through the DOM hierarchy. It triggers the event handler on the container because the container is an ancestor of the buttons. As a result, you'll see the following output in the console when clicking Button 1, for example:

```
Button 1 Clicked
Event Target: <button id="button1">Button 1</button>
Current Target: <div id="container">...</div>
Container Clicked
Event Target: <button id="button1">Button 1</button>
Current Target: <div id="container">...</div>
```

4. What is the purpose of the `event.preventDefault()` method, and when would you use it?

Answer :
`event.preventDefault()` is used to prevent the default behavior of an event. It's often used when you want to stop a form submission or prevent a link from navigating to a new page

---

☕

Coffee Break 10m

☕

Video of the Week 10m

- DOM Events

Coding Challenge 40m

- 01 - Capital Letters

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 10m

- Next week's plan
- QA Session

Code Challenge Solution

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Text Capitalization</title>
  </head>
  <body>
    <textarea id="inputText" rows="10" cols="100">

    </textarea>
    <br />
    <button id="convertButton">Convert</button>
    <p id="outputText"></p>

    <script>
    document.getElementById("convertButton").addEventListener("click", function(){
          const inputText = document.getElementById("inputText").value;
          const outputText = convertText(inputText);
          document.getElementById("outputText").innerText = outputText;
        });

      function convertText(text) {
        const smallWords =
["a","an","the","and","but","or","nor","for","so","yet","to","of","in","on","at","b
y","with","up","as","is"];
        return text.split(" ").map((word, index) => {
            if (
              smallWords.includes(word.toLowerCase()) &&
              index !== 0 &&
              index !== text.split(" ").length - 1
            ) {
              return word.toLowerCase();
            }
            return capitalize(word);
          })
          .join(" ");
      }

      function capitalize(word) {
        return word.charAt(0).toUpperCase() + word.slice(1).toLowerCase();
      }
    </script>
  </body>
</html>
```