Q1 – Closure ??

- (Closure) هي عباره عن Function بتحتفظ بالوصول لل variables في النطاق بتعها و حتى بعد انتهاء النطاق بتعها.

- لفهم ال (Closure) بشكل أفضل، عندنا المثال ده.

```
function outerFunction() {
    let outerVariable = 10;

    function innerFunction() {
        console.log(outerVariable);
    }

    return innerFunction;
}

const closureFunction = outerFunction(); // closureFunction now holds the innerFunction

closureFunction(); // Outputs: 10
```

- في المثال ده ، عندنا outerFunction فيها var اسمه outerVariable و Function اسمها innerFunction. تحتوي innerFunction على وصول إلى outerVariable، حتى بعد انتهاء تنفيذ outerFunction وإرجاعها. لما بستدعي ال closureFunction، بتحتفظ ال function بالوصول لل outerVariable في نطاقها وبتطبع القيمه عادي .

- ال (Closure) تعد آلية قوية بحيث انها بتسمح بإنشاء fun بتحتوي على بيانات خاصة وتحقيق التجزئة. مع العلم انها بتستخدم بشكل واسع في سيناريوهات مثل التعامل مع events، والاستدعاءات الراجعة، وإدارة الحالة في البرمجة الوظيفية.
- ومع ذلك، المفروض استخدام ال (Closure) بعناية لتجنب تسرب الذاكرة غير المقصود، بحث انها ممكن تسيب ال var في الذاكره واحنا مش عايزينه

Q2 - bind vs call vs apply ??

- bind و call و apply هما ثلاث طرق في لغة الجافاسكريبت بنستخدمهم للتحكم في سياق استدعاء الدالة (قيمة this). وبيستخدمه بشكل شائع لتحديد السياق بشكل صريح عند استدعاء الدوال. نبزه مختصره عن كل واحده فيهم :

1 – bind

- بتنشئ طريقة bind دالة جديدة بنفس هيكل الدالة الأصلية و بتسمحلك بتحديد قيمة this بشكل دائم للدالة دي.
- بترجع دالة جديدة فيها قيمة محددة لـ this عند استدعائها، بالإضافة ل أي معاملات إضافية بمررها ليها.

--- الصيغة

```
const boundFunction = originalFunction.bind(thisArg[, arg1[, arg2[, ...]]]);

```

--- مثال

```
const person = {
    name: "Ali",
    fun: function() {
        console.log(`Hello, my name is ${this.name}.`);
    }
};

const anotherPerson = {
    name: "Ahmed"
};

const boundGreet = person.fun.bind(anotherPerson);
boundGreet(); // Outputs: Hello, my name is Ahmed.
```

call – 2

- بنستخدم طريقة call لاستدعاء الدالة فورًا وتحديد قيمة this بشكل صريح. وممكن برضو نمرر معلومات اضافيه علي شكل قائمة منفصلة بفاصلة.

--- الصيغة

```
originalFunction.call(thisArg[, arg1, arg2, ...]);
```

--- مثال

```
const person = {
    name: "Ali",
    fun: function() {
    console.log(`Hello, my name is ${this.name}.`);
    }
};

const anotherPerson = {
    name: "Ahmed"
};

person.fun.call(anotherPerson); // Outputs: Hello, my name is Ahmed.
```

apply – 3

- بتشبه طريقة apply طريقة call، بس بتقبل سياق this كمعامل أول، والمعامل الثاني هو مصفوفة أو كائن يشبه المصفوفة بيحتوي على المعاملات الي بنمررها للدالة.

--- الصيغة

```
originalFunction.apply(thisArg, [argsArray]);
```

--- مثال.

```
const person = {
    name: "Ali",
    fun: function(city, country) {
        console.log(`Hello, my name is ${this.name}. I'm from ${city}, ${country}.`);
    }
    };

const anotherPerson = {
    name: "Ahmed"
};

person.fun.apply(anotherPerson, ["Egypt", "Cairo"]);
// Outputs: Hello, my name is Ahmed. I'm from Egypt, Cairo.
```