**CLOSURE**

Outside function variable can be accessed inside the function.

What is the use of this closure is to maintain private variable

**CALL BACK**

Passing function as argument into another function.

Example

**FIRST ORDER FUNCTION AND HIGHER ORDER FUNCTION**

1. Ability to treat functions as values
2. Ability to treat function as a variable

Example:- <https://www.geeksforgeeks.org/what-is-first-class-citizen-in-javascript/>

**HIGHER ORDER FUNCTION**

Functon receive another function as arguments or return function as result.

1. Ability to pass a function as arguments

2. Ability to return a function from another function

Eg:- https://www.scaler.com/topics/nodejs/javascript-higher-order-functions/

**ARROW FUNCTION**

Allow us to write shorter function

let myFunction = (a, b) => a \* b;

**WHAT IS TYPE SCRIPT**

TypeScript is a strongly typed, object-oriented, compiled programming language that builds on JavaScript.

**CSS PREPROCESSOR:-**

1. It is a scripting language.
2. DRY – Don’t repeat yourself.
3. It allows programming concept such as variables, operations, functions, inheritance, and rules of selector nesting.
4. Each CSS preprocessor has its own way of writing code syntax, which is then compiled into regular CSS.
5. Some of the preprocessors are
6. SASS – Syntactically awesome stylesheet
7. LESS – Leaner Awesome stylesheet
8. STYLUS
9. CSS-CRUSH
10. MYTH

**Pros of using CSS preprocessors**

1. It makes your code more maintainable. For example, you can declare your brand colors in one place: $primaryColor, $secondaryColor, etc. If your brand colors change later, you only have to update them in one place now.
2. Write DRY CSS, a.k.a. Don’t Repeat Yourself. CSS preprocessors make it easy to reuse styles, meaning you don’t have to write the same code over and over.
3. They make your code more organized. Rather than sprawling sheets of styles, you can group your code and be more specific. Less repetition is shorter and more readable.
4. It’s more efficient. That repetition takes time! Especially updating it later when the design changes!

**Cons of using CSS preprocessors**

1. Debugging is harder. Since you’re reusing code, it could take longer to find where the problem is.
2. Additional complication time. Since the browser doesn’t read this more advanced version of CSS, it needs to compile it into regular CSS before showing the style.
3. Can produce very large CSS files. The source files will be more concise, but the generated CSS files could be huge. This could cause additional time for a request to complete.

SOLID Principle:-

1. S -- Single Responsibility Principle

Objects or entities should be open for extension, but closed for modification

1. O — Open closed principle

A class should have one and only one reason to change, meaning that a class should only have one job

1. L — Liskov substitution principle

Let q(x) be a property provable about objects of x of type T. Then q(y) should be provable for objects y of type S where S is a subtype of T.

1. I – Interface Segregation Principle

Many Client specific interfaces are better than one general purpose interface

1. D — Dependency Inversion principle

A client should never be forced to implement an interface that it doesn’t use or clients shouldn’t be forced to depend on methods they do not use.

* Depend upon abstraction not concretions

Ref: <https://medium.com/@cramirez92/s-o-l-i-d-the-first-5-priciples-of-object-oriented-design-with-javascript-790f6ac9b9fa>

<https://www.freecodecamp.org/news/solid-principles-explained-in-plain-english/>

<https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design>

**EVENT DELEGATION IN JAVASCRIPT:-**

<ul id="itemList">

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

const il = document.getElementById('itemList');

il.addEventListener("click", (event) => {

if(event.target.tagName === 'LI') {

console.log(event.target.textContent)

} })

In this example, a click event listener is attached to the <ul> element. The listener checks if the clicked element (event.target) is an <li>. If it is, the text content of that <li> is logged to the console.

Ref : https://www.linkedin.com/pulse/understanding-event-delegation-javascript-adekola-olawale-4bzbf