**CLOSURE**

Outside function variable can be accessed inside the function.

**CALL BACK**

Passing function as argument into another function.

Example

**FIRST ORDER FUNCTION AND HIGHER ORDER FUNCTION**

**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.

**Example Code:-**

