Caltech | Center for Technology & Management Education

Full Stack Java Developer

simplilearn

JavaScript

TECHNOLOGY

Functions

Learning Objectives

By the end of this lesson, you will be able to:

- Declare a function
- Describe functional expression and how it is used
- Categorize hoisting into a variable and function hoisting
- Learn what arguments are and how they are used
- Discuss getter and setter methods

Learning Objectives

By the end of this lesson, you will be able to:

- Understand the try-catch statement using syntaxes
- Learn the local and global scope
- Differentiate between let and var keywords
- Discuss the “This” keyword and how it is used

A Day in the Life of a Full Stack Developer

You are working on an application development project, where you see the application is working fine. However, when sent for alpha testing, the application exits abruptly in a few test cases.

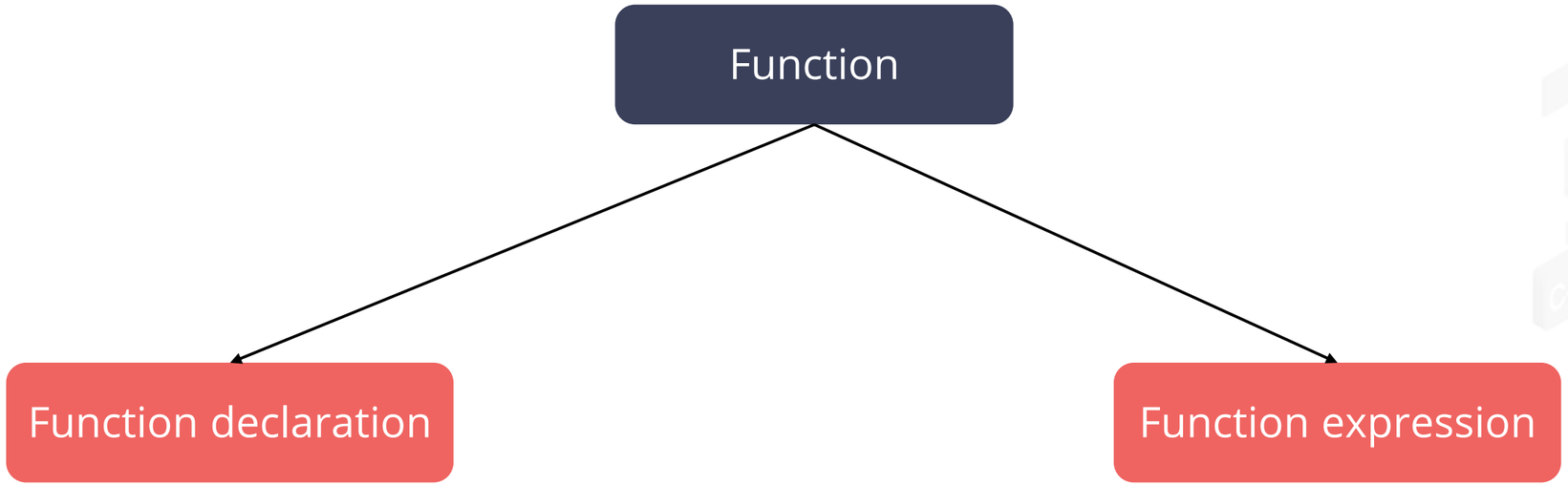
You have been assigned the task to fix this. You decide to use the try-catch block in the code, where the exceptions can be caught, and the application responds to such cases.

To do so, you need to understand functional expressions, arguments, and try-catch statements.

Functions

Functions

A function in JavaScript is the procedure of a set of statements that performs a task or calculates a value.



Functions

Function declaration starts with the function keyword.

```
function calAge (birthYear)
{
        return 2021 - birthYear;
}
```

Statements are written with curly braces.

Functions

A function expression does not have a function name, and it is referred to as an anonymous function.

```
const myAge = function(birthYear)
{
            return 2021 - birthYear
}
```

Functions

A function declaration can be defined before calling it, whereas a function expression throws an error.

Hoisting

Hoisting

Hoisting is a JavaScript behavior that enables to move of all the declarations before a function or variable.

```
console.log(pic); // undefined
var pic;
```

The variable pic is only declared and has no value, an undefined value is assigned to it.

Variable Hoisting

Keyword var is hoisted, and the let and const do not allow the hoisting.

```
// program to display value
x = 10;
console.log(x);
var x; //  10
```

Variable x is used before declaring it, and the program works perfectly and displays the output 10.

Function Hoisting

In this hoisting, the function can be called before declaring it.

```
// program to print txt
greeting
 function greeting () {
   console.log( 'Hello , I am reet');
}
```

Arguments

Arguments

Arguments are array-like objects accessible inside the functions that contain the value of the arguments passed to that function.

```
function fun1(x, y, z) {
  console.log(arguments[0]);
  // expected output: 1
  console.log(arguments[1]);
  // expected output: 2
  console.log(arguments[2]);
  // expected output: 3
}
fun1(1, 2, 3);
```

The arguments object is a global variable that can be accessed from anywhere.

Arguments

Rest parameters allow a function to accept the indefinite number of arguments as an array.

A function with any number of arguments can be called using the rest parameters.

```
function functionname(...parameters)    //... is
the rest parameter (triple dots)
{
statement;
}
```

The default parameters are a new feature introduced in the ES6 version of JavaScript.

Arguments

Syntax and example:

```
function total   ( a =6, b = 9   {
          // return total
       return a + b;
}
console.log(total( 4, 7)); // 11
console.log(total(9)); // 18
console.log(total());
```

Getters and Setters

Getters and Setters

There are two types of object properties in JavaScript:

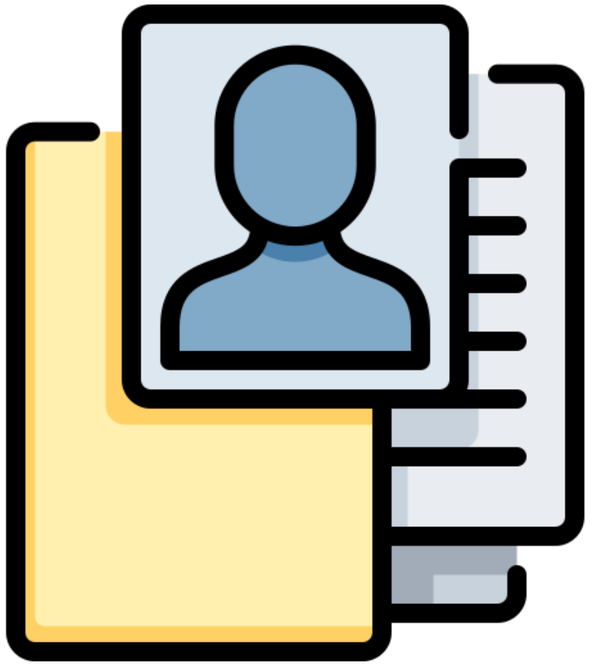
Data properties

Accessor properties

Data Properties

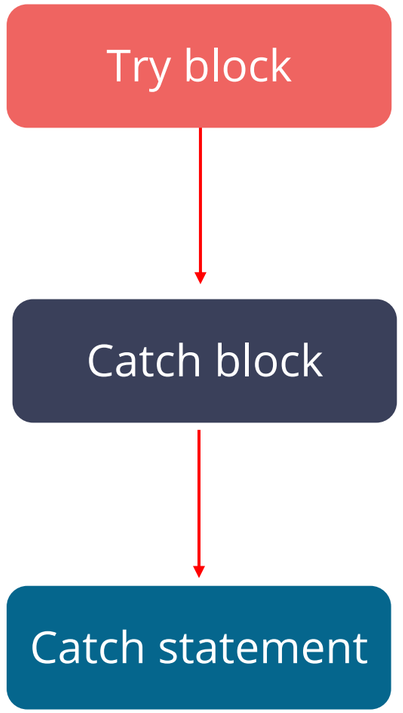
The data property sets or returns the value of the data attribute of an object.

```
const student
{
    // data property
   first Name:  'Ahana;
}
```



Accessor Properties

In JavaScript, the accessor properties are the methods that get or set the value of an object.



Accessor Properties

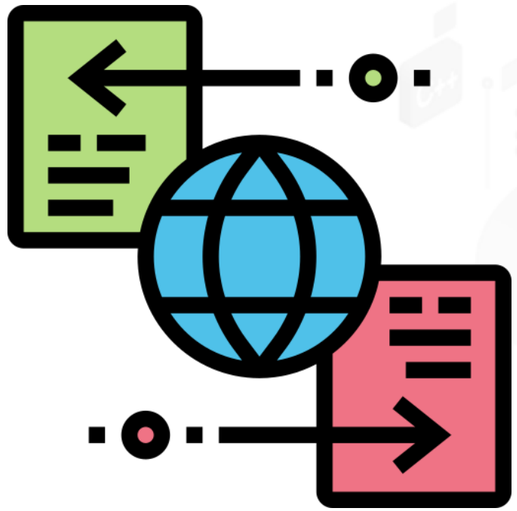
To access or get the properties of an object, the getter method is used.

```
const student
// data property
firstName:'Ahana';
// accessor property (getter)
getName(){
return this.firstName;
}
// accessing data property
console.log(student.firstName);
// accessing getter methods
console.log(student.getName);
// trying to access as a method
console.log(student.getName())
```

Accessor Properties

Setter methods are used to change or set the values of an object.

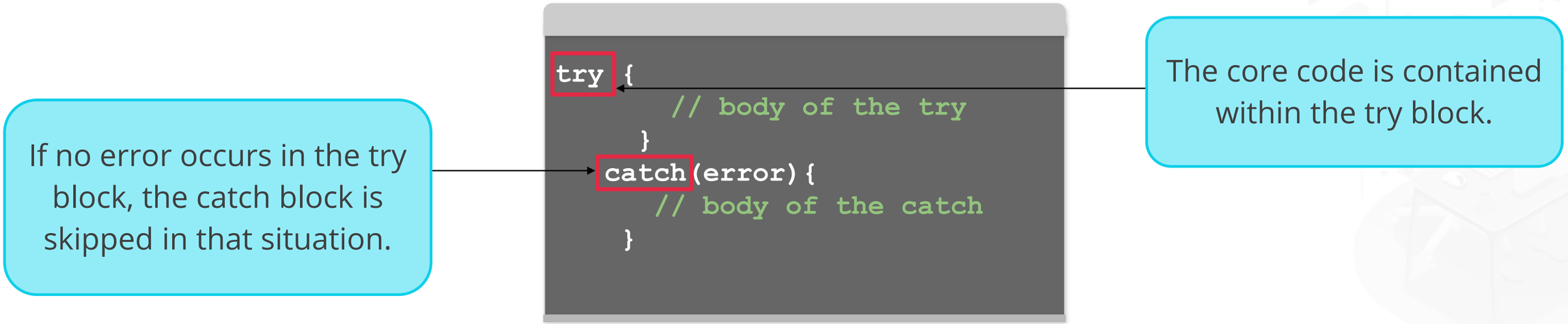
```
const student = {
      firstName: 'Ahana';
     // accessor property (setter)
      set changeName(newName); // Ahana
// change (set) object property using a setter
      student.changeName = 'Rose';
   console.log(student.firstName); // Rose
```



Try and Catch

Try and Catch

The try...catch statement is used to handle the exceptions.



Try and Catch

Example:

```
// program to show try...catch
const numerator= 200, denominator = 'd';
try {
     console.log(numerator/denominator);
    // forgot to define variable d
    console.log(d);
}
catch(error) {
    console.log('An error caught');
    console.log('Error message: ' + error);
}
```

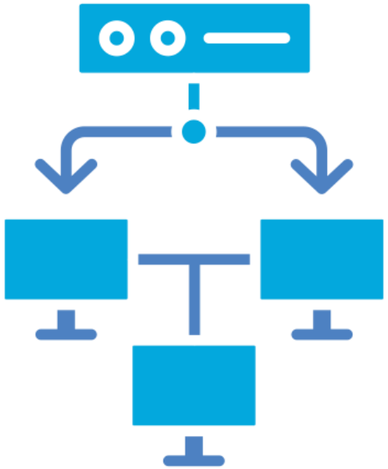
The d variable is not defined.

Local and Global Scope

Local and Global Scope

A variable in JavaScript has two sorts of scope:

Global scope



Local scope

Global Scope

Global scope refers to a variable that is declared outside the function at the start of the program.

Example:

The variable x is declared at the top of the code.

```
// program to print a text
let x = "hiii";
function greeting () {
    console.log(x);
}
greeting(); // hiii
```

Local Scope

A function can also contain a local variable that can be accessed only in the function.

Example:

The variable x is a global variable.

```
 x = "hiii";
function greeting() {
     let y = "Walt"
    console.log(x + y);
}
greeting();
console.log(x + y); // error
```

The variable y is the local variable.

Let vs. Var

Let vs. Var

The let keyword is limited to the block.

It is used to declare a variable and cannot be hoisted or accessed globally.

Example:

```
function varGreeting() {
          let x = 10;
             let x = 20; // syntax error
           // identifier x is already declared
            console.log(x);
        }
     varGreeting();
```

Let vs. Var

The var keyword declares a variable that can be redeclared and updated in the same scope.

Var keyword – Example:

```
function varGreeting() {
            let x = 10;
            let x = 20; // x is replaced
          // identifier x is already declared
            console.log(x);
       }
    varGreeting();
```

The var keyword can be declared and accessed globally.

This Keyword

This Keyword

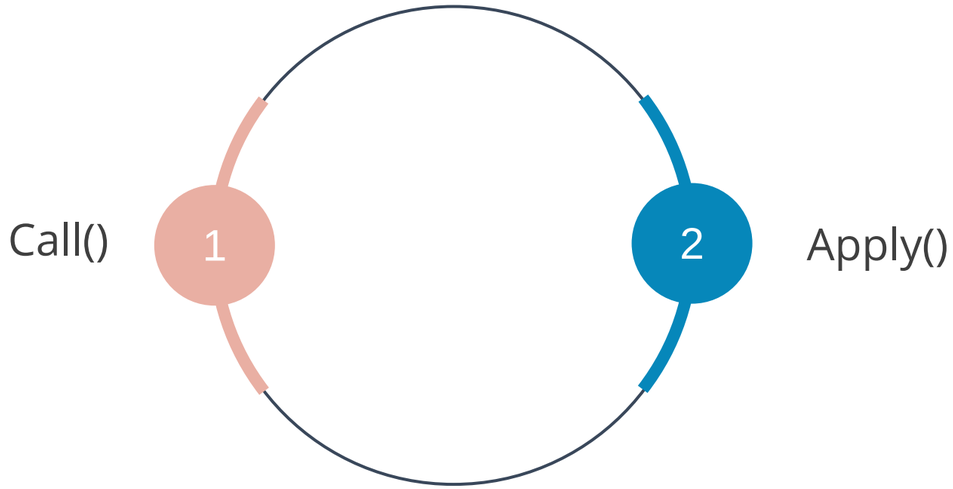
The ‘this’ keyword in JavaScript refers to the current object and the owner object in the method.

```
fullName : function()
{
  return this.email + " " + this.phone;
}
```

This keyword refers to the current object.

This Keyword

The ‘this’ keyword refers to the global object and is undefined in the strict mode function.



Reference of “this” from inside the function cannot be changed.

Key Takeaways

- A function in JavaScript is the procedure of a set of statements that performs a task or calculates a value.
- Hoisting is a JavaScript behavior that enables to move of all the declarations before a function or variable.
- Arguments are array-like objects accessible inside the functions that contain the value of the arguments passed to that function.
- The try...catch statement is used to handle the exceptions.
- The let keyword is limited to the block. It is used to declare a variable and cannot be hoisted or accessed globally.

TECHNOLOGY

Thank You

simplilearn