Coding Bootcamp

JavaScript

TECHNOLOGY

Functions

Learning Objectives

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

- Analyze the sequence of variable and function declaration executions in JavaScript code due to hoisting behavior
- Utilize hoisting in JavaScript to predict variable and function declaration behavior within scripts or functions
- Apply arguments within JavaScript functions to manipulate passed argument values
- Employ the try and catch statement to handle JavaScript code exceptions and errors

TECHNOLOGY

Functions

simplilearn

Functions

It is a block of reusable code that performs a specific task.

The two main ways of defining functions in JavaScript are:

- Function declaration
- Function expression

Functions

A 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; it is called an anonymous function.

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

Create a Function to Sort the Product List Based on Price

Problem Statement:

You have been asked to demonstrate how to sort a list of products based on their prices using a custom function in JavaScript.

Outcome:

By completing this demo, you will be able to effectively sort a list of products based on their prices using a custom function in JavaScript.

Note: Refer to the demo document for detailed steps:
01_Creating_a_Function_to_Sort_the_Product_List_Based_on_Price

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Declare and populate the products array
2. Define a function to sort the products array

Create a Function to Filter the Product List

Problem Statement:

You have been asked to create a function to filter an array of product objects based on specific criteria.

Outcome:

By completing this task, you'll be able to create a function in JavaScript to filter an array of product objects based on specific criteria.

Note: Refer to the demo document for detailed steps:
02_Creating_a_Function_to_Filter_the_Product_List

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Define the filter function

Hoisting

Hoisting

It is a JavaScript behavior that enables all the declarations to move 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');
}
```

Working on Hoisting

Problem Statement:

You have been asked to understand hoisting in JavaScript and demonstrate how it works using an example.

Outcome:

By completing this task, you will understand hoisting in JavaScript and be able to demonstrate how it works using an example.

Note: Refer to the demo document for detailed steps:
03_Working_on_Hoisting

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Define a function called greet

Arguments

Arguments

They 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 argument's object is a global variable that can be accessed from anywhere.

Arguments

Rest parameters allow a function to accept an 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

They are special functions in JavaScript that allow you to define how a particular property of an object is accessed and modified.

There are two types of object properties in JavaScript:

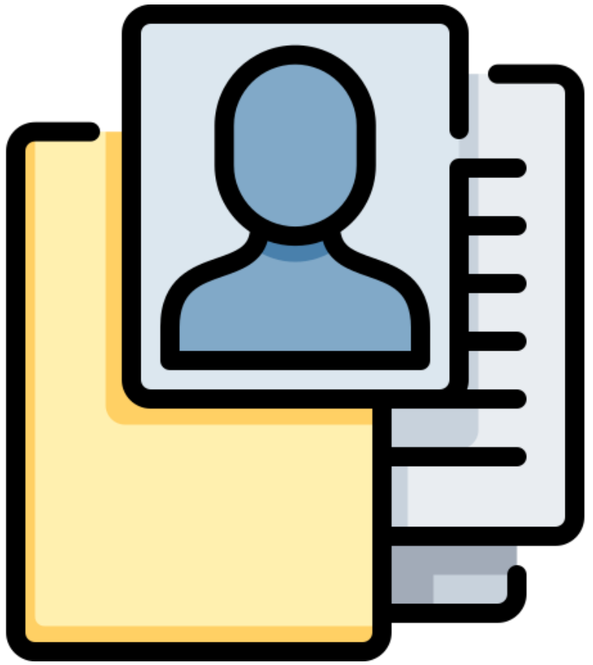
Data properties

Accessor properties

Data Properties

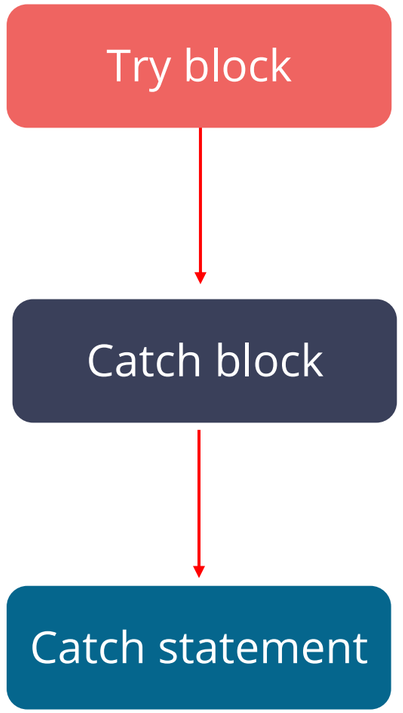
The data property sets or returns the value of the data attribute of an object.

```
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

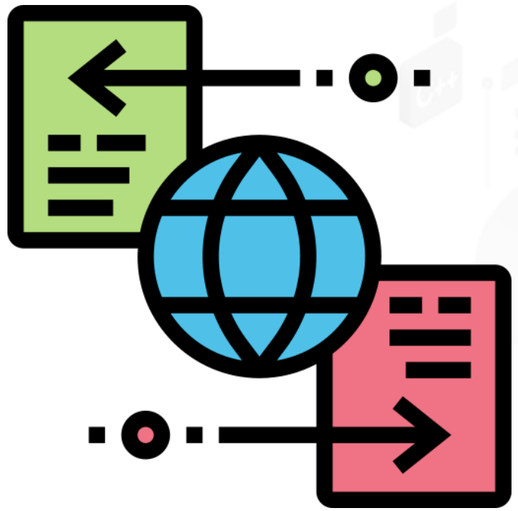
The getter method accesses or gets an object's properties.

```
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 change or set the values of an object.

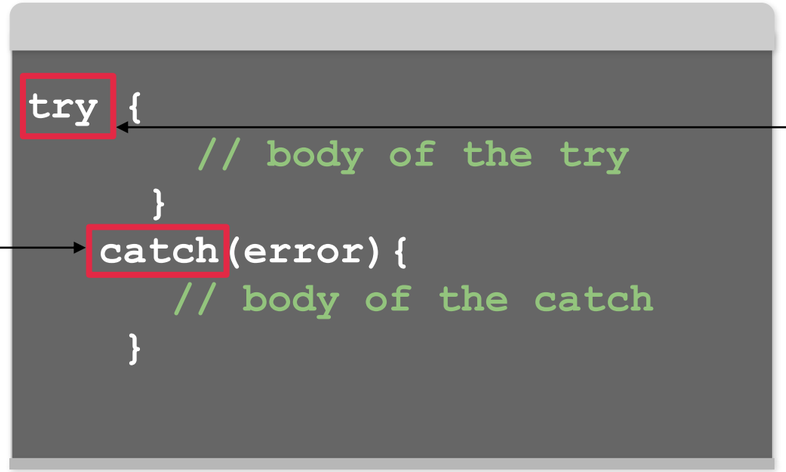
```
const student = {
      firstName: 'Ahana';
     // accessor property (setter)
      set changeName(newName); // Ahana
// change (set) object property using a setter
      student.changeName = 'Rose';
   console.log(student.firstName); // Rose
```



Try and Catch

Try and Catch

The try and catch statement handles the exceptions.



If no error occurs in the try block, the catch block is skipped in that situation.

The core code is contained within the try block.

Try and Catch

Example:

```
// program to show try and 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.

Working with Try and Catch

Problem Statement:

You have been asked to demonstrate the implementation of error handling using the try and catch block in JavaScript.

Outcome:

By completing this task, you'll be able to demonstrate the implementation of error handling in JavaScript using the try and catch block.

Note: Refer to the demo document for detailed steps:
04_Working_with_Try_Catch

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

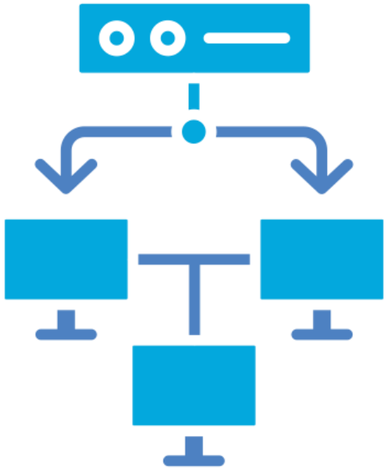
1. Implement error handling using the try and catch block

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 declared outside the function at the program's start.

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 declares 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

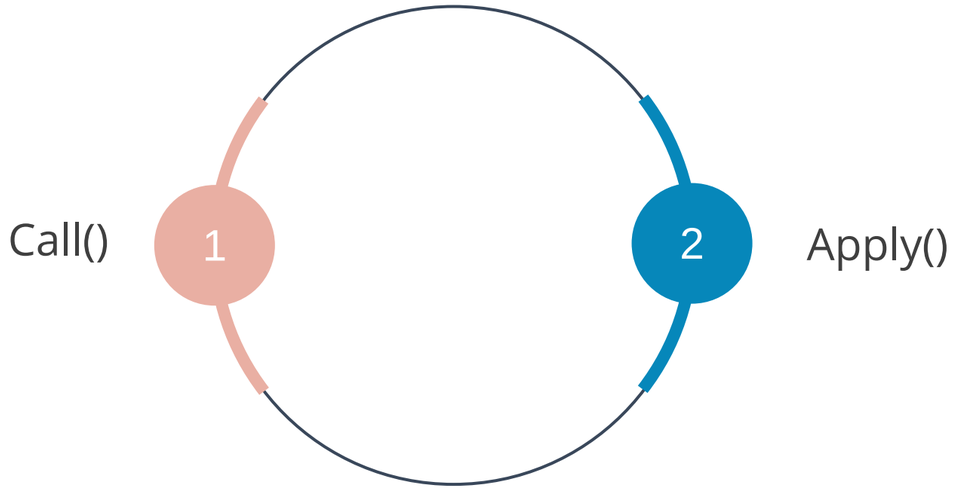
The this keyword in JavaScript refers to the current object and the owner object in the method.

```
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.



References to **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 all the declarations to move 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 and catch statement handles the exceptions.
- The let keyword is limited to the block. It declares a variable and cannot be hoisted or accessed globally.

TECHNOLOGY

Thank You

simplilearn