Javascript Functions

WRITING MODULAR CODE WITH FUNCTIONS

PRATEEK PAREKH
SENIOR STAFF ENGINEER

@ prparekh83

Introduction

- Function
- Arguments
- Function and Block Scope
- Immediately Invoked Function Expression (IIFE)
- Closures

Functions

Function

A function is a block of organized, reusable code that is used to perform a single, related action.

Functions

```
function greetings(name) {
    console.log('Hello ' + name);
}
greetings('John'); // Hello John
```

Functions

```
function greetings() {
     return ‘Hello John!’;
}
let message = greetings();
console.log(message); // Hello John!
```

Argument

An argument is a value that we pass to the function when we invoke it

Parameter

A parameter is a variable that we list as part of a function defintion

Function with Arguments

```
function sum(num1, num2) {
    return num1 + num2;
}
let result = sum(2, 3);
console.log(result); // 5
```

Function with Arguments

```
function greetings(name) {
    console.log('Hello' + ' ' + name);
}
let name = 'John';
greetings(name); // Hello John
name = 'Mary';
greetings(name); // Hello Mary
```

The Arguments Object

```
function printAll() {
    for(let i = 0; i < arguments.length; i++) {
        console.log(arguments[i]);
    }
}
printAll(1, 2, 3, 4, 5);   // 1 2 3 4 5
printAll(10, 20); // 10 20
```

Demo

Defining a simple Function

Defining a Function with a return statement

Defining a Function with arguments

Using the Arguments object

Function Scope

Function Scope

```
function greeting() {
    let message = 'Hello';
}
greeting();
console.log(message);       // ReferenceError: message is not defined
```

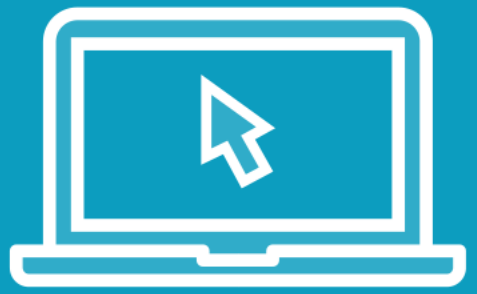
Function Scope

```
function greeting() {
    let message = 'Hello';
    let sayHi = function hi() {
          console.log(message);
    };
    sayHi();    // Hello
}
greeting();
```

Function Scope

```
function greeting() {
    let message = 'Hello';
    let sayHi = function hi() {
        let message = 'Hi';
    };
    sayHi();
    console.log(message);        // Hello
}
greeting();
```

Demo



Function Scope

Nested Function Scope

Block Scope

Variables declared with the var keyword or within function declarations DO NOT have block scope

Block Scope

```
let message = "Hello";
if (message === "Hello") {
    let count = 100;
}
console.log(count);       // Error
```

Block Scope

```
let message = 'Hello';
if (message === 'Hello') {
    let message = 'Inside if block';
    console.log(message);    // Inside if block
}
console.log(message);        // Hello
```

Demo

Block scope with let and var types

Immediately Invoked Function Expression (IIFE)

Function Expression

Define a function and assign it to a variable

Immediately Invoked

Invoking the function right away where it's defined

Function

```
function greeting() {
    console.log('Hello');
}
greeting();  // Hello
```

IIFE

```
(function() {
    console.log(“Hello”);    // Hello
})();
```

Closures

Closures

```
let greeting = (function() {
    let message = 'Hello';
    let getMessage = function() {
        return message;
    };
})();
console.log(greeting.message);    // undefined
```

Closures

```
let greeting = (function() {
    let message = 'Hello';
    let getMessage = function() {
        return message;
    };
    return {
        getMessage: getMessage,
    };
})();
console.log(greeting.getMessage());    // Hello
```

Closures

```
function setupCounter( val ) {
    return function counter() {
        return val++;
    }
}
let counter1 = setupCounter(0);
console.log(counter1());    // 0
console.log(counter1());    // 1
let counter2 = setupCounter(10);
console.log(counter2());    // 10
```

Demo

Using the IIFE pattern with Closures

Summary

- Function
- Arguments
- Function Scope
- Block Scope
- Immediately Invoked Function Expression (IIFE)
- Closures