Functions and Scope

Mark Zamoyta
SOFTWARE DEVELOPER AND EDUCATOR

@markzamoyta

Introduction

Function and Block Scope

IIFE's

Closures

The this Keyword

call(), apply() and bind()

Arrow Functions

Default Parameters

Function Scope

Function Scope

```
function startCar(carId) {

    let message = 'Starting...';

}

startCar(123);
console.log(message);     // undefined
```

Function Scope

```
function startCar(carId) {
    let message = 'Starting...';
    let startFn = function startCar() {
       console.log(message);   // 'Starting...'
    };
    startFn();
}
startCar(123);
```

Function Scope

```
function startCar(carId) {
    let message = 'Starting...';
    let startFn = function startCar() {
       let message = 'Override';
    };
    startFn();
    console.log(message);   // 'Starting...'
}
startCar(123);
```

Block Scope

Block Scope

```
if (5 === 5) {
    let message = 'Equal';
}
console.log(message);    // Error
```

Block Scope

```
let message = 'Outside';
if (5 === 5) {
    let message = 'Equal';
    console.log(message);   // Equal
}
console.log(message);    // Outside
```

IIFE's

IIFE

Immediately Invoked Function Expression

Function

```
function() {
    console.log('in function');
}
```

Immediately Invoked Function Expression

```
(function() {
    console.log('in function');
})();
```

IIFE

```
let app = (function() {
    let carId = 123;
    console.log('in function');
    return { };
})();
```

Closures

Example Closure

```
let app = (function() {
    let carId = 123;
    let getId = function() {
        return carId;
    };
    return {
        getId: getId
    };
})();
console.log( app.getId() );
```

The this Keyword

The this Keyword

```
let fn = function() {
    console.log (this === window);
};

fn();  // true
```

The this Keyword

```
let o = {
    carId: 123,
    getId: function() {
        return this.carId;
    }
};

console.log( o.getId() );  // 123
```

call and apply

call

```
let o = {
    carId: 123,
    getId: function() {
        return this.carId;
    }
};

let newCar = { carId: 456 };

console.log( o.getId.call(newCar) ); // 456
```

apply

```
let o = {
    carId: 123,
    getId: function(prefix) {
        return prefix + this.carId;
    }
};

let newCar = { carId: 456 };

console.log( o.getId.apply(newCar, ['ID: ']) );
// ID: 456
```

bind

bind

```
let o = {
    carId: 123,
    getId: function() {
        return this.carId;
    }
};

let newCar = { carId: 456 };

let newFn = o.getId.bind(newCar);

console.log( newFn() );  // 456
```

Arrow Functions

Arrow Functions

```
let getId = () => 123;

console.log( getId() );  // 123
```

Arrow Functions

```
let getId = prefix => prefix + 123;

console.log( getId('ID: ') );  // ID: 123
```

Arrow Functions

```
let getId = (prefix, suffix) => prefix + 123 + suffix;

console.log( getId('ID: ', '!') );  // ID: 123!
```

Arrow Functions

```
let getId = (prefix, suffix) => {
    return prefix + 123 + suffix;
};

console.log( getId('ID: ', '!') );  // ID: 123!
```

Arrow functions do not
have their own "this" value.

"this" refers to the
enclosing context.

Default Parameters

Default Parameters

```
let trackCar = function(carId, city='NY') {
    console.log(`Tracking ${carId} in ${city}.`);
};

console.log( trackCar(123) );
// Tracking 123 in NY.
console.log( trackCar(123, 'Chicago'));
// Tracking 123 in Chicago.
```

Summary

- Function and Block Scope
- IIFE's
- Closures
- The this Keyword
- call(), apply() and bind()
- Arrow Functions
- Default Parameters