JavaScript Objects and Functions

Introduction to JavaScript Objects

- Objects are collections of key-value pairs.
- Keys are strings, and values can be any data type.
- Objects are used to store and manage data.

```
const person = {
    name: "John",
    age: 30,
    city: "New York"
};
```

Accessing Object Properties

- Dot notation: `object.property`
- Bracket notation: `object["property"]`

```
console.log(person.name); // John
console.log(person["age"]); // 30
```

Modifying Object Properties

- Assign a new value to an existing property.
- Add a new property by assigning a value.

```
person.age = 31;
person.country = "USA";
console.log(person);
```

Deleting Object Properties

- Use the `delete` operator to remove a property.

```
delete person.city;
console.log(person);
```

Introduction to Functions

- Functions are reusable blocks of code.
- Defined using the `function` keyword or arrow syntax.

```
function greet(name) {
    return `Hello, ${name}!`;
}

const greetArrow = (name) => `Hello, ${name}!`;
```

Function Parameters and Arguments

- Parameters are placeholders in function definitions.
- Arguments are actual values passed to functions.

```
function add(a, b) {
    return a + b;
}

console.log(add(2, 3)); // 5
```

Function Return Values

- Functions can return values using the `return` statement.
- If no `return` is specified, the function returns `undefined`.

```
function multiply(a, b) {
    return a * b;
}

console.log(multiply(4, 5)); // 20
```

Function Expressions

- Functions can be assigned to variables.
- Useful for passing functions as arguments.

```
const square = function(x) {
    return x * x;
};

console.log(square(4)); // 16
```

Arrow Functions

- Shorter syntax for function expressions.
- Do not have their own `this` context.

```
const subtract = (a, b) ⇒ a - b;
console.log(subtract(10, 5)); // 5
```

Methods in Objects

- Functions can be properties of objects.
- Called methods when defined inside objects.

```
const calculator = {
    add: function(a, b) {
        return a + b;
    },
    subtract: (a, b) => a - b
};

console.log(calculator.add(5, 3)); // 8
console.log(calculator.subtract(9, 4)); // 5
```

`this` Keyword

- Refers to the object that is calling the method.
- Behavior depends on how the function is called.

```
const user = {
    name: "Alice",
    greet: function() {
        return `Hello, ${this.name}!`;
    }
};

console.log(user.greet()); // Hello, Alice!
```

Constructor Functions

- Used to create multiple objects with similar properties.
- Conventionally start with a capital letter.

```
function Person(name, age) {
    this.name = name;
    this.age = age;
}

const person1 = new Person("Bob", 25);
console.log(person1);
```

Prototypes

- Mechanism by which objects inherit properties.
- Each function has a `prototype` property.

```
Person.prototype.sayHello = function() {
    return `Hi, I'm ${this.name}`;
};

console.log(person1.sayHello()); // Hi, I'm Bob
```

Summary

- Objects store key-value pairs.
- Functions are reusable blocks of code.
- Methods are functions within objects.
- `this` keyword refers to the calling object.
- Constructor functions and prototypes enable object creation and inheritance.