ADVANCED

JAVASCRIPT

CONCEPT REVIEW

Mario Ruci | Senior Software Engineer

Learn more on
https://devmasters.pro

Inheritance

JavaScript implements prototypal inheritance, a unique and flexible mechanism that allows objects to inherit properties and methods directly from other objects. In this model, each object has an internal link to another object called its prototype, forming a chain of prototypes.

```
// Example of Inheritance
function Animal(name) {
  this.name = name;
}

Animal.prototype.sayHello = function() {
  console.log('Hello, I am ' + this.name);
};

function Dog(name, breed) {
  Animal.call(this, name);
  this.breed = breed;
}

Dog.prototype = Object.create(Animal.prototype);

Dog.prototype.bark = function() {
  console.log('Woof!');
};

var myDog = new Dog('Buddy', 'Labrador');
myDog.sayHello(); // Outputs: Hello, I am Buddy
myDog.bark(); // Outputs: Woof!
```

Prototype Chaining

In JavaScript, objects can inherit properties and methods from other objects forming a chain of prototypes.

```
// Example of Prototype Chaining
function Vehicle(make, model) {
  this.make = make;
  this.model = model;
}

Vehicle.prototype.drive = function() {
  console.log('Vroom!');
};

function Car(make, model, color) {
  Vehicle.call(this, make, model);
  this.color = color;
}

Car.prototype = Object.create(Vehicle.prototype);

Car.prototype.honk = function() {
  console.log('Honk!');
};

var myCar = new Car('Toyota', 'Camry', 'Blue');
myCar.drive(); // Outputs: Vroom!
myCar.honk(); // Outputs: Honk!
```

Memory Management in JavaScript

JavaScript uses automatic memory management through garbage collection to handle memory allocation and deallocation.

```
// Garbage Collection Cycle
// 1. Mark - Identify objects in use
// 2. Sweep - Remove unreferenced objects

// Example with Circular Reference
function createCircularReference() {
  var objA = { data: 'Object A' };
  var objB = { data: 'Object B' };

  objA.reference = objB;
  objB.reference = objA;

  // Circular reference, but both objects are unreachable
  objA = null;
  objB = null;
}

createCircularReference(); // After execution, both objA and objB are eligible for garbage colle
```

Concurrency Models in JavaScript

JavaScript follows a concurrency model based on an event loop, which allows asynchronous execution and handling of multiple tasks without blocking the main thread.

```
// Example of Asynchronous Execution
function asynchronousTask() {
  console.log('Start task');

  setTimeout(function() {
    console.log('Task completed after 2 seconds');
  }, 2000);

  console.log('Continuing with other tasks');
}

asynchronousTask(); // Outputs: Start task, Continuing with other tasks, Task completed after 2 s
```

Event Loop in JavaScript

The event loop is a fundamental part of the JavaScript concurrency model, managing the execution of code by continuously checking the call stack and the message queue.

```
// Example of Event Loop
console.log('Start script');

setTimeout(function() {
  console.log('Timeout callback executed');
}, 0);

console.log('End script');

// Outputs: Start script, End script, Timeout callback executed (after the main thread is free)
```

Enumerability of Properties in JavaScript

In JavaScript, enumerability determines whether a property will be iterated over in certain operations, like for...in loops. By default, properties added directly to an object are enumerable, but some built-in properties and methods may not be.

```
// Example of Enumerability
var person = { name: 'John', age: 30 };

// Making 'age' property non-enumerable
Object.defineProperty(person, 'age', { enumerable: false });

// Iterating over properties
for (var prop in person) {
  console.log(prop); // Outputs: name
}

console.log(Object.keys(person)); // Outputs: ['name']
```

Ownership of Properties in JavaScript

In JavaScript, property ownership refers to whether a property is directly defined on an object or inherited from its prototype chain. Understanding ownership is crucial when working with objects and their prototypes.

```
// Example of Property Ownership
function Animal() {
  this.legs = 4;
}

Animal.prototype.tail = true;

var cat = new Animal();
cat.sound = 'Meow';

// Checking property ownership
console.log(cat.hasOwnProperty('legs')); // Outputs: true
console.log(cat.hasOwnProperty('tail')); // Outputs: false
console.log(cat.hasOwnProperty('sound')); // Outputs: true
```

DEV MASTERS

WANT MORE?

LIKE | COMMENT | FOLLOW

Mario Ruci | Senior Software Engineer

Learn more on
https://devmasters.pro