In JavaScript, method overriding refers to the concept where a subclass (child class) provides a specific implementation of a method that is already defined in its superclass (parent class). This concept is widely used in object-oriented programming (OOP).

Here are some examples of method overriding in JavaScript:

Example 1: Basic Method Overriding with Classes

In this example, we have a `Car` class with a method `describe`, and a `SportsCar` subclass that overrides the `describe` method.

Javascript code:-

```
// Parent class
class Car {
  describe() {
    console.log("This is a car.");
  }
}

// Child class overriding the describe method
class SportsCar extends Car {
  describe() {
    console.log("This is a sports car.");
  }
}

const car = new Car();
car.describe(); // Output: This is a car.

const sportsCar = new SportsCar();
sportsCar.describe(); // Output: This is a sports car.
```

Example 2: Overriding with Constructor and Method in Subclass

Here, we override the `drive` method in the `ElectricCar` subclass, while also adding new properties.

Javascript code

```
// Parent class
class Car {
  constructor(make, model) {
    this.make = make;
    this.model = model;
  }

  drive() {
    console.log(`Driving a ${this.make} ${this.model}`);
  }
}
```

```
// Child class overriding the drive method
class ElectricCar extends Car {
  constructor(make, model, batteryLife) {
    super(make, model); // Call parent class constructor
    this.batteryLife = batteryLife;
  }

  // Overriding the drive method
  drive() {
    console.log(`Driving a ${this.make} ${this.model} with
${this.batteryLife}% battery left.`);
  }
}

const electricCar = new ElectricCar("Tesla", "Model S", 85);
electricCar.drive(); // Output: Driving a Tesla Model S with 85% battery
left.
```

Example 3: Method Overriding with `super`

You can call the method from the parent class using the `super` keyword inside the overridden method.

Javascript code

```
// Parent class
class Animal {
  speak() {
    console.log("Animal makes a sound");
  }
}

// Child class overriding the speak method
class Dog extends Animal {
  speak() {
    super.speak(); // Call the parent class method
    console.log("Dog barks");
  }
}

const dog = new Dog();
dog.speak();
// Output:
// Animal makes a sound
// Dog barks
```

Example 4: Method Overriding in Prototype Chain

JavaScript also supports method overriding through the prototype chain, although using classes is generally preferred in modern JavaScript.

Javascript code

```
// Parent function constructor
function Animal() {}

Animal.prototype.speak = function() {
  console.log("Animal speaks");
};

// Child function constructor
function Dog() {}

// Inheriting from Animal
Dog.prototype = Object.create(Animal.prototype);

// Overriding the speak method
Dog.prototype.speak = function() {
  console.log("Dog barks");
};

const animal = new Animal();
animal.speak(); // Output: Animal speaks

const dog = new Dog();
dog.speak(); // Output: Dog barks
```

Example 5: Overriding with Default and Custom Methods

A default method can be overridden with a more specific implementation in the subclass.

Javascript code

```
// Parent class
class Shape {
  area() {
    console.log("Calculating area of the shape");
  }
}

// Child class overriding the area method
class Rectangle extends Shape {
  constructor(width, height) {
    super();
    this.width = width;
    this.height = height;
  }

  area() {
    return this.width * this.height; // Custom implementation for Rectangle
```

```
  }
}

const shape = new Shape();
shape.area(); // Output: Calculating area of the shape

const rectangle = new Rectangle(5, 8);
console.log(rectangle.area()); // Output: 40
```

In these examples, the method in the child class (`SportsCar`, `ElectricCar`, `Dog`, `Rectangle`, etc.) overrides the method defined in the parent class (`Car`, `Animal`, `Shape`). The child class can call the parent class's method using `super`, or it can completely replace the parent class method with its own implementation.