Fundamentals of JavaScript Programming from the Ground Up

JavaScript is a versatile programming language that enables you to add interactive features to websites. It's used everywhere on the web and is essential for creating dynamic web content. Learning JavaScript involves understanding several core concepts, which we'll explore step by step, ensuring that the material is accessible even for kids.

Variables

Variables in JavaScript are like containers for storing data values. Think of them as labeled boxes where you can store something (like toys or books) and then find it again using the label on the box.

In JavaScript, you can declare a variable using `var`, `let`, or `const`. For example:

```
let age = 10; // 'let' allows you to change the value later
const name = "Alex"; // 'const' is for values that won't change
```

Conditional Logic (If-Else)

Conditional logic in JavaScript is like making a decision. If something is true, you do one thing; otherwise, you do something else. It's like deciding if it's raining: if it is (`if`), you take an umbrella; otherwise (`else`), you don't.

Here's a simple example:

```
if (age > 18) {
  console.log("You can vote!");
} else {
  console.log("You're too young to vote.");
}
```

Practice Codes with Solutions

Let's dive into some practice codes to understand these concepts better. Each code snippet is designed to be simple and educational, perfect for young learners.

1. **Hello World**

   - **Code**: Display "Hello, World!" in the console.

     ```
     console.log("Hello, World!");
     ```

   - **Explanation**: This code uses `console.log()` to print a message to the console, a basic starting point for JavaScript programming.

2. **Using Variables**

- **Code**: Store your age and display it.

```
let myAge = 12;
console.log("I am " + myAge + " years old.");
```

- **Explanation**: This code demonstrates how to use a variable (`myAge`) to store a number and then use it within a message.

3. **Basic Math**

- **Code**: Add two numbers and display the result.

```
let sum = 5 + 3;
console.log("The sum is " + sum + ".");
```

- **Explanation**: This shows how to perform a simple addition and store the result in a variable (`sum`).

4. **Changing Variables**

- **Code**: Change the value of a variable.

```
let count = 10;
count = 15; // Changing the value
console.log("The count is now " + count + ".");
```

- **Explanation**: Demonstrates how a variable's value can be updated after its initial declaration.

5. **Simple If-Else**

- **Code**: Use conditional logic to check if a number is greater than 10.

```
let number = 8;
if (number > 10) {
  console.log("The number is greater than 10.");
} else {
  console.log("The number is 10 or less.");
}
```

- **Explanation**: This introduces the `if-else` statement for making decisions based on conditions.

6. **If-Else with Variables**

- **Code**: Decide what to wear based on the weather.

```
let weather = "sunny";
if (weather === "rainy") {
  console.log("Don't forget your umbrella!");
} else {
  console.log("Wear your sunglasses!");
}
```

- **Explanation**: Shows how to use string variables and conditions to make decisions.

7. **Using `const`**

- **Code**: Declare a constant and try to change it.

```
const favoriteFood = "pizza";
console.log("My favorite food is " + favoriteFood + ".");
// Uncommenting the next line will cause an error because
'favoriteFood' is a constant.
// favoriteFood = "sushi";
```

- **Explanation**: Explains the concept of constants, which are variables whose values cannot be changed.

8. **Multiple Conditions**

- **Code**: Use `else if` to check multiple conditions.

```
let score = 85;
if (score >= 90) {
  console.log("A grade");
} else if (score >= 80) {
  console.log("B grade");
} else {
  console.log("Below B grade");
}
```

- **Explanation**: Demonstrates how to use `else if` for more complex decision-making.

9. **Combining Variables and Math**

- **Code**: Calculate the area of a rectangle.

```
let length = 5;
let width = 3;
```

```
let area = length * width;
console.log("The area of the rectangle is " + area + ".");
```

- **Explanation**: Shows how to use variables to store values and mathematical operations to calculate something (in this case, an area).

10. **Incrementing a Variable**

- **Code**: Increase a variable's value by 1.

```
let count = 0;
count = count + 1; // Incrementing
console.log("The count is " + count + ".");
```

- **Explanation**: Teaches how to update a variable by incrementing its value, a common operation in programming.

Exercises with Solutions

Now, let's reinforce what we've learned with some exercises. Each exercise is designed to practice the concepts of variables and conditional logic.

1. **Exercise**: Create a variable to store your name and print a greeting.

- **Solution**:

```
let myName = "Charlie";
console.log("Hello, " + myName + "!");
```

2. **Exercise**: Calculate and display the result of multiplying two numbers.

- **Solution**:

```
let number1 = 7;
let number2 = 4;
let result = number1 * number2;
console.log("The result is " + result + ".");
```

3. **Exercise**: Use a variable to store a number and then check if it is odd or even using `if-else`.

- **Solution**:

```
let number = 9;
```

```
if (number % 2 === 0) {
  console.log("The number is even.");
} else {
  console.log("The number is odd.");
}
```

4. **Exercise**: Store the current temperature in a variable and print whether it's hot or cold.

- **Solution**:

```
let temperature = 30;
if (temperature > 25) {
  console.log("It's hot outside!");
} else {
  console.log("It's cold outside!");
}
```

5. **Exercise**: Create a variable for a password and check if it's correct.

- **Solution**:

```
let password = "secret123";
if (password === "secret123") {
  console.log("Access granted.");
} else {
  console.log("Access denied.");
}
```

6. **Exercise**: Calculate the perimeter of a square and log it to the console.

- **Solution**:

```
let sideLength = 5;
let perimeter = sideLength * 4;
console.log("The perimeter of the square is " + perimeter + ".");
```

7. **Exercise**: Store your age in a variable and check if you're a teenager (age 13-19).

- **Solution**:

```
let myAge = 15;
if (myAge >= 13 && myAge <= 19) {
  console.log("You're a teenager.");
```

```
} else {
  console.log("You're not a teenager.");
}
```

8. **Exercise**: Create a variable for the day of the week and print a message if it's Monday.

- **Solution**:

```
let dayOfWeek = "Monday";
if (dayOfWeek === "Monday") {
  console.log("It's the start of the week!");
}
```

9. **Exercise**: Store a number in a variable and print whether it's positive, negative, or zero.

- **Solution**:

```
let number = -5;
if (number > 0) {
  console.log("The number is positive.");
} else if (number < 0) {
  console.log("The number is negative.");
} else {
  console.log("The number is zero.");
}
```

10. **Exercise**: Create a variable to store the number of apples you have. If you have more than 10 apples, print "Lots of apples!" Otherwise, print "Not many apples."

- **Solution**:

```
let apples = 8;
if (apples > 10) {
  console.log("Lots of apples!");
} else {
  console.log("Not many apples.");
}
```

Arrays

Arrays in JavaScript are like lists that can store multiple items under a single name. You can think of an array as a row of boxes, each containing something different, but all stored together on the same shelf.

For example, to store a list of your favorite fruits, you can create an array like this:

```
let fruits = ["Apple", "Banana", "Cherry"];
```

You can access each item using its position (index) in the list, but remember, in JavaScript, counting starts from 0. So, `fruits[0]` is "Apple".

Objects

Objects in JavaScript are collections of properties, where a property is an association between a name (or key) and a value. You can think of an object as a box with labeled compartments, each holding a specific thing.

For example, to describe a book, you might use an object like this:

```
let book = {
  title: "The Little Prince",
  author: "Antoine de Saint-Exupéry",
  yearPublished: 1943
};
```

You can access the properties of an object using either dot notation (`book.title`) or bracket notation (`book["title"]`).

Loops

Loops in JavaScript are used to perform repetitive tasks. It's like telling your computer to jump rope and count each jump until it reaches a certain number.

A common loop in JavaScript is the `for` loop, which has three parts: the starting point, the condition to keep going, and what to do after each loop (like incrementing a counter).

For example, to log numbers 1 through 5 to the console:

```
for (let i = 1; i <= 5; i++) {
  console.log(i);
}
```

Practice Codes with Solutions

Arrays

1. **Creating an Array**

    - **Code**:

```
let colors = ["Red", "Green", "Blue"];
console.log(colors);
```

- **Explanation**: This creates an array named `colors` with three items and prints the whole array.

2. **Accessing Array Elements**

- **Code**:

```
console.log("My favorite color is " + colors[0]);
```

- **Explanation**: Accesses the first element (`"Red"`) from the `colors` array and prints it.

3. **Changing an Array Element**

- **Code**:

```
colors[2] = "Yellow";
console.log(colors);
```

- **Explanation**: Changes the third element of the `colors` array from `"Blue"` to `"Yellow"`.

4. **Array Length**

- **Code**:

```
console.log("There are " + colors.length + " colors in the array.");
```

- **Explanation**: Uses the `.length` property to find out how many items are in the `colors` array.

5. **Adding to an Array**

- **Code**:

```
colors.push("Purple");
console.log(colors);
```

- **Explanation**: Adds a new item (`"Purple"`) to the end of the `colors` array.

Objects

6. Creating an Object

- **Code**:

```
let cat = {
  name: "Whiskers",
  age: 3,
  color: "gray"
};
console.log(cat);
```

- **Explanation**: Defines an object `cat` with properties for its name, age, and color, then prints the object.

7. Accessing Object Properties

- **Code**:

```
console.log(cat.name + " is " + cat.age + " years old.");
```

- **Explanation**: Accesses the `name` and `age` properties of the `cat` object and combines them in a message.

8. Adding a New Property

- **Code**:

```
cat.breed = "Persian";
console.log(cat);
```

- **Explanation**: Adds a new property `breed` to the `cat` object.

Loops

9. Looping Through an Array

- **Code**:

```
for (let i = 0; i < colors.length; i++) {
  console.log("Color at position " + i + " is " + colors[i]);
}
```

- **Explanation**: Uses a `for` loop to go through each item in the `colors` array and prints its position and value.

10. **Looping Through an Object's Properties**

- **Code**:

```
for (let property in cat) {
  console.log(property + ": " + cat[property]);
}
```

- **Explanation**: Uses a `for-in` loop to iterate over each property in the `cat` object and prints the property name and its value.

Exercises with Solutions

Arrays

1. **Exercise**: Create an array of your top three favorite movies.

- **Solution**:

```
let movies = ["Toy Story", "Finding Nemo", "The Lion King"];
console.log(movies);
```

2. **Exercise**: Add a new movie to the end of the `movies` array and print the updated array.

- **Solution**:

```
movies.push("Shrek");
console.log(movies);
```

3. **Exercise**: Replace the second movie in your array with a new one and print the whole array.

- **Solution**:

```
movies[1] = "Frozen";
console.log(movies);
```

Objects

4. **Exercise**: Create an object to describe your favorite book, including properties for the title, author, and number of pages.

    - **Solution**:

```
let favoriteBook = {
  title: "Harry Potter and the Sorcerer's Stone",
  author: "J.K. Rowling",
  pages: 309
};
console.log(favoriteBook);
```

5. **Exercise**: Add a property to your book object for the year it was published and print the updated object.

    - **Solution**:

```
favoriteBook.yearPublished = 1997;
console.log(favoriteBook);
```

Loops

6. **Exercise**: Use a loop to print each movie in your `movies` array on a new line.

    - **Solution**:

```
for (let i = 0; i < movies.length; i++) {
  console.log(movies[i]);
}
```

7. **Exercise**: Create an array of numbers and use a loop to calculate the sum.

    - **Solution**:

```
let numbers = [1, 2, 3, 4, 5];
let sum = 0;
for (let i = 0; i < numbers.length; i++) {
  sum += numbers[i];
}
console.log("Sum: " + sum);
```

8. **Exercise**: Use a `for-in` loop to print all the properties of your favorite book object.

    - **Solution**:

```
for (let prop in favoriteBook) {
  console.log(prop + ": " + favoriteBook[prop]);
}
```

9. **Exercise**: Create an array of your favorite foods. Use a loop to print "I love [food]" for each item.

    - **Solution**:

```
let favoriteFoods = ["Pizza", "Ice Cream", "Chocolate"];
for (let i = 0; i < favoriteFoods.length; i++) {
  console.log("I love " + favoriteFoods[i]);
}
```

10. **Exercise**: Create an object representing a student, including properties for their name, grade, and a list of subjects they are taking. Use a loop to print each subject.

    - **Solution**:

```
let student = {
  name: "Emma",
  grade: 10,
  subjects: ["Math", "Science", "History"]
};
for (let i = 0; i < student.subjects.length; i++) {
  console.log(student.name + " is taking " + student.subjects[i]);
}
```

Newest ES6 and ES7 Features

ES6 (ECMAScript 2015) and ES7 (ECMAScript 2016) introduced several new features to JavaScript, making the language more powerful and easier to work with. Let's break down some of these features in a way that's accessible to everyone, including kids.

Arrow Functions

Arrow functions provide a shorter syntax for writing functions. Instead of using the `function` keyword, you use an arrow (`=>`) to define a function. Imagine an arrow function as a slingshot that quickly sends input to an output.

Traditional Function:

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

Arrow Function:

```
const add = (a, b) => a + b;
```

`let` and `const`

Before ES6, `var` was the only way to declare variables. ES6 introduced `let` and `const` for more flexible and safer variable declarations.

- `let`: Use it when the variable's value might change. It's like saying, "Let this be something now, but it can be something else later."
- `const`: Use it for variables that won't change. It's like making a promise that the value will stay constant.

Functions, Parameters-Arguments, and Return Values

Functions are like magic recipes that take some ingredients (parameters), do something magical (the code inside the function), and then produce a magic potion (return value). The parameters are what you give to the function, and the return value is what you get back.

Practice Codes with Solutions

Let's explore these concepts through some fun and simple practice codes.

1. **Simple Arrow Function**
    - **Code**: Create an arrow function that adds two numbers.

      ```
      const add = (a, b) => a + b;
      console.log(add(5, 3)); // Output: 8
      ```

    - **Explanation**: This arrow function takes two numbers, adds them, and returns the result.
2. **Arrow Function with No Parameters**
    - **Code**: An arrow function that returns "Hello, World!".

      ```
      const sayHello = () => "Hello, World!";
      ```

```
console.log(sayHello()); // Output: Hello, World!
```

- **Explanation**: This function doesn't need any ingredients (parameters); it always gives back the same greeting.

3. **Using `let` to Declare Variables**

- **Code**: Use `let` to declare a variable and change its value.

```
let age = 10;
age = 12; // It's okay to change it
console.log("Age is " + age); // Output: Age is 12
```

- **Explanation**: We declare `age` with `let` because we know it might change as someone gets older.

4. **Using `const` to Declare Constants**

- **Code**: Use `const` to declare a constant value.

```
const birthYear = 2010;
console.log("Birth Year: " + birthYear); // Output: Birth Year: 2010
```

- **Explanation**: The birth year won't change, so we use `const` to make it constant.

5. **Function with Return Value**

- **Code**: Create a function that returns the square of a number.

```
function square(number) {
  return number * number;
}
console.log(square(4)); // Output: 16
```

- **Explanation**: This recipe (function) takes a number, multiplies it by itself, and gives back the result.

6. **Arrow Function with Multiple Lines**

- **Code**: Write an arrow function that checks if a number is even.

```
const isEven = (number) => {
  return number % 2 === 0;
};
console.log(isEven(10)); // Output: true
```

- **Explanation**: This function uses curly braces `{}` for multiple lines of code, ending with a `return` statement.

7. **Default Parameters**

- **Code**: Create a function with default parameters.

```
const greet = (name = "friend") => "Hello, " + name + "!";
console.log(greet("Alice")); // Output: Hello, Alice!
console.log(greet()); // Output: Hello, friend!
```

- **Explanation**: If we don't provide a name, the function uses "friend" as a default.

8. **Function Returning Another Function**

- **Code**: A function that creates and returns a greeting function.

```
function createGreeting(greeting) {
  return function(name) {
    return greeting + ", " + name + "!";
  };
}
const sayHello = createGreeting("Hello");
console.log(sayHello("Charlie")); // Output: Hello, Charlie!
```

- **Explanation**: This is like a magic kit that creates customized greeting spells.

9. **`const` with Arrays**

- **Code**: Use `const` with an array and modify the array's contents.

```
const fruits = ["apple", "banana"];
fruits.push("orange"); // We can add to the array
console.log(fruits); // Output: ["apple", "banana", "orange"]
```

- **Explanation**: Even though `fruits` is a constant, we can still change the contents of the array.

10. **Template Literals**

- **Code**: Use template literals to include variables in a string.

```
let animal = "dog";
let action = "bark";
```

```
console.log(`The ${animal} goes ${action}!`); // Output: The dog goes bark!
```

- **Explanation**: Template literals (backticks `` ` ``) make it easy to mix text with variables, like filling in the blanks in a story.

Exercises with Solutions

Now, let's practice these concepts with some exercises.

1. **Exercise**: Write an arrow function that multiplies two numbers.
    - **Solution**:

      ```
      const multiply = (a, b) => a * b;
      ```

2. **Exercise**: Create an arrow function that takes a name and returns a greeting message using that name.
    - **Solution**:

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

3. **Exercise**: Use `let` to declare a variable for your age, then change it to represent someone else's age.
    - **Solution**:

      ```
      let myAge = 10;
      myAge = 8; // Changed to another age
      ```

4. **Exercise**: Declare a `const` for the number of days in a week and print it.
    - **Solution**:

      ```
      const daysInWeek = 7;
      console.log(daysInWeek);
      ```

5. **Exercise**: Write a function that returns the remainder of a division operation.
    - **Solution**:

```
function remainder(a, b) {
  return a % b;
}
```

6. **Exercise**: Create an arrow function that checks if a number is less than 10.

   - **Solution**:

   ```
   const isLessThanTen = (number) => number < 10;
   ```

7. **Exercise**: Write a function with a default parameter that prints a greeting.

   - **Solution**:

   ```
   const greet = (name = "there") => `Hi, ${name}!`;
   ```

8. **Exercise**: Create a function that returns an arrow function, which in turn returns the addition of two numbers.

   - **Solution**:

   ```
   function getAdder() {
     return (a, b) => a + b;
   }
   ```

9. **Exercise**: Using `const`, declare an array of your favorite fruits and add another fruit to it.

   - **Solution**:

   ```
   const favoriteFruits = ["apple", "banana"];
   favoriteFruits.push("mango");
   ```

10. **Exercise**: Write a function that uses template literals to return a person's name and age in a sentence.

    - **Solution**:

    ```
    function introduce(name, age) {
      return `My name is ${name} and I am ${age} years old.`;
    }
    ```

Higher Level Functions and Callbacks

In JavaScript, functions are not just blocks of code to run; they can also be passed around like any other value. This means you can give a function as an argument to another function. These "passed-in" functions are known as **callbacks**.

Imagine you have a robot that can perform tasks for you. You can give this robot a list of tasks (functions) and also tell it to run a special task (callback) when it finishes the main tasks. This is very similar to how callbacks work in JavaScript.

Array and String Methods

Arrays and strings in JavaScript come with a set of built-in methods that allow you to manipulate them easily. These methods can transform the data, retrieve specific elements, and more. For arrays, methods like `forEach`, `map`, `filter`, and `reduce` are particularly powerful when combined with callback functions. For strings, methods like `toUpperCase`, `toLowerCase`, `split`, and `includes` provide various ways to handle and manipulate text.

Practice Codes with Solutions

Let's explore some practice codes that demonstrate the use of higher-level functions, callbacks, and array/string methods. Each code snippet is designed to be engaging and understandable, even for kids.

1. **Using `forEach` to Log Array Items**

    - **Code**:

      ```
      let fruits = ["apple", "banana", "cherry"];
      fruits.forEach(function(fruit) {
        console.log(fruit);
      });
      ```

    - **Explanation**: This code goes through each item in the `fruits` array and prints it out. Think of `forEach` as telling your robot to go through a list and show each item to you.

2. **Transforming Array Items with `map`**

    - **Code**:

      ```
      let numbers = [1, 2, 3];
      let squares = numbers.map(function(number) {
        return number * number;
      });
      ```

```
console.log(squares);
```

- **Explanation**: This example takes an array of numbers and creates a new array (`squares`) with each number squared. The `map` method is like telling your robot to change each item in a list according to your instructions.

3. **Filtering an Array with `filter`**

- **Code**:

```
let ages = [5, 12, 18, 21, 30];
let adults = ages.filter(function(age) {
  return age >= 18;
});
console.log(adults);
```

- **Explanation**: This shows how to use `filter` to create a new array that only includes adults (ages 18 and above). It's like telling your robot to pick out only certain items from a list based on a rule.

4. **Combining Array Items with `reduce`**

- **Code**:

```
let numbers = [1, 2, 3, 4];
let sum = numbers.reduce(function(total, number) {
  return total + number;
}, 0);
console.log(sum);
```

- **Explanation**: This code adds up all the numbers in an array. The `reduce` method is like telling your robot to combine all items in a list into one, step by step.

5. **Changing String Case**

- **Code**:

```
let message = "Hello World";
console.log(message.toUpperCase()); // Converts to uppercase
console.log(message.toLowerCase()); // Converts to lowercase
```

- **Explanation**: Demonstrates how to change the case of all characters in a string using `toUpperCase` and `toLowerCase`.

6. **Splitting a String into an Array**

- **Code**:

```
let sentence = "JavaScript is fun";
let words = sentence.split(" "); // Splits the sentence into words
console.log(words);
```

- **Explanation**: This shows how to turn a sentence into an array of words using `split`, separating the string wherever it finds a space.

7. **Checking If an Array Includes an Item**

- **Code**:

```
let pets = ["dog", "cat", "rabbit"];
let hasCat = pets.includes("cat");
console.log("Has a cat: " + hasCat);
```

- **Explanation**: Demonstrates how to check if an array includes a certain item using `includes`, which returns `true` or `false`.

8. **Finding the First Matching Item with `find`**

- **Code**:

```
let books = [
  { title: "The Hobbit", author: "J.R.R. Tolkien" },
  { title: "Harry Potter", author: "J.K. Rowling" }
];
let book = books.find(function(book) {
  return book.title === "The Hobbit";
});
console.log(book);
```

- **Explanation**: This example finds the first item in an array that matches a condition. Here, it finds a book by its title.

9. **Using `slice` to Copy Part of an Array**

- **Code**:

```
let colors = ["red", "green", "blue", "yellow"];
let primaryColors = colors.slice(0, 3);
console.log(primaryColors);
```

- **Explanation**: Shows how to create a new array (`primaryColors`) that contains only a part of the original `colors` array, using `slice`.

10. **Replacing Parts of a String with `replace`**

- **Code**:

```
let greeting = "Hello, Jane!";
let newGreeting = greeting.replace("Jane", "John");
console.log(newGreeting);
```

- **Explanation**: Demonstrates how to replace part of a string with another string using `replace`. Here, it changes the name in a greeting message.

Exercises with Solutions

Now, let's solidify your understanding with some exercises that apply these concepts in new ways.

1. **Exercise**: Create an array of your favorite foods and use `forEach` to print each one.

- **Solution**:

```
let favoriteFoods = ["Pizza", "Ice Cream", "Tacos"];
favoriteFoods.forEach(function(food) {
  console.log(food);
});
```

2. **Exercise**: Given an array of numbers, use `map` to create a new array with each number doubled.

- **Solution**:

```
let numbers = [2, 4, 6];
let doubled = numbers.map(function(number) {
  return number * 2;
});
console.log(doubled);
```

3. **Exercise**: Use `filter` to find all even numbers in an array.

- **Solution**:

```
let numbers = [1, 2, 3, 4, 5, 6];
```

```
let evens = numbers.filter(function(number) {
  return number % 2 === 0;
});
console.log(evens);
```

4. **Exercise**: Sum all numbers in an array using `reduce`.

- **Solution**:

```
let numbers = [1, 2, 3, 4];
let total = numbers.reduce(function(sum, number) {
  return sum + number;
}, 0);
console.log(total);
```

5. **Exercise**: Convert a string to an array of words using `split`.

- **Solution**:

```
let phrase = "I love JavaScript";
let words = phrase.split(" ");
console.log(words);
```

6. **Exercise**: Check if an array of pets includes "hamster".

- **Solution**:

```
let pets = ["dog", "cat", "fish"];
let includesHamster = pets.includes("hamster");
console.log("Includes hamster: " + includesHamster);
```

7. **Exercise**: Find the first number greater than 10 in an array.

- **Solution**:

```
let numbers = [3, 8, 11, 9];
let firstGreaterTen = numbers.find(function(number) {
  return number > 10;
});
console.log(firstGreaterTen);
```

8. **Exercise**: Create a new array that contains only the first 2 elements of the original array.

- **Solution**:

```
let colors = ["red", "green", "blue", "yellow"];
let firstTwo = colors.slice(0, 2);
console.log(firstTwo);
```

9. **Exercise**: Replace "goodbye" with "hello" in a string.

- **Solution**:

```
let farewell = "Goodbye, world!";
let greeting = farewell.replace("Goodbye", "Hello");
console.log(greeting);
```

10. **Exercise**: Given an array of names, use `map` to create an array of greetings.

- **Solution**:

```
let names = ["Alice", "Bob", "Charlie"];
let greetings = names.map(function(name) {
  return "Hello, " + name + "!";
});
console.log(greetings);
```