Javascript Continued

Sheet 4

1 Javascript Objects

1. Create an object that represents a person, with properties for their name, age, and address. Use object references to create a second object that represents a family, with properties for the parents and children.

2. Write a function that takes an object as an argument and returns a deep copy of the object, so that any changes made to the copy won't affect the original.

3. Create an object that represents a shopping cart, with properties for the items and their prices. Write a function that calculates the total cost of the items in the cart.

4. Write a function that takes an object as an argument and deletes any properties with a value of null or undefined.

5. Create an object that represents a book, with properties for the title, author, and publication date. Write a method that returns the number of years since the book was published.

6. Write a constructor function for a car, with properties for the make, model, and year. Add a method to the prototype that calculates the age of the car.

7. Write a function that takes an object as an argument and returns a string representation of the object, using the object's properties and values.

8. Create an object that represents a bank account, with properties for the account number, balance, and interest rate. Write a method that calculates the interest earned on the account over a specified period of time.

9. Write a function that takes an object as an argument and adds a new property to the object, with a value of the current date and time.

10. Create an object that represents a music playlist, with properties for the songs and their artists. Use optional chaining to access a property of a nested object, and return a default value if the property doesn't exist.

2 Object.keys, values, entries

1. Write a function that takes an object as input and returns an array containing all keys of the object.
2. Write a function that takes an object as input and returns an array containing all values of the object.
3. Write a function that takes an object as input and returns an array containing all key-value pairs of the object as arrays.
4. Write a function that takes an object and a key as inputs and returns true if the object has the specified key, otherwise false.
5. Write a function that takes an object as input and returns the number of keys in the object.

3 Strings

1. Write a function that takes a string as an input and returns the same string with the first character capitalized.
2. Write a function that takes a string as an input and returns the same string with all whitespace removed.
3. Write a function that takes a string as an input and returns the same string with all vowels replaced by '*'.
4. Write a function that takes a string as an input and returns true if the string is a palindrome (reads the same forwards and backwards), otherwise false.

4 Arrays

1. Write a function that takes an array of numbers as input and returns the sum of the numbers.
2. Write a function that takes an array of strings as input and returns the length of the longest string.
3. Write a function that takes an array of numbers as input and returns a new array with all even numbers removed.
4. Write a function that takes two arrays as input and returns a new array that contains all elements that are in both arrays.
5. Write a function that takes an array of numbers as input and returns a new array with all numbers squared.

5 Map and Set

1. Write a function that takes an array of numbers as input and returns a new array with all duplicates removed.
2. Write a function that takes an array of strings as input and returns a new array with all duplicates removed.
3. Write a function that takes an array of objects as input and returns a new array with all objects sorted by a specified property (e.g. age).
4. Write a function that takes an iterable (e.g. array, string) as input and returns a new set containing all unique elements.
5. Write a function that takes two sets as input and returns a new set that contains all elements that are in either set.

6 Classes

1. Create a class called `Person` with a constructor that takes a `name` and `age` parameter. Add a method called `greet` that returns a greeting with the person's name and age. Instantiate a `Person` object and call the `greet` method.
2. Create a class called `Student` that extends `Person`. Add a constructor that takes a `name`, `age`, and `grade` parameter. Override the `greet` method to include the person's grade in the greeting. Instantiate a `Student` object and call the `greet` method.
3. Create a class called `Shape` with a constructor that takes `x` and `y` coordinates. Add a method called `move` that updates the `x` and `y` coordinates. Instantiate a `Shape` object and call the `move` method.
4. Add a static method to the `Shape` class called `getDistance` that takes two `Shape` objects as parameters and calculates the distance between them using the Pythagorean theorem. Instantiate two `Shape` objects and call the `getDistance` method.
5. Create a class called `Rectangle` that extends `Shape`. Add a constructor that takes `x`, `y`, `width`, and `height` parameters. Override the `move` method to update the `x` and `y` coordinates as well as the `x` and `y` coordinates of the opposite corner of the rectangle. Instantiate a `Rectangle` object and call the `move` method.
6. Add a static method to the `Rectangle` class called `getArea` that takes a `Rectangle` object as a parameter and calculates its area. Instantiate a `Rectangle` object and call the `getArea` method.

7. Create a class called `BankAccount` with a constructor that takes a `name` and `balance` parameter. Add a method called `deposit` that adds an amount to the balance. Add a method called `withdraw` that subtracts an amount from the balance. Create a `checking` and `savings` account and call the `deposit` and `withdraw` methods on each.

8. Create a class called `Employee` with private properties called `name` and `salary`. Add a public method called `getSalary` that returns the salary. Instantiate an `Employee` object and call the `getSalary` method.

9. Create a class called `Manager` that extends `Employee`. Add a public method called `giveRaise` that increases the salary by a specified amount. Instantiate a `Manager` object and call the `giveRaise` method.

10. Create a class called `Animal` with a protected property called `name`. Add a public method called `getName` that returns the name. Instantiate an `Animal` object and call the `getName` method.