6. Introduction to object-oriented programing

6.1 The object-oriented data model

- Using structured data like array is very helpful. But not enough to effectively describe real objects to computers. For that we use data models to better describe entities.
- **Data model**: A model that organizes elements of data and standardizes how they relate to one. Since data elements document real life, people, places and things and the events between them, the data model represents reality. For example, a car has tires, or cows have four legs. One of the most commonly used data model is the Object-Oriented Data Model
- **What is Object Oriented Data Model?**
    - We know that humans understand real word situations like people, places and things in general as objects. Object oriented data modeling is way by which these real-world situations are explained/represented as objects to computers.
- **Components of object-oriented data model**: The structure, or building blocks, of object-oriented programming include the following:
    - **Objects**: objects are things obtained from a real-world entity. Objects in programming language can correspond to real-world objects or an abstract entity. Objects are the things we are trying to explain to the computer.
    - **Attribute**: describes the properties of an object. Properties are characteristics of the thing we are describing. For example, a PERSON object includes the attributes Name, Social Security Number, and Date of Birth
    - **Method/function(action)**: Method represents the behavior of an object. Basically, it represents the real-world action. It answers the question “what does the object do?”
    - **Events**: How does an object/thing interact with other things and what happens to the thing when things interact with it?
    - **Class**: means of grouping all the objects which share the same set of attributes and methods. For example, if we have a PERSON class, a CUSTOMER class or an EMPLOYEE class share a parent PERSON class. Note that an object is an instance of a class (e.g., a particular person, place, or thing)
        - **Note**: we should know unlike other object-oriented languages, there is no classes in JavaScript, only object. To be more precise, JavaScript is a prototype-based object-oriented language, which means it doesn’t have

classes rather it defines behaviors using constructor function and then reuse it using the prototype. We will discuss this later.

- **Inheritance**: By using inheritance, new class can inherit the attributes and methods of the old class i.e., base class. Unlike most of the object-oriented programming languages where classes inherit classes, JavaScript object inherits Object i.e., certain features (property and methods) of one object can be reused by other objects.

- **Example to describe a Car using the object-oriented data model**
  - **Object**: Let’s assume we have a “car” object
  - **Properties of the object**: model, make, color, doors
  - **Methods/Actions**: Car Starts (startCar), Car Accelerates (accelerate), Car Decelerates (decelerate)
  - **Events**: enter > startCar, up arrow > accelerate, down arrow > decelerate
    - **Note**: At this point, the computer would have some understanding of what a car is. It still doesn't know everything about a car. For example, it doesn't know it has windows or tire

6.2 Object oriented programming in JavaScript

- **Definition of object-oriented programming (OOP)**:
  - OOP is an approach to problem solving where all computations are carried out using objects.
  - The basic idea of OOP is that we use objects to model real world things that we want to represent inside our JavaScript programs, and/or provide a simple way to access functionality that would otherwise be hard or impossible to make use of.
  - The focus of OOP is objects that developers want to manipulate, not the logic required to manipulate them. Most modern-day programing languages, including JavaScript, use this model.
- **Meaning of object in JavaScript**:
  - We learned about primitive and non- primitive structured data types in JavaScript. An object is a non-primitive, structured data type in JavaScript. Objects are variables too. But objects can contain many values in terms of properties and methods.
  - A JavaScript object is a collection of properties and methods, defined as a key-value pair. Each property has a key and a value.

        - You can think of an object as an associative array (a.k.a. map, dictionary, hash, lookup table). The keys in this array are the names of the object's properties

- **Creating objects in JavaScript**: There are two ways of creating objects in JavaScript; using object literal and using Objects() constructor function
    - **1. Creating object using object literal:** The object literal is a short form of creating an object. This is a way of defining an object in the { } brackets with "key/name: value" pairs separated by a comma. The key would be the name of the property and the value will be a literal value or a function.
        - **Object literal syntax**: var object-name = {key1: value1, key2: value2};
            - Example:

```
let person = {
firstName: 'Abebe',
lastName: 'Kebede'
someFunction: function () {
return this.firstName;
}
};
```

            - FirstName, lastName and someFunction are the properties of person object
            - Abebe, Kebede and function () {return this.firstName;} are property values
        - **Note**: The whole key-value pair must be declared. Declaring only a key without a value is invalid
        - **Object Methods**: Methods can be stored as properties of an object. Methods are actions that can be performed on objects.

```
objectName.functionName = function () {
// Function definition goes here
 }
```

    - Take a look at someFunction above as a property for our person object. In short, methods are functions stored in an object as one of its properties. Therefore, the person object owns the someFunction.

- **Accessing object properties**: Here the property_name is just a string or symbol. So, it can be any string, including '1foo', '!bar!', or even ' ' (a space)
    - **The dot (.) notation**
        - **Syntax**: objectName.propertyName
        - Example:

```
let person = {
firstName: 'Abebe',
lastName: 'Kebede',
someFunction: function () {
return this.firstName;
}
};
console.log(person.firstName);
```

    - **Bracket notation (array-like notation) ([ ])**:
        - **Syntax**: objectName['propertyName'];
        - Example: console.log(person['firstName']); //prints Abebe in the console
    - **Accessing object methods ()**
        - **Syntax**: objectName.methodName()
        - Example: console.log (person. SomeFunction())
        - Note: If you access a method without the () parentheses, it will return the function definition.
            - console.log (person. SomeFunction) will print the SomeFunction function, not the Abebe value

        - **Accessing Nested Objects**: An object can be a property of another object. It is called a nested object.

```
var person = {
    firstName: "James",
    lastName: "Bond",
    address: {
        city: “London”,
        country:"UK"
    }
};
 person.address.city; // returns "London"
```

    - **The "this" value/keyword:** Methods (functions within an object) usually need to access and use data stored in the object. The "this" value references the "owner" of the function, which is the object.
        - **Example**:

```
let person = {
firstName: 'Abebe',
lastName: 'Kebede',
someFunction: function () {
return this.firstName;
}
};
```

        - In the example above, “this” is the person object that "owns" the someFunction function. In other words, this.firstName means the firstName property of this object

- **2. Creating object using object() constructor function:** In addition to object literal, JavaScript objects can be created using constructor functions. Because there are a lot of things we will learn about constructor functions, we have assigned one section to discuss about creating objects using constructor function.

6.3 JavaScript object constructors

- **Constructor functions in JavaScript**:
  - **What is a constructor function?** A constructor is a function that initializes/creates an object. It is a template/blueprint for creating objects with the same methods and same properties (but different values for non-method properties)
  - **Conventions when creating constructor functions**: Constructor functions are technically regular functions but have special properties than regular functions. There are 2 conventions for constructor functions:
    - **1. They are named with capital letter first**: It is considered good practice to name constructor functions with an upper-case first letter.
    - **2. They should be executed only with the "new" operator**: The operator "new" will create a new blank object by calling the constructor function and makes the "this" key word to point at the newly created object.
  - **Use the "function" key word to create constructor functions**: We need to use the "function" keyword to define a constructor function because arrow functions can't be used to define constructor functions.
    - **Let's create an object constructor function called, Person**

```
function Person (first, last) {
this.firstName = first;
this.lastName = last;
}
```

    - **Let's create a new** object **(an instance of the object) using Person constructor function**

```
var myClassMate = new Person("Abebe", "Kebede");
```

  - **Adding a method/function to a constructor function**
    - **Let's create an object constructor function Person that has a method as property**

```
function Person(first, last) {
this.firstName = first;
this.lastName = last;
this.fullName = function() {
return this.firstName + " " + this.lastName;
};
}
```

- **Creating an instance of the object**

```
var myClassMate = new Person("Sara",
"Solomon");
console.log(myClassMate.fullName());// prints
Sara Solomon
```

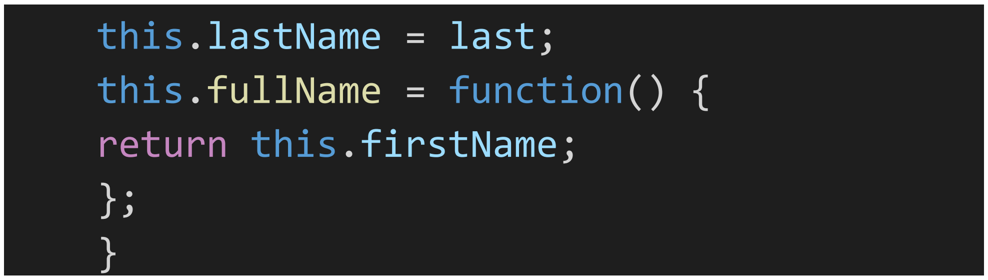
- **The value of "this", when used in an object, is the object itself**

```
var student = {
name: "Martin",
class: "12th",
studentDetails: function () {
return this.name ;
},
};
```

- The value of "this" in the above student object is the object itself

- **In a constructor function, "this" is a substitute for a future/new object that will be created using the constructor function:** Meaning, the value of "this" will be the new object after and when a new object is created.
  - Example:

```
function Person(first, last) {
    this.firstName = first;
```



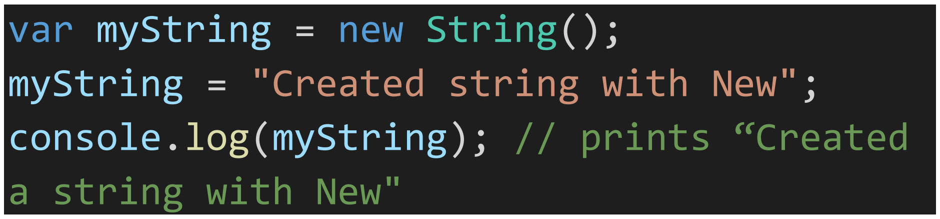
```
this.lastName = last;
this.fullName = function() {
return this.firstName;
};
}
```

        - In the constructor function Person above, “this” does not have a value. The value of this will become the new object when a new object is created using the “new” operator
    - **The “this” key word is not a variable**: “this” in JavaScript is key word, not a variable, so, we can’t change the value of “this”

6.4 Mostly used built-in JavaScript object (string object)

- We have discussed above that a JavaScript method is a property containing a function definition. Meaning, when the data stored on an object is a function, we call that a method.
    - **Properties vs methods in JavaScript**: A property is what an object has, while a method is what an object does. JavaScript methods are actions that can be performed on objects
- **Built–in JavaScript objects**: JavaScript supports a number of built-in objects that extend the flexibility of the language. Several of these objects are "borrowed" from the Java language, but JavaScript's implementation of them is different.
    - **Note**: Just like a regular JavaScript object, built-in JavaScript objects have properties and methods
    - **The most common built-in JavaScript Objects**: String(), Array(), Math(), Number(), Date(), JSON(), RegExp()
- **String object in JavaScript**: Of all JavaScript's objects, the String object is the most commonly used. It is used to work with text, and it works with a series of characters. Just like any JavaScript object, the String object also has properties and methods
    - **Examples of properties of the String object**:
        - **Length** (returns the length of the string)
        - **Prototype** (allows you to add properties and methods to an object)
    - **Examples of the String object methods**:
        - **String () method**: This method can be used to create new strings.

        - **Syntax for String object**: var variable_name = new String(string);
        - **Example**: The String () object is used to create a new string



```
var myString = new String();
myString = "Created string with New";
console.log(myString); // prints "Created
a string with New"
```

- **charAt() method**: It returns the character at the specified index.
- **indexOf() method**: It returns the index within the calling String object.
- **toLowerCase() method**: It returns the calling string value converted lower case.
- **toUpperCase() method**: It returns the calling string value converted to uppercase.
- **slice() method**: It extracts a session of a string and returns a new string.
- **split() method**: It splits a string object into an array of strings by separating the string into the substrings.

6.5 Mostly used built-in JavaScript object (array object)

  - **The Array Object**: JavaScript variables can be objects. Arrays in JavaScript are special kinds of objects. Because of this, you can have variables of different data types in the same array.
    - Just like any ordinary object in JavaScript, the array instance has a constructor function called Array().
- **Examples of the Array object properties:**
  - **length property**: returns the number of elements in an array
  - **prototype property**: Allows you to add properties and methods to an Array object
- **Examples of the Array object methods**:
  - **Array () method**: Just like the Date() and String() objects, Array() also uses the new method to create a new array
    - **Syntax for Array object**: var arr = new Array();
    - **Example**: The String () object is used to create a new string

```
var arr = new Array();
arr = [1, "Sam", 3, true];
console.log(arr); // prints [1, "Sam", 3,
true]
```

- **push() method**: Adds new elements to the end of an array, and returns the new array
- **pop() method**:: Removes an item from the end of an array
- **shift() method**:: Removes the first element of an array, and returns that element
- **unshift() method**:: Adds new elements to the beginning of an array, and returns the new array
- **isArray() method**:: Returns true if the argument is an array, or false otherwise.
- **filter() method**: Returns a new array containing all elements of the calling array for which the provided filtering function returns true.
- **toString() method**: Converts an array to a string, and returns the result
- **forEach() method**: Calls a function for each element in the array.
- **map() method**: Returns a new array containing the results of calling a function on every element in this array.
- **sort() method**: Sorts the elements of an array in place and returns the array.
- **splice() method**: Adds/Removes elements from an array

6.6 Mostly used built-in JavaScript object (Math object)

- **Math object:** It is used for performing advanced arithmetic and trigonometric functions, expanding on JavaScript's basic arithmetic operators (plus, minus, multiply, divide).
    - Unlike Array, Date and String objects, the Math object is static, so you don't need to create a new Math object in order to use it. To access the properties and method of the Math object, you merely specify the Math object, along with the method or property you wish. For example, to return the value of pi, you use:

```
var pi = Math.PI;
console.log(pi); // prints 3.141592653589793
```

        - **Examples of the Math properties:**
            - **PI (t**he numeric equivalent of PI: 3.14 etc.)
            - **SQRT2** (The square root of 2)
        - **Examples of the Math object methods**:
            - **floor() method**:: Returns the largest integer less than or equal to the argument number.
            - **min() method**:: Returns the lesser of two values
            - **max() method**:: Returns the greater of two values
            - **pow() method**:: Returns base to the exponent power, that is base exponent.
            - **random() method**:: Returns a random number
            - **round() method**:: Returns a number rounded to the nearest whole value
- **Refer this link to refer to all of JavaScript's built in. Methods**:
    - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects
- **Writing Algorithms using the built-in objects and their methods:** What's hiding amongst the crowd? A word is on the loose and now has tried to hide amongst a crowd of tall letters! Help write a function to detect what the word is, knowing the following rules.
    - **Rule 1:** The crowd word has all its letters in uppercase
    - **Rule 2**: The wanted word has all its letters in lowercase
    - **Rule 3:** Note that the word will be spread out amongst the random letters, but their letters remain in the same order
    - **Test case 1**: detectWord("UcUNFYGaFYFYGtNUH") prints "cat"
    - **Test case 2**: detectWord("bEEFGBuFBRrHgUHlNFYaYr") prints "burglar"
- **Let's solve the above problem using JavaScript built-in objects and their methods**.

```
function findHiddenWord(anyString) {
    var hiddenWordContainer = "";
    for (let i = 0; i < anyString.length; i++) {
    if (anyString[i] === anyString[i].toLowerCase())
{
    hiddenWordContainer = hiddenWordContainer +
anyString[i];
    }
    }
    return hiddenWordContainer;
}
console.log(findHiddenWord("UcUNFYGaFYFYGtNUH")); //
prints cat
```