object ----->property (function)
------>value (type of data)

JavaScript Objects

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Basics

Object literals – declaration and initialization

```
Basic syntax:-
var objectName = {
    member1Name: member1Value,
    member2Name: member2Value,
    member3Name: member3Value
};
```

Real example 1

var person = { }; // declare

```
Real example 2
var person = {
   name: ['Bob', 'Smith'],
   age: 32, gender: 'male',
   interests: ['music', 'skiing'],
   bio: function() {
       alert(this.name[0] + ' ' + this.name[1] +
       ' is ' + this.age + ' years old. He likes
       ' + this.interests[0] + ' and
       ' + this.interests[1] + '.');
    },
    greeting: function() {
       alert('Hi! I\'m ' + this.name[0]
          + '.');
    }
 };
```

Dot notation

- Object's properties and methods can be accessed using **dot notation**.
- The object name (person) acts as the **namespace** — it must be entered first to access anything **encapsulated** inside the object.

```
person.age
person.interests[1]
person.bio()
```

Dot notation – sub namespaces

It is even possible to make the value of an object member another object.

```
Original code:-
var person = { ..
   name: ['Bob', 'Smith'],
…
};
```

To access:-
person.name[0]
person.name[1]

```
Create sub-namespaces:-
var person = {
   name : {
      first: 'Bob',
      last: 'Smith'
   },
  …
  };
```

To access:-
person.name.first
person,.name.last

Bracket notation

- Another way to access object properties.

```
person.age                  →       person[‘age’]
person.name.first           →       person[‘name’][‘first’]
```

Setting object members

- Updating values using:-

Dot notation	Bracket notation
person.age 45;	person[‘name’][‘last] = ‘Cratchit’;

person.farewell = function() {
    alert(“Bye everybody”);
}
	person[‘eyes’] = ‘hazel’;

You’ve been using object all along!

- //Using a method available on an instance of the String class

```
myString.split(‘,’);
```

- //Using methods available in the Document class

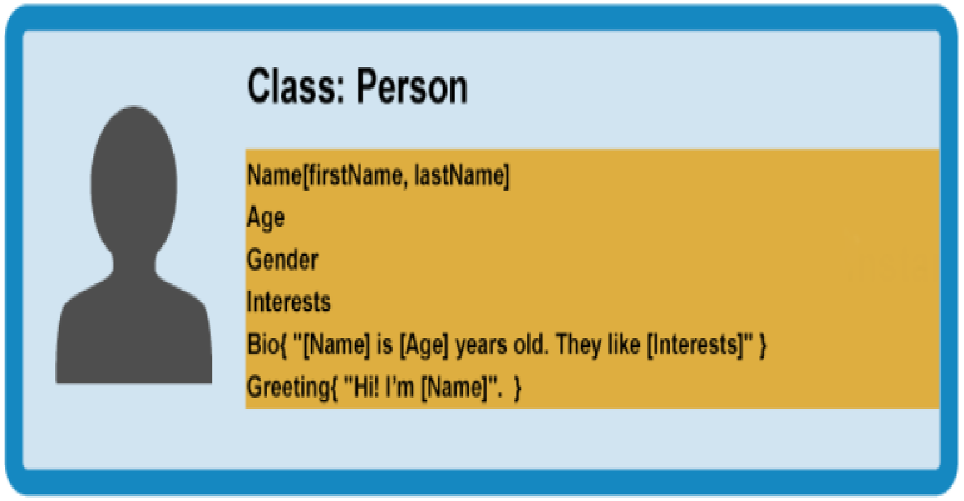
```
var myDiv = document.createElement(‘div’);
var myVideo = document.querySelector(‘video’);
```

Object-oriented JavaScript

Object-oriented

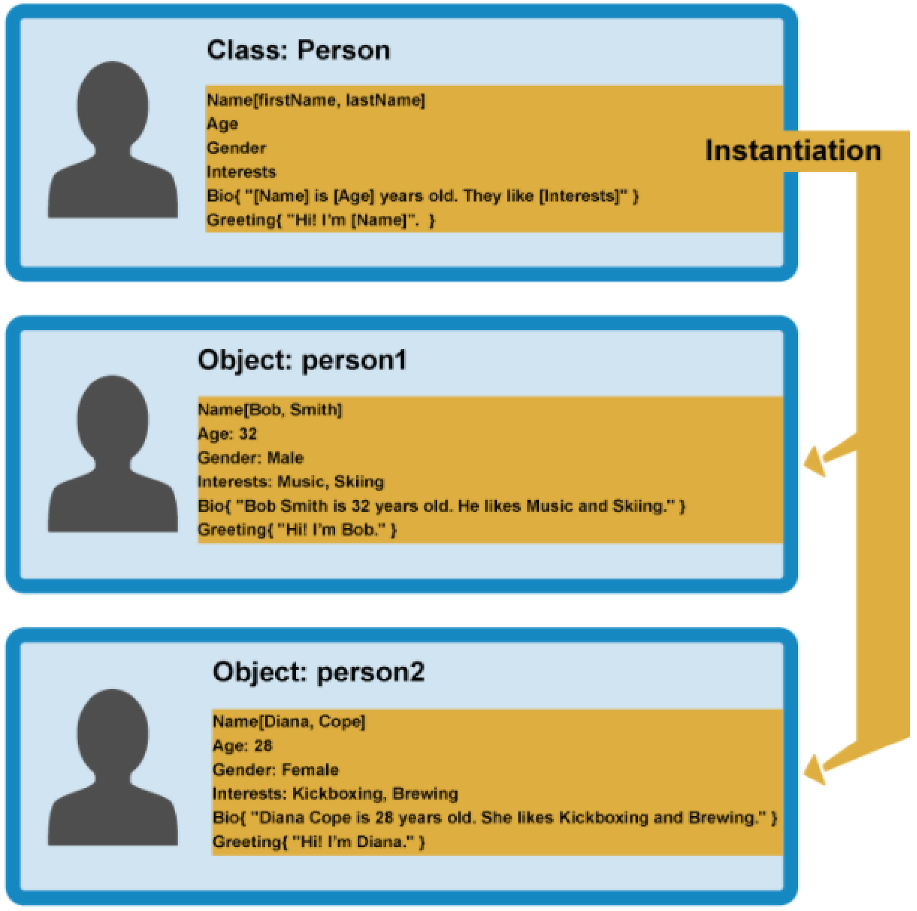
- OOP is that we use objects to model real world things that we want to represent inside our programs
  - provide a simple way to access functionality that would otherwise be hard or impossible to make use of.
- Objects can contain related data and code, which represent information about the thing you are trying to model, and functionality or behavior that you want it to have. Object data (and often, functions too) can be stored neatly (the official word is **encapsulated**) inside an object package (which can be given a specific name to refer to, which is sometimes called a **namespace**), making it easy to structure and access; objects are also commonly used as data stores that can be easily sent across the network.

Object template and instances



Class: Person

Name[firstName, lastName]
Age
Gender
Interests
Bio{ "[Name] is [Age] years old. They like [Interests]" }
Greeting{ "Hi! I'm [Name]". }



Class: Person

Name[firstName, lastName]
Age
Gender
Interests
Bio{ "[Name] is [Age] years old. They like [Interests]" }
Greeting{ "Hi! I'm [Name]". }

Object: person1

Name[Bob, Smith]
Age: 32
Gender: Male
Interests: Music, Skiing
Bio{ "Bob Smith is 32 years old. He likes Music and Skiing." }
Greeting{ "Hi! I'm Bob." }

Object: person2

Name[Diana, Cope]
Age: 28
Gender: Female
Interests: Kickboxing, Brewing
Bio{ "Diana Cope is 28 years old. She likes Kickboxing and Brewing." }
Greeting{ "Hi! I'm Diana." }

Constructor

```
Constructor
function createNewPerson(name) {
   var obj = {};
   obj.name = name;
   obj.greeting = function() {
   alert('Hi! I\'m ' + this.name + '.');
   };
return obj;
}
```

simplify

```
A simplified Constructor
function Person(name) {
 this.name = name;
 this.greeting = function() {
 alert('Hi! I\'m ' + this.name + '.');
 };
}
```

Object instances

```
Constructor
function Person(name) {
 this.name = name;
 this.greeting = function() {
 alert('Hi! I\'m ' + this.name + '.');
 };
}
```

```
Create instance
var person1 = new Person('Bob');
var person2 = new Person('Sarah');
```

Object prototypes

- JavaScript is often described as a **prototype-based language** — each object has a **prototype object**, which acts as a template object that it inherits methods and properties from
- Constructor:

```
function Person(first, last, age, gender, interests)  {
  // property and method definitions
  this.first = first;
  this.last = last;
  //...
}
```

- Creating instances

```
var person1 = new Person('Bob', 'Smith', 32, 'male', ['music', 'skiing']);
```

Thank you.