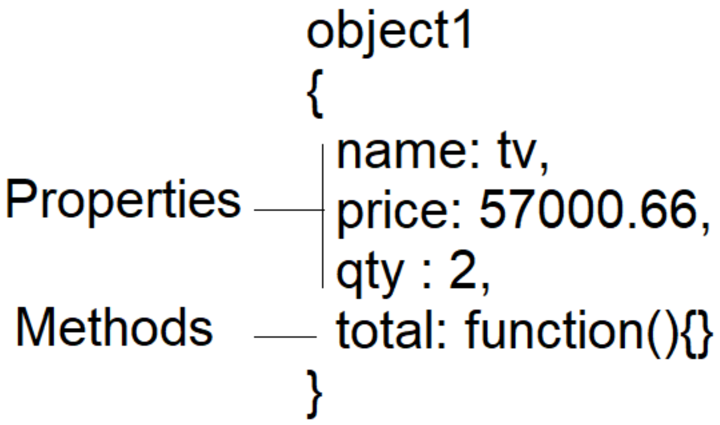
Object Type

- Object in computer programming was introduced in early 1960's by "**Alan Kay**".
- Object can keep all related data and functionality at one memory reference.
- Object comprises of data and functionality.
- Object is a set of properties and methods.



```
object1
{
  name: tv,
  price: 57000.66,
  qty : 2,
  total: function(){}
}
```

```
object2
{
  name:"mobile"
  price: 12000,
}
```

- Data is stored in properties.
- Functionality is defined in methods.
- In JavaScript earlier version object is also known as "pseudo class".

Syntax:

```
var object = {
   property: value,
   method: function(){}
}
```

- You can access object property within the object by using **“this” keyword.**
- You can access object property **outside the object by using object name.**

  Syntax:

  ```
  object
  {
     this.property
     this.method()
  }
  object.property
  object.method()
  ```

- Later in early 1967 “**Johan Olay, Kristian Nygaard**” introduced the concept of reusing object with class. [OOP]
- The first OOP language was **SIMULA 67**, Small Talk, C++, Java, .NET Languages

Ex:

```
<script>
   function f1(){
     var product = {
        Name: "",
        Price: 0,
        Qty: 0,
        Total: function(){
           return this.Qty * this.Price;
        },
        Print: function(){
           document.write(`
            Name : ${this.Name} <br>
            Price: ${this.Price}<br>
            Qty: ${this.Qty}<br>
            Total: ${this.Total()}
           <br>`);
        }
     }
```

```
        product.Name = "Samsung TV";
        product.Price = 4000.44;
        product.Qty = 2;
        product.Print();
        document.write("<hr>");
        product.Name = "Nike Casuals";
        product.Price = 2000.44;
        product.Qty = 3;
        product.Print();

    }
    f1();
</script>
```

JSON Type Data

[JavaScript Object Notation]

- It is a format for data.
- **It is a collection objects.**

Ex:

```
<script>
   function f1(){
      var products = [
         {Name: "TV", Price: 45000.44,
Cities:['Delhi', 'Hyd']},
         {Name: "Mobile", Price: 12000.33,
Cities: ['Hyd','Chennai']}
      ];
      for(var product of products) {
         document.write(product.Name + "-" +
product.Price + "-" + product.Cities.toString()
+ "<br>");
      }
   }
   f1();
</script>
```

Ex: Adding Rows into Table Dynamically

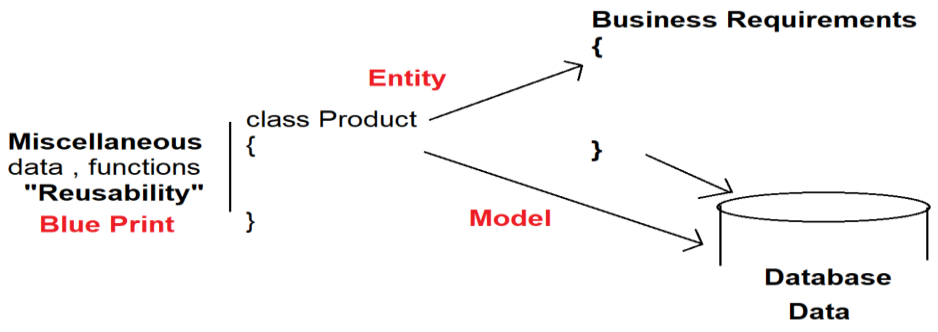
```
<!DOCTYPE html>
<html>
   <head>
      <title>Dynamic Table</title>
      <link rel="stylesheet"
href="../node_modules/bootstrap/dist/css/bootst
rap.css">
      <script>
          var products = [
            {Name: "JBL Speaker", Price: 4500.55,
Photo: "../Images/jblspeaker.jpg"},
            {Name: "Earpods", Price: 2500.55, Photo:
"../Images/earpods.jpg"},
            {Name: "Nike Casuals", Price: 6500.55,
Photo: "../Images/shoe.jpg"},
            {Name: "Lee Boot", Price: 1500.55,
Photo: "../Images/shoe1.jpg"},
         ];
         function bodyload(){
            var tbody=
document.getElementById("tbody");
            for(var item of products)
            {
               var tr = document.createElement("tr");
               var tdName =
document.createElement("td");
               var tdPrice =
document.createElement("td");
               var tdPhoto =
document.createElement("td");

               tdName.innerHTML = item.Name;
               tdPrice.innerHTML = item.Price;

               var pic = new Image();
               pic.src= item.Photo;
               pic.height="50";
               pic.width="50";
```

```
                tdPhoto.appendChild(pic);
                tr.appendChild(tdName);
                tr.appendChild(tdPrice);
                tr.appendChild(tdPhoto);

                tbody.appendChild(tr);
              }
            }
        </script>
    </head>
    <body onload="bodyload()" class="container-
fluid">
        <h2>Product Details</h2>
        <table class="table table-hover">
            <thead>
              <tr>
                <th>Name</th>
                <th>Price</th>
                <th>Preview</th>
              </tr>
            </thead>
            <tbody id="tbody">

            </tbody>
        </table>
    </body>
  </html>
```

OOP in JavaScript

Class in OOP:

- Class is a **program template**. It comprises of **sample data and functionality**, which you can implement in any application and customize according to your requirements.
- Code Reusability
- JavaScript supports only certain characteristics and features of OOP.
- JavaScript introduced class-based OOP from ES5.
- Class is **a logical entity** when it is mapping to business requirements.
- Class is **a model** when it is mapping to data requirements.
- Class is **a blue print** when it is defined with miscellaneous data logic for reusability.



-Class is a **program template with data and logic**.

-**Data is defined in properties and logic in methods**.
-Classes a configured by using **“class” keyword**.

Syntax:

```
class ClassName
{
  //members;
}
```

- A typical JavaScript class comprises of
  - Properties
  - Methods
  - Constructor

Properties:

- Properties a mutable.
- Properties contain data that can change according to state and situation.

- Class in JavaScript can be configured by using “class” keyword and 2 different techniques
  - Class Declaration
  - Class Expression

Ex:

```
<script>
   // Class Declaration
   class Category
   {


   }
   // Class Expression
   var Product = class
   {


   }
</script>
```

-Properties are accessible with in the class by using "**this" keyword** and outside the class by using an **instance of class**.

-Class can't contain variable declarations, hence data is stored only in properties.

FAQ: Can we define a variable in class?

No. Variable can be defined in Module scope not in class scope.

FAQ: Can we define a function in class?

No. Function can be defined in module scope, not in class scope.

FAQ: How functionality is defined in class?

Functionality is configured by using methods.

FAQ: Why a variable is not allowed in class?

- Class requires to modify its functionality and data according to state and situation.
- Hence variable is not allowed.

-**Functions** always are intended to return a value.

-**Methods** always are intended not to return a value, just define a functionality.

-**Procedure** may or may not return value, it changes its behaviour according to state and situation.

-**In JavaScript method and function both have the behaviour of "Procedure".**

-JavaScript class will not allow functions, only methods.

“In JavaScript outside class you will have functions and within the class you will have methods”

Ex:

```
<script>
   class Product
   {
     Name = "";
     Price = 0;
     InStock = true;
     Qty = 0;
     Total(){
        return this.Qty * this.Price;
     }
     Print(){

document.write(`Name=${this.Name}<br>Price=${this.Price}
<br>Qty=${this.Qty}<br>Total=${this.Total()}`);
     }
```

```
    }
    var tv = new Product;
    tv.Name = "Samsung TV";
    tv.Price = 34000.44;
    tv.Qty = 2;
    tv.Print();
</script>
```