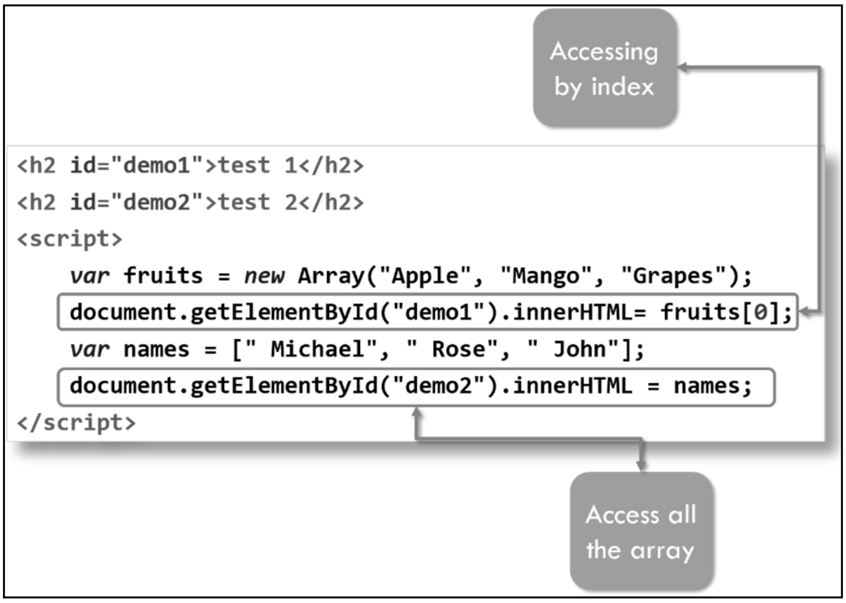
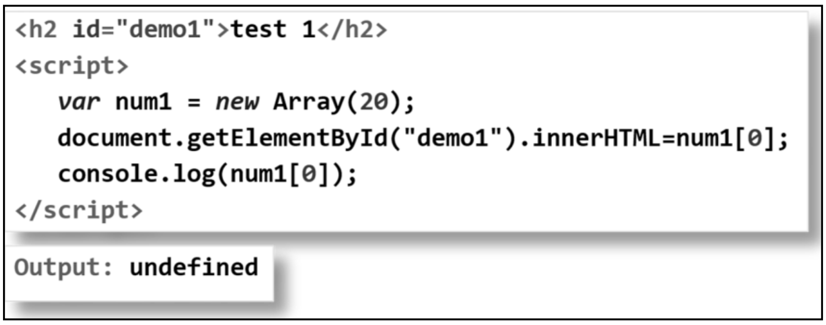
Array

- One of the most used data structures is the array. An array is a list of things. It can be values, objects, and lists of lists.
- There are two (2) ways to initialize an array. First, initialize an array using the array method. Second, use the square bracket notation.
    - `var x = new Array("Hello", "World");`
    - `var x = ["Hello","World"];`
- To access an array, you can set the index number or access all the array.



```
<h2 id="demo1">test 1</h2>
<h2 id="demo2">test 2</h2>
<script>
    var fruits = new Array("Apple", "Mango", "Grapes");
    document.getElementById("demo1").innerHTML= fruits[0];
    var names = [" Michael", " Rose", " John"];
    document.getElementById("demo2").innerHTML = names;
</script>
```

- When using the array method, you sometimes need to avoid only one (1) integer (e.g., **`array(2)`**). It may result to an output of "undefined". Still, it is better to use the open and close bracket **`[]`** for storing data in an array.
    - Example:

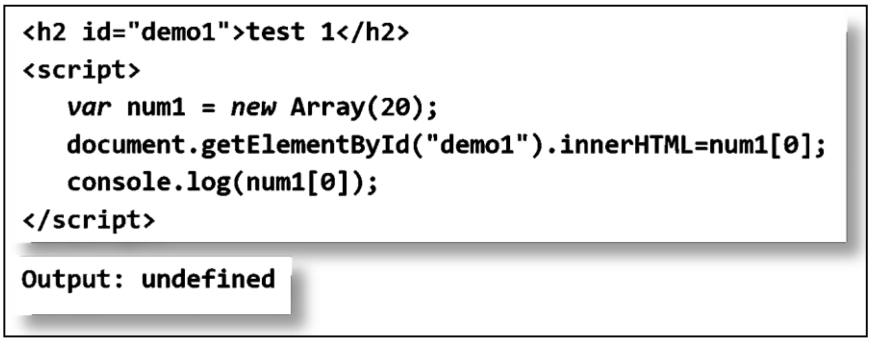


```
<h2 id="demo1">test 1</h2>
<script>
    var num1 = new Array(20);
    document.getElementById("demo1").innerHTML=num1[0];
    console.log(num1[0]);
</script>
```

```
Output: undefined
```

Array Length

- The array length is used to get the total number of an array.

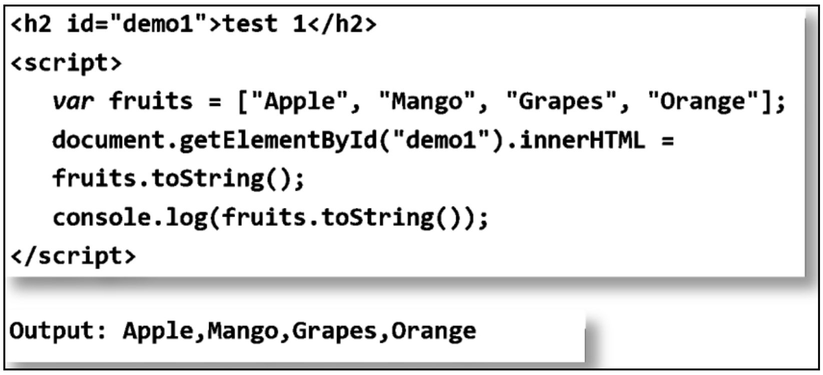


```
<h2 id="demo1">test 1</h2>
<script>
    var num1 = new Array(20);
    document.getElementById("demo1").innerHTML=num1[0];
    console.log(num1[0]);
</script>
```

```
Output: undefined
```

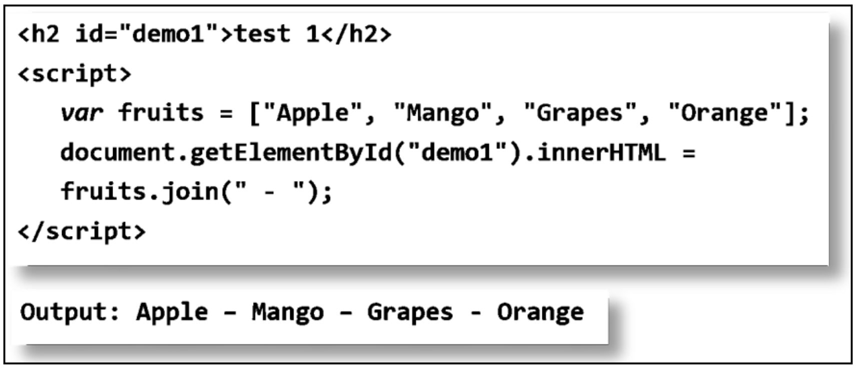
Array Methods

- **`toString()`**
    - This method is used to convert the array into a string of array values with a separated comma.



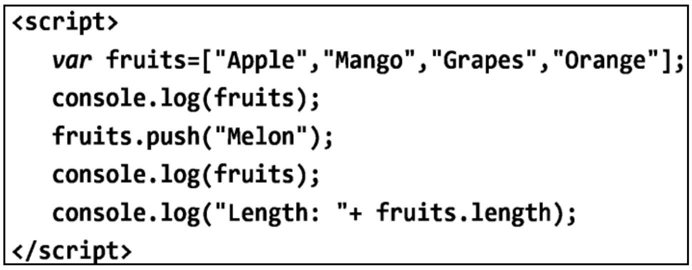
```
<h2 id="demo1">test 1</h2>
<script>
   var fruits = ["Apple", "Mango", "Grapes", "Orange"];
   document.getElementById("demo1").innerHTML =
   fruits.toString();
   console.log(fruits.toString());
</script>

Output: Apple,Mango,Grapes,Orange
```

- `join()`
    - The **`join()`** array method combines all array elements into a string.

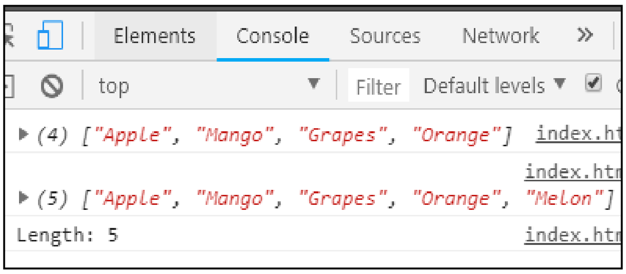


```
<h2 id="demo1">test 1</h2>
<script>
   var fruits = ["Apple", "Mango", "Grapes", "Orange"];
   document.getElementById("demo1").innerHTML =
   fruits.join(" - ");
</script>

Output: Apple - Mango - Grapes - Orange
```

- `push()`
    - The **`push()`** method is used to add another data to the end of an array and returns the new length of an array.

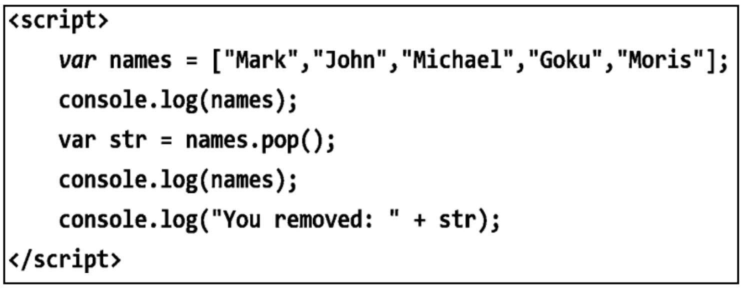


```
<script>
   var fruits=["Apple","Mango","Grapes","Orange"];
   console.log(fruits);
   fruits.push("Melon");
   console.log(fruits);
   console.log("Length: "+ fruits.length);
</script>
```

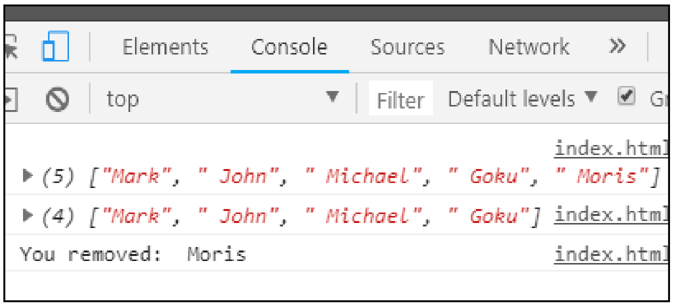


```
▶ (4) ["Apple", "Mango", "Grapes", "Orange"]  index.ht
                                               index.htr
▶ (5) ["Apple", "Mango", "Grapes", "Orange", "Melon"]
Length: 5                                      index.htr
```

- `pop()`
    - The **`pop()`** array method removes the last element of an array.



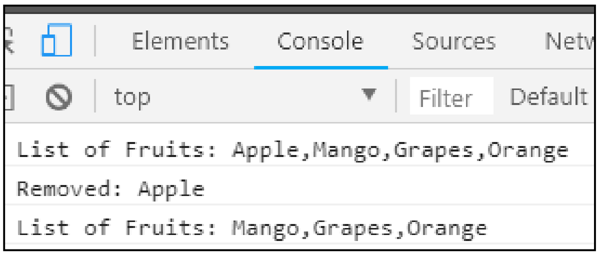
```
<script>
   var names = ["Mark","John","Michael","Goku","Moris"];
   console.log(names);
   var str = names.pop();
   console.log(names);
   console.log("You removed: " + str);
</script>
```



```
                                                 index.html
▶ (5) ["Mark", " John", " Michael", " Goku", " Moris"]
▶ (4) ["Mark", " John", " Michael", " Goku"]  index.html
You removed:  Moris                            index.html
```

- `shift()`
    - This **`shift()`** array method is the same as the **`pop()`** array method. Both of them remove an array element or data. The difference between these two (2) array methods is that the **`pop()`** method removes the last element, while the **`shift()`** method removes the first element.
    - This method returns the remove string array element.

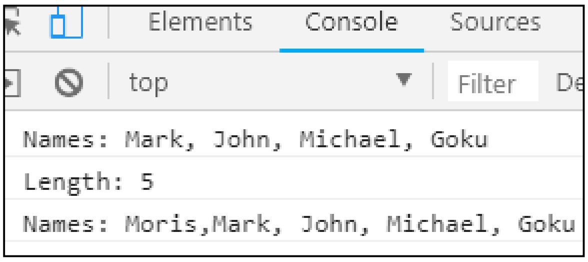
```
<script>
    var fruits = ["Apple","Mango","Grapes","Orange"];
    console.log("List of Fruits: "+fruits);
    console.log("Removed: "+ fruits.shift());
    console.log("List of Fruits: "+fruits);
</script>
```



```
List of Fruits: Apple,Mango,Grapes,Orange
Removed: Apple
List of Fruits: Mango,Grapes,Orange
```

- `unshift()`
  - This **`unshift()`** array method is used to add new data at the beginning.
  - This method also returns the new array length.

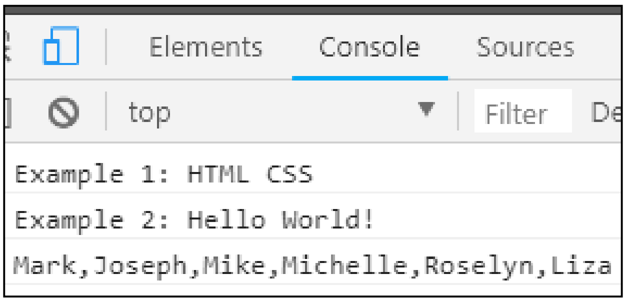
```
<script>
    var names = ["Mark"," John"," Michael"," Goku"];
    console.log("Names: " + names);
    var str = names.unshift("Moris");
    console.log("Length: " + str);
    console.log("Names: " + names);
</script>
```



```
Names: Mark, John, Michael, Goku
Length: 5
Names: Moris,Mark, John, Michael, Goku
```

- `concat()`
  - Concatenation is commonly known for combining two (2) or more strings.

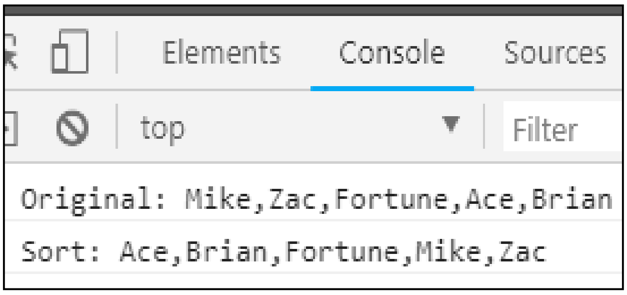
```
<script>
    var str1 ="Example 1: " + "HTML" + " CSS";
    console.log(str1);
    var str2 ="Example 2: ".concat("Hello ","World!");
    console.log(str2);
    var bName = ["Mark","Joseph","Mike"];
    var fName = ["Michelle","Roselyn","Liza"];
    console.log(bName.concat(fName).toString());
</script>
```



```
Example 1: HTML CSS
Example 2: Hello World!
Mark,Joseph,Mike,Michelle,Roselyn,Liza
```

- `sort()`
  - The **`sort()`** method is used to sort the array elements alphabetically.

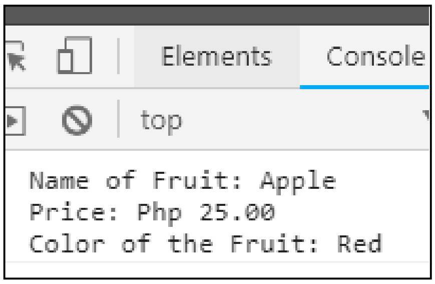
```
<button onclick="srt()">Click</button>
<script>
    var names = ["Mike","Zac","Fortune","Ace","Brian"];
    console.log("Original: " + names);
    function srt(){
      names.sort();
      console.log("Sort: " + names);
    }
</script>
```



```
Original: Mike,Zac,Fortune,Ace,Brian
Sort: Ace,Brian,Fortune,Mike,Zac
```

- `reverse()`
  - The **`reverse()`** method is used to reverse an array's elements.
  - This can be used in a sort method to reverse the array elements.

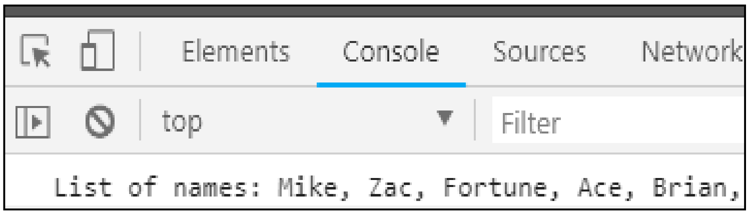
```
<script>
var fruit = {Name:"Apple", Price:
"Php 25.00", Color:"Red"};

var item ="Name of Fruit: "+
fruit.Name + "\nPrice: " +
fruit.Price + "\nColor of the Fruit:
" + fruit.Color;

console.log(item);
</script>
```



```
Name of Fruit: Apple
Price: Php 25.00
Color of the Fruit: Red
```

- `foreach()`
    - The foreach method is used to process the elements in an array.
    - The function of the foreach uses three (3) arguments:
        - **Value** – the value of the current element (required)
        - **Index** – the array index of the current element (optional)
        - **Array** – the array object the current element belongs to (optional)

```
<script>
    var txt = "List of names: ";
    var arrNames = ["Mike","Zac","Fortune","Ace","Brian"];
    arrNames.forEach(strNames);
    function strNames(value, index, array){
        txt = txt + value + ", ";
    }
    console.log(txt);
</script>
```



```
List of names: Mike, Zac, Fortune, Ace, Brian,
```

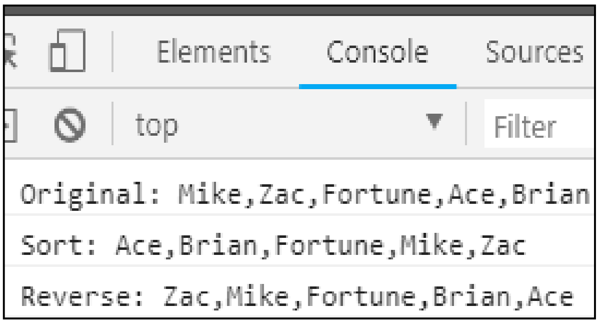
Object

An object can be a Boolean, Integer, String, Array, Regular Expressions, Functions, Math, and Dates. In JavaScript, you can create an object without creating a class. An object also has properties and methods.

Object Property

- An object property is the attribute of an object. To define a property, you need to assign a value.
- There are two (2) ways to access a property:
    - Dot notation
    - Square bracket notation.

```
<script>
    var names = ["Mike","Zac","Fortune","Ace","Brian"];
    console.log("Original: "+names);
    names.sort();
    console.log("Sort: " + names);
    names.reverse();
    console.log("Reverse: " + names);
</script>
```

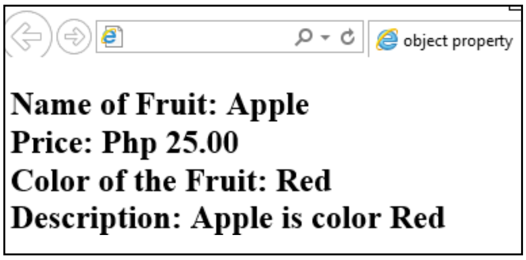


```
Original: Mike,Zac,Fortune,Ace,Brian
Sort: Ace,Brian,Fortune,Mike,Zac
Reverse: Zac,Mike,Fortune,Brian,Ace
```

Object Method

- A method refers to a function that is associated with an object, in which an object contains a syntax to perform a specific task.

```
<h2 id="frt">
</h2>
<script>
    var fruit = {
        Name : "Apple",
        Price : "Php 25.00",
        Color : "Red",
        Description: function(){
            return this.Name + " is color " + this.Color;
        }
    };
var item ="Name of Fruit: "+ fruit.Name +
        "<br> \nPrice: " + fruit.Price +
        "<br> \nColor of the Fruit: " + fruit.Color +
        "<br> \nDescription: " + fruit.Description();
    document.getElementById("frt").innerHTML = item;
    console.log(item);
</script>
```



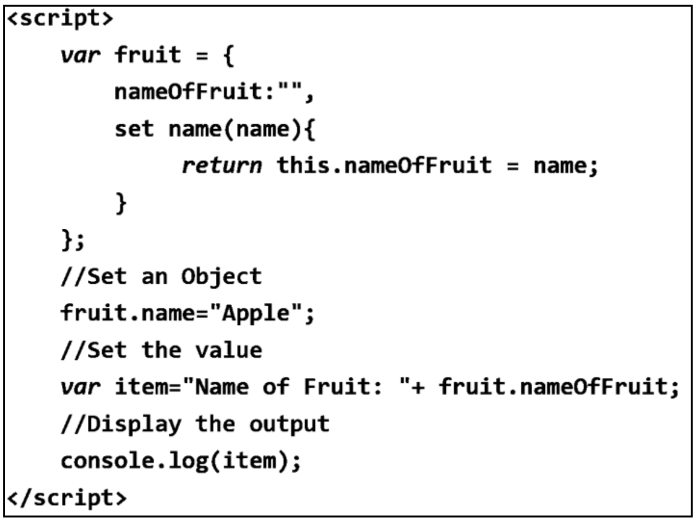
Name of Fruit: Apple
Price: Php 25.00
Color of the Fruit: Red
Description: Apple is color Red

Accessors

- **Getter**
  - Getter or get method gets the value of a property.

```
<script>
    var fruit = {
        nameOfFruit:"Apple",
        get name(){
            return this.nameOfFruit;
        }
    };
    var item ="Name of Fruit: "+ fruit.name;
    console.log(item);
</script>
```

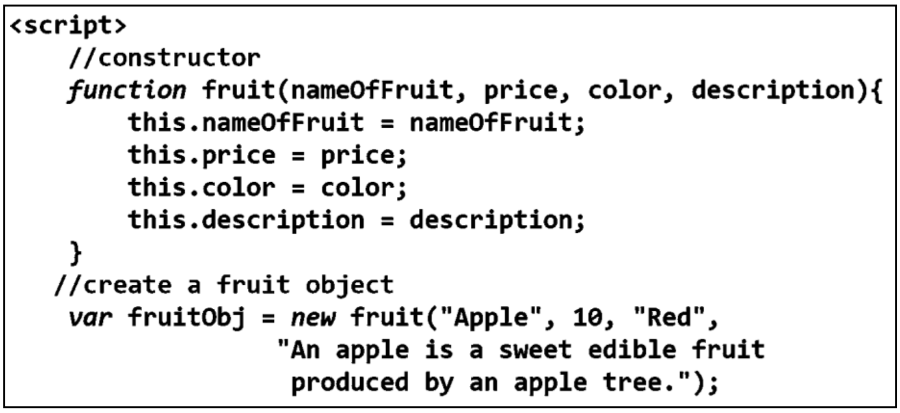
- **Setter**
  - The setter or set method sets the value of a specific property.



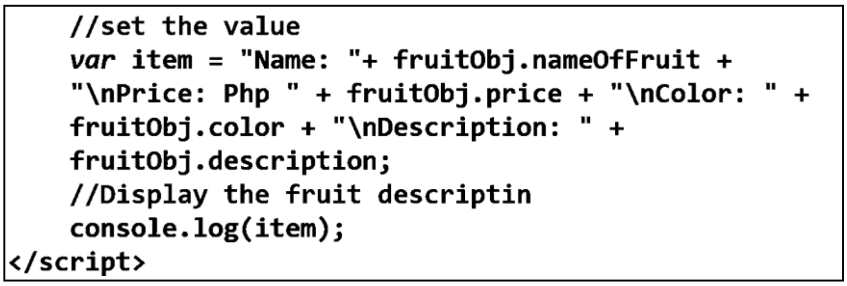
```
<script>
    var fruit = {
        nameOfFruit:"",
        set name(name){
             return this.nameOfFruit = name;
        }
    };
    //Set an Object
    fruit.name="Apple";
    //Set the value
    var item="Name of Fruit: "+ fruit.nameOfFruit;
    //Display the output
    console.log(item);
</script>
```

Constructor

- A constructor is a method that contains two (2) or more parameters, and it is used for creating and initializing an object inside the class.



```
<script>
    //constructor
    function fruit(nameOfFruit, price, color, description){
        this.nameOfFruit = nameOfFruit;
        this.price = price;
        this.color = color;
        this.description = description;
    }
   //create a fruit object
    var fruitObj = new fruit("Apple", 10, "Red",
                  "An apple is a sweet edible fruit
                   produced by an apple tree.");
```



```
    //set the value
    var item = "Name: "+ fruitObj.nameOfFruit +
    "\nPrice: Php " + fruitObj.price + "\nColor: " +
    fruitObj.color + "\nDescription: " +
    fruitObj.description;
    //Display the fruit descriptin
    console.log(item);
</script>
```

REFERENCES:

1. Connolly, R. & Hoar, R. (2015). *Fundamentals of web development*. New Jersey: Pearson Education, Inc.
2. Lemay, L., Colburn, R., & Kyrnin, J. (2016). *Sams teach yourself html, CSS and JavaScript web publishing in one hour a day* (7th Ed.). New Jersey: Pearson Education, Inc.
3. Kraus, J. (2016). *Introducing web development*. California: Apress Media, LLC.