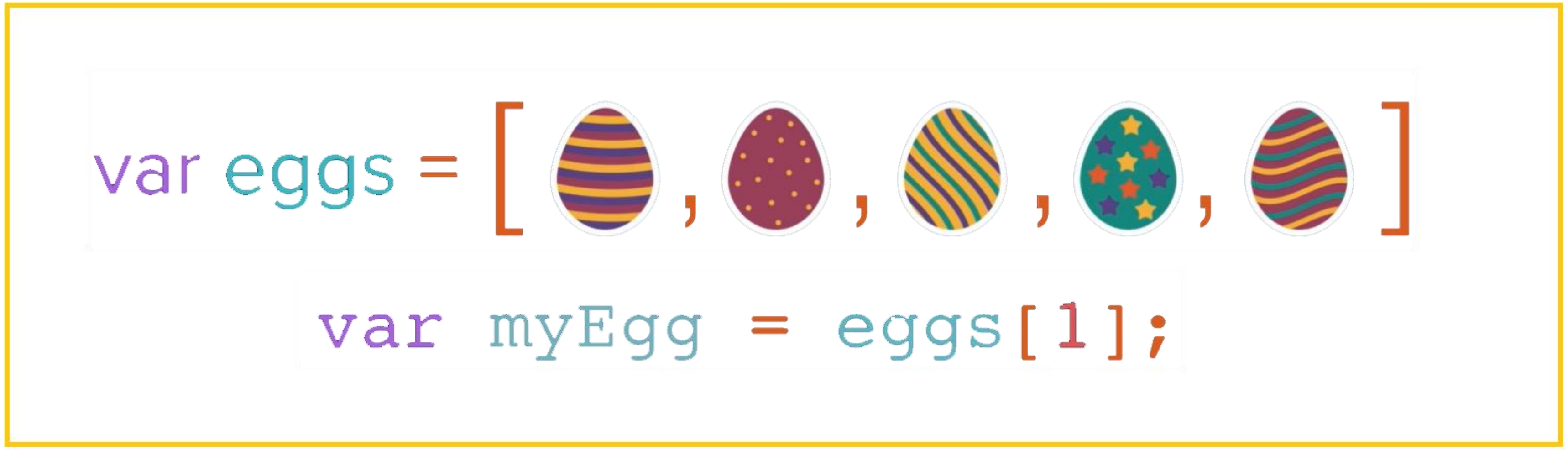
JAVASCRIPT

Prepared By: Rubab Javaid

JavaScript Arrays

- Array is collection of items, that are related and these items can be stored together into the same container.



```
var myEgg = eggs[1];
```

JavaScript Arrays

- An array can hold many values under a single name, and you can access the values by referring to an index number.
- JavaScript arrays are used to store multiple values in a single variable.
- **Syntax:**
  - var *array_name* = [*item1*, *item2*, ...];
- **Example**
  - var cars = ["Saab", "Volvo", "BMW"];

Access the Elements of an Array

- You access an array element by referring to the **index number.**
  - var cars = ["Saab", "Volvo", "BMW"];
    document.getElementById("demo").innerHTML = cars[0];

Changing an Array Element

- This statement changes the value of the first element in cars:

  -
    ```
    var cars = ["Saab", "Volvo", "BMW"];
    cars[0] = "Opel";
    document.getElementById("demo").innerHTML = cars[0];
    ```

Arrays are Objects

- Arrays are a special type of objects. The typeof operator in JavaScript returns "object" for arrays.
- But, JavaScript arrays are best described as arrays.
- **Array:**
  - Arrays use **numbers** to access its "elements". In this example, person[0] returns John:
    - var person = ["John", "Doe", 46];
- **Object:**
  - Objects use **names** to access its "members". In this example, person.firstName returns John:

- var person = {firstName:"John", lastName:"Doe", age:46};

Array Elements Can Be Objects

- JavaScript variables can be objects. Arrays are special kinds of objects.
- Because of this, you can have variables of different types in the same Array.
- You can have objects in an Array. You can have functions in an Array. You can have arrays in an Array:
  - myArray[0] = Date.now;
    myArray[1] = myFunction;
    myArray[2] = myCars;
- **Array Properties and Methods**
- The real strength of JavaScript arrays are the built-in array properties and methods:
- var x = cars.length; // The length property returns the number of elements
  var y = cars.sort(); // The sort() method sorts arrays

The length Property

- The length property of an array returns the length of an array (the number of array elements).
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
    fruits.length;   // the length of fruits is 4

Array Accessing

- **Accessing the First Array Element**
  - fruits = ["Banana", "Orange", "Apple", "Mango"];
    var first = fruits[0];
- **Accessing the Last Array Element**
  - fruits = ["Banana", "Orange", "Apple", "Mango"];
    var last = fruits[fruits.length - 1];

Looping Array Elements

- The safest way to loop through an array, is using a for loop:

  -
    ```
    var fruits, text, fLen, i;
    fruits = ["Banana", "Orange", "Apple", "Mango"];
    fLen = fruits.length;

    text = "<ul>";
    for (i = 0; i < fLen; i++) {
      text += "<li>" + fruits[i] + "</li>";
    }
    text += "</ul>";
    ```

Adding Array Elements

- The easiest way to add a new element to an array is using the push() method:
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
    fruits.push("Lemon");   // adds a new element (Lemon) to fruits
- New element can also be added to an array using the length property:
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
    fruits[fruits.length] = "Lemon";   // adds a new element (Lemon) to fruits

Adding Array Elements

- Adding elements with high indexes can create undefined "holes" in an array:
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
    fruits[6] = "Lemon";   // adds a new element (Lemon) to fruits

Converting Arrays to Strings

- The JavaScript method toString() converts an array to a string of (comma separated) array values.
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
    document.getElementById("demo").innerHTML = fruits.toString();
- **Result:**
  - Banana,Orange,Apple,Mango

Converting arrays to Strings

- The join() method also joins all array elements into a string.
- It behaves just like toString(), but in addition you can specify the separator:
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
    document.getElementById("demo").innerHTML = fruits.join(" * ");
- **Result:**
  - Banana * Orange * Apple * Mango

Popping and Pushing

- When you work with arrays, it is easy to remove elements and add new elements.
- This is what popping and pushing is:
- Popping items **out** of an array, or pushing items **into** an array.

Popping

- The pop() method removes the last element from an array:
- **Example**
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
    fruits.pop();   // Removes the last element ("Mango") from fruits
- The pop() method returns the value that was "popped out":
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
    var x = fruits.pop();   // the value of x is "Mango"

Pushing

- The push() method adds a new element to an array (at the end):
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
    fruits.push("Kiwi");   // Adds a new element ("Kiwi") to fruits
- The push() method returns the new array length:
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
    var x = fruits.push("Kiwi");   // the value of x is 5

Shifting Elements

- Shifting is equivalent to popping, working on the first element instead of the last.
- The shift() method removes the first array element and "shifts" all other elements to a lower index.

  -
    ```
    var fruits = ["Banana", "Orange", "Apple", "Mango"];
    fruits.shift();            // Removes the first element "Banana" from fruits
    ```
- The shift() method returns the string that was "shifted out":

  -
    ```
    var fruits = ["Banana", "Orange", "Apple", "Mango"];
    var x = fruits.shift();    // the value of x is "Banana"
    ```

Shifting Element

- The unshift() method adds a new element to an array (at the beginning), and "unshifts" older elements:
  - var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.unshift("Lemon");   // Adds a new element "Lemon" to fruits
- The unshift() method returns the new array length.
  - var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.unshift("Lemon");   // Returns 5

Changing Elements

- Array elements are accessed using their **index number:**
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
    fruits[0] = "Kiwi";   // Changes the first element of fruits to "Kiwi“
- The length property provides an easy way to append a new element to an array:
  - var fruits = ["Banana", "Orange", "Apple", "Mango"];
    fruits[fruits.length] = "Kiwi";   // Appends "Kiwi" to fruits

JavaScript dates

```
var d = new Date();
document.getElementById("demo").innerHTML = d;
```

- Years
- Months
- Days
- Hours
- Seconds
- Milliseconds

JavaScript Get Date Methods

Method	Description
getFullYear()	Get the **year** as a four digit number (yyyy)
getMonth()	Get the **month** as a number (0-11)
getDate()	Get the **day** as a number (1-31)
getHours()	Get the **hour** (0-23)
getMinutes()	Get the **minute** (0-59)
getSeconds()	Get the **second** (0-59)
getMilliseconds()	Get the **millisecond** (0-999)
getTime()	Get the time (milliseconds since January 1, 1970)
getDay()	Get the weekday as a number (0-6)

Set Date Methods

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

```
<script>
var d = new Date();
d.setFullYear(2020);
document.getElementById("demo").innerHTML = d;
</script>
```