DATA STRUCTURES

By: FUNCTION----Group 3

Arrays and Objects

Collections of Data

Table of contents.

Arrays in JS

An Array is used to store an ordered collection of data. It is better and convenient way of storing the data of the same type.

For example:

- A list of songs in a playlist.
- A collection of game levels on a console.
- A list of comments on medium.

Creating Arrays -- Using Array literal syntax

```
// To make an empty array

let favBooks = [  ];

// To make an array of strings

let colors = [ “red”, “green”, “yellow”, “blue”];

// To make an array of Numbers

let even = [ 2, 4, 6, 8];
```

Reference Types --- Data structures.

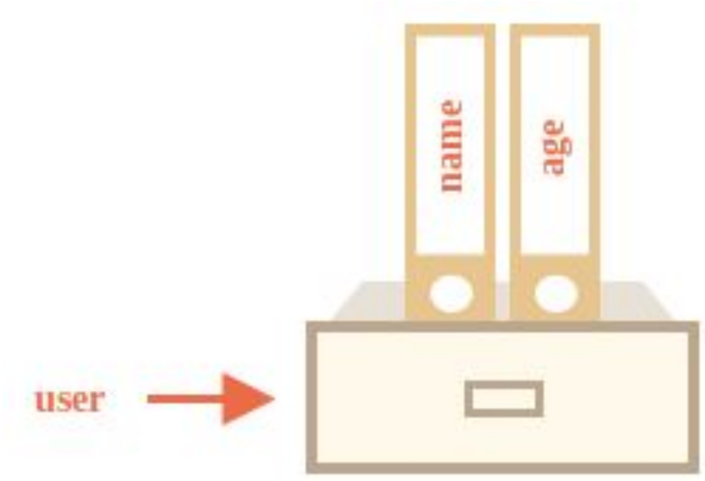
Note:

- **Primitive** types are stored as the actual value in the Variable.
- In Arrays and Objects, the variable stores a **reference** to where that array is in memory.
- The reference can be related to a <u>unique id</u> in memory.

Objects In JS

Objects are a collection of unordered but related properties.
Usually with Key and value pairs for each property.

- Rather than accessing data using index, we use the custom keys.



Creating Objects --- using object literal syntax

```
// To make an object, we use the curly braces.

var car = {

        name: “benz”,

        color: “white”,

        model: 2020

}
```

Key : value

ARRAY METHODS

- Array. push()
- Array.unshift()
- Array. Pop()
- Array. Shift()
- Array. Splice()

Array.push()

An array.push() adds an element at the end of an Array and returns the new length.

For example:

var selectedStudents = [‘‘*Jane”*, *“Jancita”*, *“Molly”]*

selectedStudents.push(“Mike"); **/* the new length is 4 */**

>>>> now selectedStudents = [“*Jane”*, *“Jancita”*, *“Molly”*, *“Mike”]*

Note:

The new item(s) will be added at the end of the array.

This method changes the length of the array.

Array.unshift()

This method adds a new element to an array (at the beginning).

For example:

var selectedStudents = [*“Jane”, “Jancita”, “Molly”]*

selectedStudents. unshift(“Mike");

>>> now selectedStudents = [“Mike”, “*“Jane”, “Jancita”, “Molly”]*

Array.pop()

This array method removes the last element from an array

For example:

var selectedStudents = [‘‘*Jane”*, “*Jancita”*, “*Molly”]*

Selected students.pop();

>>> now selectedStudents = [‘‘*Jane”*, “*Jancita”]*

Note:

Removes the last element “molly”

Array.shift()

This array method removes the first array element and "shifts" all other elements to a lower index.

For example:

var selected students = [‘‘*Jane*”, “*Jancita*”, “*Molly*”*]*

selected students.Shift();

>>> now selectedStudents = [“*Jancita*”, “*Molly*”*]*

Note:

Shifting is equivalent to popping, working on the first element instead of the last

CONDITIONS WITH ARRAYS

CONDITIONS WITH ARRAYS

DECLARATION OF AN ARRAY.

There are two ways of declaring an array;

1- var car = [];

2- var car = new array();

INITIATION OF AN ARRAY.

1- var car = [“Benz”,”BMW”,”Ford”]

ACCESSING AN ARRAY

2- Var car = new array(“Benz”,”Ford”,”BMW”);

ACCESSING AN ARRAY.

Var car =[“benz”,”ford”,”bmw”];

car[0] // benz

car [1] //ford

car[2] //bmw

CONDITIONAL STATEMENTS

Conditional statements help as check for a specific condition if met the code below is executed.

Example;

```
if(condition){

//code goes here

}
```

For many conditions we use

CONDITIONAL STATEMENT

For many conditions we use;

```
if(condition){

//code goes here

}else if(second condition){

//code goes here

}else {

}
```

CONDITIONAL WITH ARRAYS

OR USE A SWITCH CONDITION.

To access arrays with conditions we need to loop through the array loops like; for each(), for() loops;

Example next slide

CONDITIONAL WITH ARRAYS

```
Var car [“benz”, “ford”, “bmw”]

for(let i=0; i<car.length; i++){

Console.log(car[i]);

}

Or

Car.forEach(function(ev){

Console.log(ev);

})
```

CONDITIONAL WITH ARRAYS

```
Var car = [“benz”, “ford”, “bmw”];

for(let i=0; i<car.length; i++){if ( car[0] == “benz”){

console.log(“the first car is a benz”)

}else if(car[1] == “ford”){

console.log(“the second car is a ford”)

}else{

console.log(“the last car in the array is a bmw”)

}

}
```

Loops in Arrays

LOOPS:

These are variables that are used to repeatedly run a block of code. Loops are an easy way to do something over and over again.

There are different kinds of loops and these are:

For loop: loops through a block of code a number of times.

For/in loop: Loops through properties of an object:

While loop: Loops through a block of code until a specific condition is true.

Do/while loop: Loops through a block of code while a specified condition is true.

For/of loop: Loops through the value of an iterable object.