




**Imagine You Are Out for a Walk.  
What Might You See?**

# Things That Might Be Visible


- A car \_\_\_\_\_ 


- Traffic lights \_\_\_\_\_ 

- Your neighbour \_\_\_\_\_ 

- A pet dog \_\_\_\_\_ 

- Traffic police \_\_\_\_\_ 

- Advertisement hoardings \_\_\_\_\_ 

- A residential building \_\_\_\_\_ 

- An office building \_\_\_\_\_ 

- A cafe \_\_\_\_\_ 

- A bicycle \_\_\_\_\_ 

- An ATM booth \_\_\_\_\_ 

# What Is an Object?

For e.g., for a car you may say  
It has attributes like color, make, model, registration number, fuel capacity etc.  
It also has behaviour like start, pause (break), stop  
For e.g., for a pet dog you may say  
It has name, breed, color, legs, tail etc.  
It has behavioural characteristics such as it wags tail, barks, walks, sits, runs, jumps, etc.

# Object: Attributes and Behaviors

- An object is a thing (person, place, animal, vehicle, etc.) that can be described through its attributes and behaviors. Attributes are characteristics or qualities that describe an object.

Object	Characteristic Attributes	Behavior or Capabilities
Car	Color, make, model, registration number, fuel capacity, etc.	Start engine, turn left/right, accelerate, stop, turn on wipers, etc.
Person	Name, age, height	Speak, eat, walk, run, sleep, etc.
Bank Account	Account number, account type, account balance	Deposit money, withdraw money, view balance
Television	Screen size, shape, display type	Switch on, switch off, increase/decrease volume, change channels, etc.
Course	Name, location, days offered, credit hours, professor	Add student, delete student, get batch schedule, etc.



# Characteristics of Objects

- Each type of object has its own set of attributes and behavior that distinguishes it from other objects.
- Objects have external attributes that are externally visible to the public.
  - For example, a car's registration number or a person's age.
- Objects can also have some internal attributes that are hidden from the public.
  - For example, a car engine's internal parts, a person's internal organs.

# Collection of Objects

- You can observe a line of people waiting at the ATM to withdraw money.
  - There may be a finite number of people, say 10. Each one is a bank customer with common attributes and behavior but has a unique identity.
- Sometimes, you can see a queue of cars waiting in the parking lot to park their vehicle based on availability using a digital meter.
  - Each one of them is of the car object type, having common attributes and behavior but having a unique registration number.
  - Based on the total number of slots in the parking lot, you can park the car in the free slot available that is displayed on the digital meter.
- Sometimes, objects can be collectively stored, and you may need to manipulate the objects inside them.

## Let's Discuss

- Do objects exist only in the real world?
- Can we have objects in software programs?
- How do we store a collection of objects?



# Utilize Arrays to Model Aggregate Data







## Learning Objectives

- Create objects to model data
- Create arrays to model aggregate data
- Explain array structure
- Apply `for` and `for...of` loops to access array elements
- Pass an array as a function parameter

Let's consider a scenario where a customer is placing an order for a take away meal.

Multiple objects are involved and tracked while you place an order.

Customer: Person who places the order

Food Items: List of food items and their quantities

Order: Restaurant uses order details to deliver the food items to the customers

It is important to identify various objects, their attributes, and their behavior to implement them in a program.

# Scenario: Order Placement For a Meal





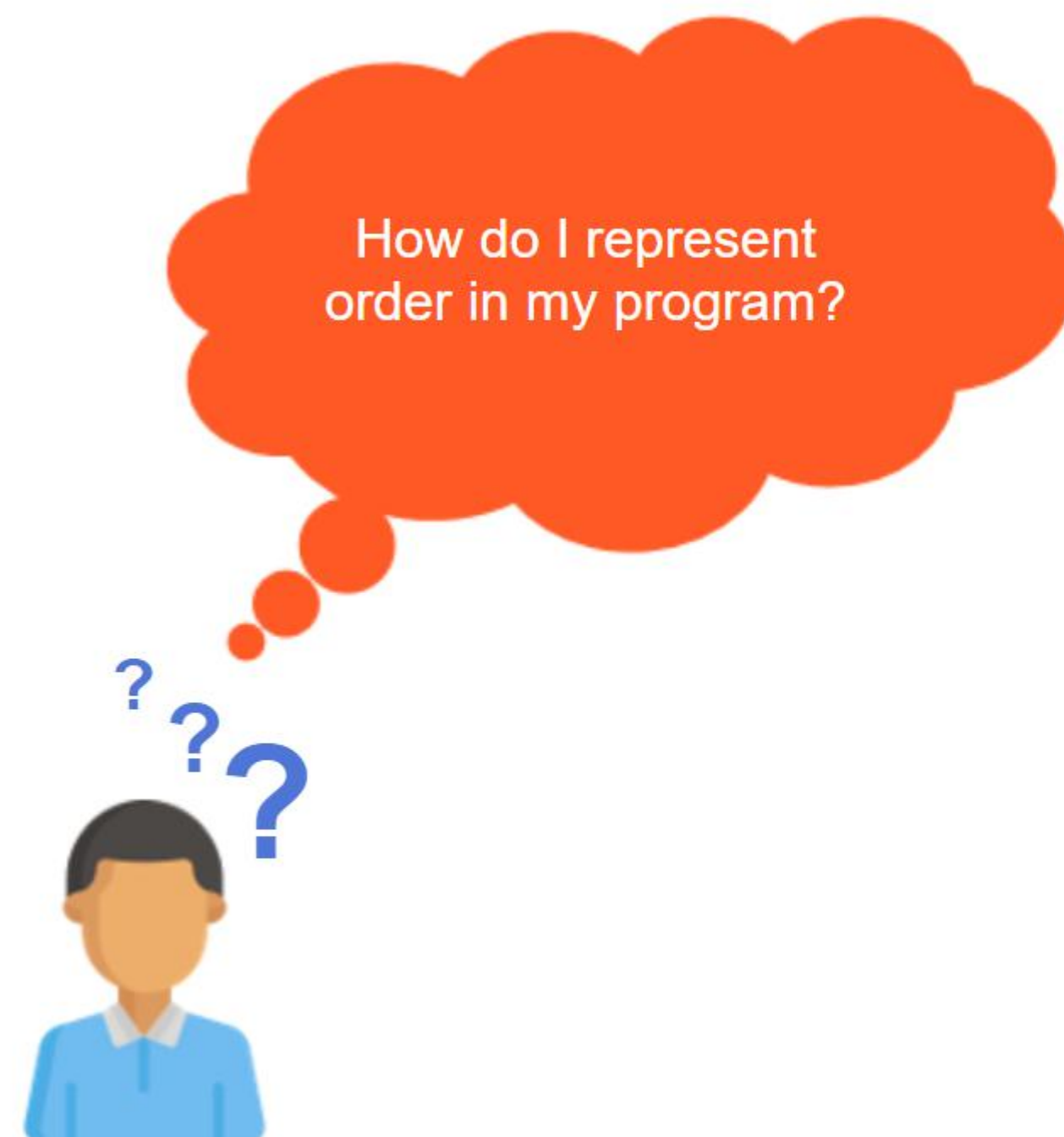
- An order has attributes like
- Order number
- Order date
- Item name
- Quantity ordered
- Customer name
- Deliverability
- An order's behavior can be
- Generated
- Cancelled
- Printed
- Re-ordered

# Let's Analyze an Order Object



Slide Note

Menu





## Model an Order object

Create a JavaScript object that models an order with specifications like orderNumber, orderDate, itemName, price, quantity and customerName .

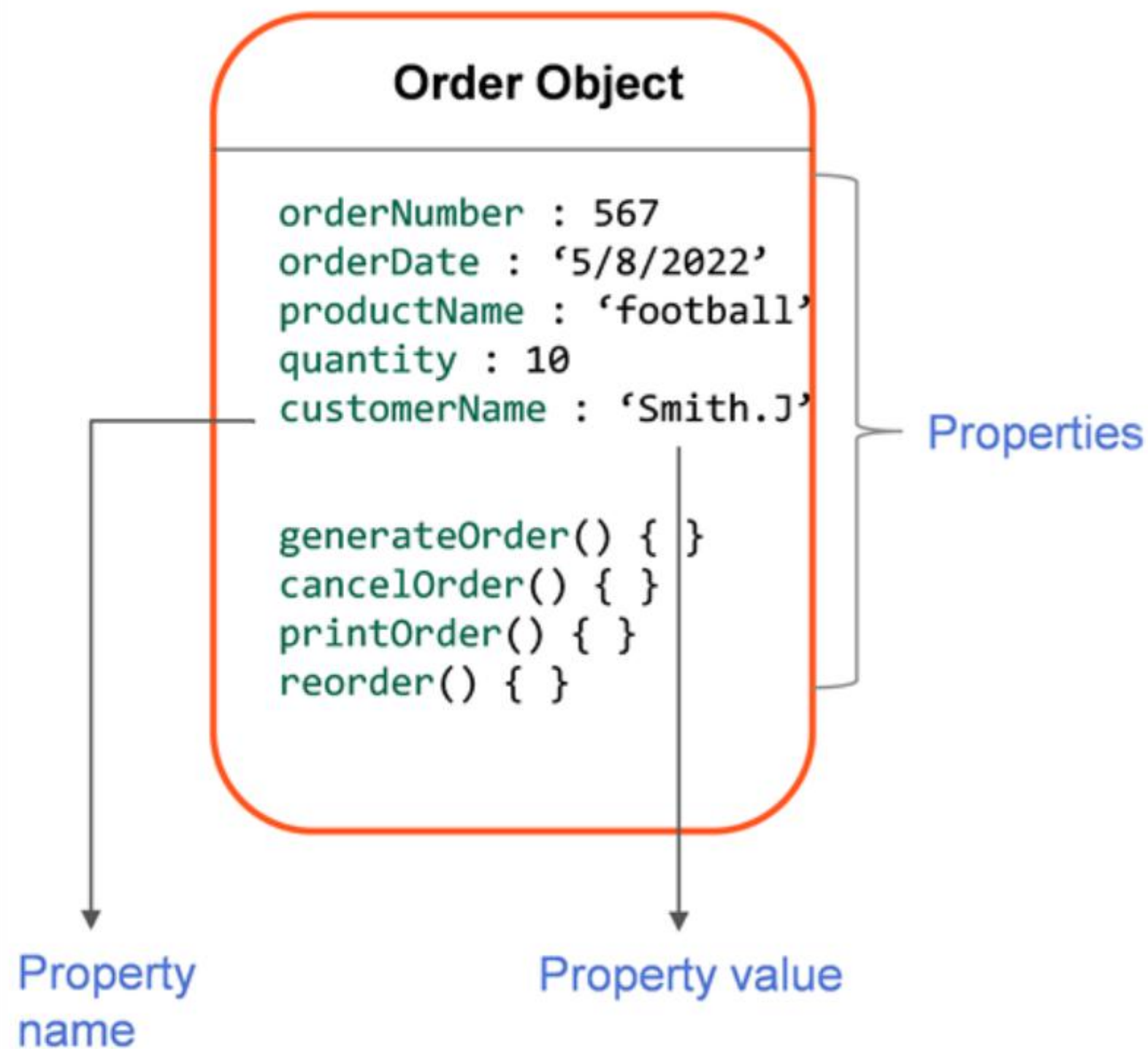
Display the specifications using the dot operator.

Click here for [demo solution](#).

DEMO



# Summarizing Object Learning



## Object in JavaScript

- An object in JavaScript is a collection of properties.
- A property is an association between key and value.
  - Key is the name of the property
  - Value is the data assigned to the key
- The property could be a:
  - Variable that stores attributes' data
  - Function that represents behavior
- Function inside an object is termed **Method**.



The other alternative way is to create object using constructor.

Constructor is a function that specifies name, properties and methods of object.

Inside function, use keyword *this* to assign values to the object's properties based on the values passed to the function.

Create an instance of the object with new.

In below code:

Employee is name of object type

firstName, lastName, email are its properties

```
function Employee(firstName, lastName, email) {
```

```
  this.firstName = firstName;
```

```
  this.lastName = lastName;
```

```
  this.email = email;
```

```
}
```

```
const employee = new Employee('Nancy','Davilio','nancy.d@gmail.com');
```

# Create Objects in JavaScript

- Popularly, objects in JavaScript can be created using an initializer.
- Initializer uses object literal notation, i.e. { }.
- Object initializers are expressions.
- Each object initializer results in a new object being created.
- In the code shown,
  - employee is an object.
  - firstName, lastName, email, and fullName() are properties of an employee.

**Note:** fullName() is a method which combines firstName and lastName.

Property value

```
const employee = {  
  firstName: "Tina",  
  lastName: "Keith",  
  email: "tina.k@gmail.com",  
  fullName() {  
    return firstName + "." +  
    lastName  
  }  
}
```

Property name



# Think Like a Programmer

- 10 Orders → How many object variables will be created ?
- 15 Customers --> Should we create 15 different object variables to represent 15 customers?
- 100 Food Items
- 15 Waiters

**Arrays can resolve these problems as they are used to store aggregate data in a single variable name.**

# Array: Complex Aggregate Datatype

- An array is a complex data type.
  - Specifically, it belongs to a category of complex data types called "aggregate" data types.
- The word *aggregate* is used to describe these data types because the name associated with a variable of these data types represents an aggregation of data.
- The name of the array collectively represents all the elements of the array.
- In the example below, `stu_name` is the array name, and the values inside the square brackets are the array elements.

```
let stu_name = ["Britto", "Charles", "Thomas"];
```

- These are zero-indexed,
- The first element of an array is at index 0
- The second is at index 1, and so on.
- The last element is at the value of the array's size minus 1.

# Characteristics of an Array

- JavaScript arrays are not primitive but are instead array objects.
- JavaScript arrays are resizable and can contain a mix of different data types.

```
let marks = [55,69,95,83,71];  
  
let random = ["Apple", "Kiwi", 8, 98.5];
```

- To identify different elements of an array, we use the concept of indexing.
- An index is the position at which the element is stored in the array.
- Array elements are zero-indexed:
  - The first element of an array is at index 0.
  - The second is at index 1, and so on.
  - The last element is at the value of the array's size minus 1.
- Array size is the number of elements in an array.
- Elements are stored contiguously in memory, i.e., in consecutive memory locations.

# Create an Array

- Arrays can be created in multiple ways.
- The easiest way to create an array is using an array literal.

```
let stu_name = ["Britto", "Charles", "Thomas"];
```

stu\_name: Variable name for the array

[ ]: Square brackets are the notation for array declaration.

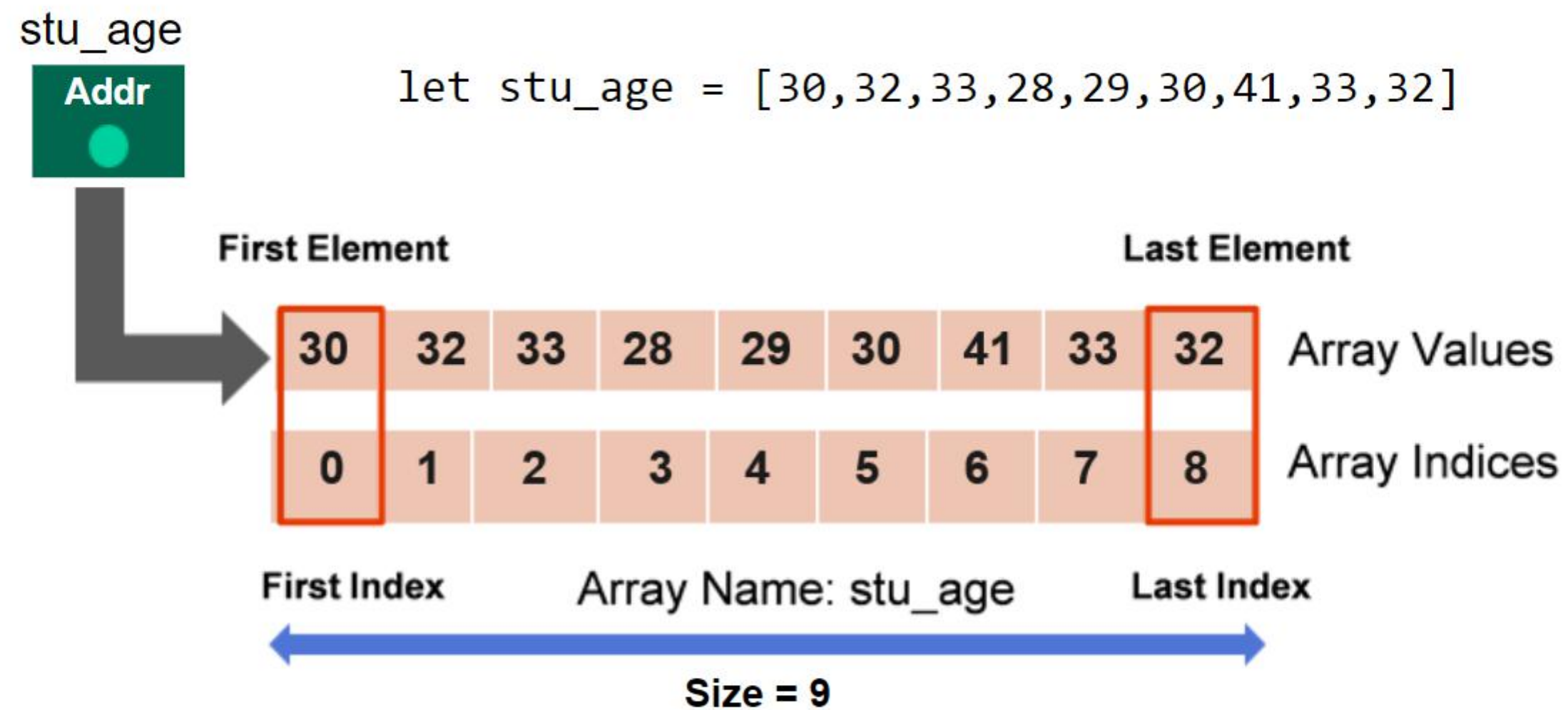
- You can also create an array and then provide elements by initializing with values.

```
let stu_name = [];  
stu_name[0] = "Britto"; //first element's index is 0  
stu_name[1] = "Charles"; //second element's index is 1  
Stu_name[2] = "Thomas";
```



# An Array Name Is the Address of the Array Object

- Since arrays are objects, the standalone name of the array is the reference to the memory address where the array object is stored.



# Array Built-in Length Property

- JavaScript provides built in properties and methods for the array object.
- One of the important properties is `length`.
- The `length` property reflects the number of elements in an array.

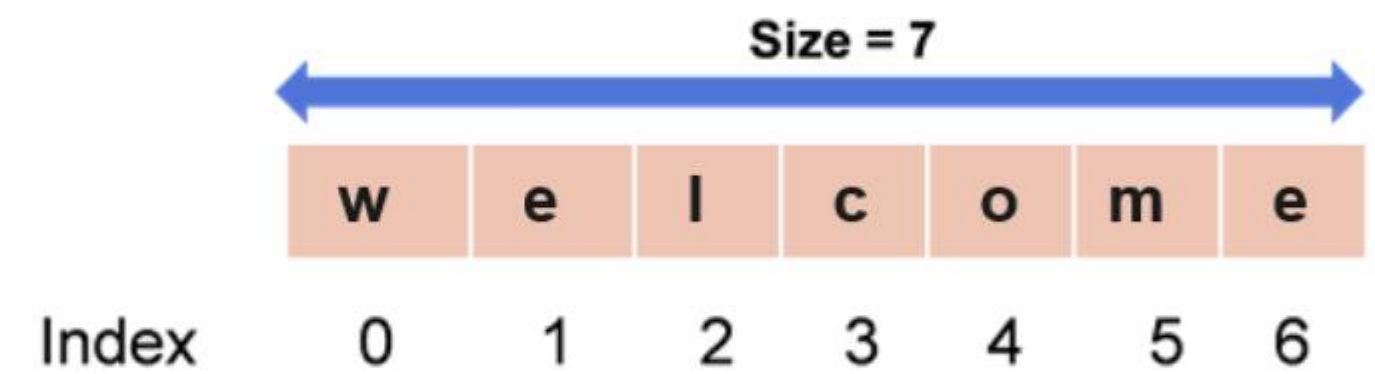
```
let marks = [55, 69, 95, 83, 71];  
let random = ['Apple', 'Kiwi', 8, 98.5];  
  
//Array size is determined using length property  
let size = marks.length; //returns 5  
size = random.length; //returns 4
```



# Access Arrays Using Index

- Since the name of the array refers collectively to the aggregated data structure containing all the items stored in the array, a method is required to identify the individual items.
- You access an array element by referring to the index number using [ ] notation.

```
let greet =  
['w', 'e', 'l', 'c', 'o', 'm', 'e'];  
//Returns the number of elements  
let size= greet.length;  
//Accessing first element  
console.log( greet[0]); // 'w'  
//Accessing second element  
console.log( greet[1]); //'e'  
//Accessing last element  
console.log( greet[size-1]); //'e'
```



## An Array Used to Store Age of Players in Soccer Team

Write a JavaScript function to store the age of all the players of a soccer team and represent them in an array.

1. Create an array by assigning values to the array.
2. Print all the values of an array using index position.

Click here for the [demo solution](#).

DEMO





# Access Array Elements Using Loops

## Access Arrays Using For Loop

- Enumerating every element of an array line-by-line is cumbersome.
- Loops are used to access array elements by their index position.
- The for loop is used to iterate through an array and print the elements of the array.
- Initialize `i = 0` as indexing start with 0.
- Increment `i` to get the next index position.
- The loop iterates until the last element of the array by checking Boolean condition `i < size`, where `size` is the number of elements in the array.
- Note: The last index is `(size-1)` since the index begins with 0.

```
let fruits = ["Apple ", "Mango ",  
"Banana"];  
let size = fruits.length;  
//index value begins from 0 to  
(size-1)  
for( i = 0; i < size; i++) {  
    console.log( fruits[i]);  
}
```

Apple  
Mango  
Banana

You can use "let" instead of "const" when there is a need to reassign the variable inside the for...of block.

```
let fruits = ["Apple ", "Mango ",  
"Banana"];  
let fruitString;  
for( const fruit of fruits) {  
  fruitString += fruit;  
}  
console.log(fruitString);
```

Apple Mango Banana

## For...of Loop

- A for...of statement loops through the values of an iterable object like an arrays, a string, etc.
- The loop traverses the array until the last element.
- For each iteration, it stores the element in the variable and executes the body of the for...of loop.
- In this example:
  - In the first iteration, "fruit" will be "Apple."
  - In the second iteration, "fruit" will be "Mango."
  - In the last iteration, "fruit" will be "Banana."
- This loop never accidentally exceeds the bounds of the array since it always stops at the last element; hence, it is much safer to use.
- This loop was introduced in the ES6 version.



# Quick Check

How do you determine the number of elements in an array?

```
let employee = ['Thomas', 'Peter', 'James'];
```

1. `employee.size()`
2. `employee.length`
3. `employee.size`
4. `employee.length()`



# Quick Check: Solution

How do you determine the number of elements in an array?

```
let employee = ['Thomas', 'Peter', 'James'];
```

1. `employee.size()`
2. **`employee.length`**
3. `employee.size`
4. `employee.length()`



## Sum of Even Numbers

Write a JavaScript function that accepts an array of numbers, iterate it using a for...of loop, and print the sum of all even numbers.

Click here the [demo solution](#).

DEMO





## Nested Loops in Arrays

Write a JavaScript function to extract each character in an array of string names using nested for loop.

Click here for [demo solution](#).

DEMO





## Think and Tell

- You may need to use an array to calculate the maximum number from a given set of numbers using a JavaScript function.
- How can we pass an array as a parameter to the JavaScript function?

```
function calculateMax(listOfMarks){  
  //code to calculate max value  
  let maxValue = -Infinity;  
  for (item of listOfMarks) {  
    if (item > maxValue)  
      maxValue = item;  
  }  
  return maxValue;  
}  
  
let marksList =[45, 65, 78, 92, 70,  
39];  
console.log(`Max Value:  
${calculateMax(marksList)}`);
```

## Arrays Passed as Parameters

- An array can be passed to a method as a parameter just as normal variables can.
- Globally, a marksList array is created.
- The calculateMax method has a parameter of array type called listOfMarks.
- When calling the method calculateMax() globally, we pass only the reference variable marksList without the [] brackets.



## Arrays as Parameters in Functions

Write a JS function that converts all the even numbers present in the array to the odd numbers and stores it in another array.

Find the maximum of marks from an array of marks passed as parameter to a JavaScript function.

Click here for the [demo solution](#).

DEMO



## Arrays as a Return Type

- An array can be a return type from a method.
- A variable of array type needs to be created to hold the value that is getting returned from the `evenToOdd()` method.
- In the code shown, the `evenToOdd()` method returns an array which contains only odd numbers.
- There's no need to specify the `[]` along with the array name while returning the value.

```
function evenToOdd(arr){  
  let newArray = [];  
  //code to convert array values from  
  even to odd number  
  return newArray;  
}  
let arr = [23, 56, 12, 32, 81, 95,  
27];  
//prints 23,57,13,33,81,95,27  
let result = evenToOdd(arr);  
console.log(`Result Array:  
${result}`);
```



```
const colors = ["Red", "Blue",  
"Green "];  
const [red, blue, green] = colors;  
console.log(red); //Red  
console.log(blue); //Blue  
console.log(green); //Green
```

## Destructuring an Array

- **Destructuring** in JavaScript is a simplified method of extracting multiple properties from an array.
- It takes the structure and deconstructs it down into its own constituent parts through assignments by using a syntax that looks like array literals.
- Without de-structuring, if we want to extract data from an array, we need to use a separate variable to assign each individual element of an array and repeat the process again and again.



## Reverse an Array Using Array Destructuring

Write a JS function which reverses the array elements using Array destructuring.

Click here for the [demo solution](#).

DEMO



## Array Spread Operator

- The JavaScript spread operator is denoted by three dots ( ... ).
- It allows us to quickly copy all or part of an existing array or object into another array or object.
- The spread operator is often used in combination with the array de-structuring.

```
const colors1 = ["Violet", "Indigo", "Blue"];
const colors2
= ["Green", "Yellow", "Orange", "Red"];
let colors = [...colors1, ...colors2];
//Prints all the rainbow colors
console.log(colors);
//Assign the first and second items
from num array to variables and put the
rest in an array
const num = [1,2,3,4,5,6];
const [one, two,...rest] = num;
```



# Quick Check

Predict the output of the following code:

```
let values = [1, 2, 3, 4, 5, 6, 7];  
let size = values.length;  
for( i = 0; i <= size; i++) {  
    console.log( values[i]);  
}
```

1. 1,2,3,4,5,6,7
2. 1,2,3,4,5,6,7, undefined
3. Syntax Error
4. Runtime Error





# Quick Check: Solution

Predict the output of the following code:

```
let values = [1, 2, 3, 4, 5, 6, 7];  
let size = values.length;  
for( i = 0; i <= size; i++) {  
    console.log( values[i]);  
}
```

1. 1,2,3,4,5,6,7
2. **1,2,3,4,5,6,7, undefined**
3. Syntax Error
4. Runtime Error

