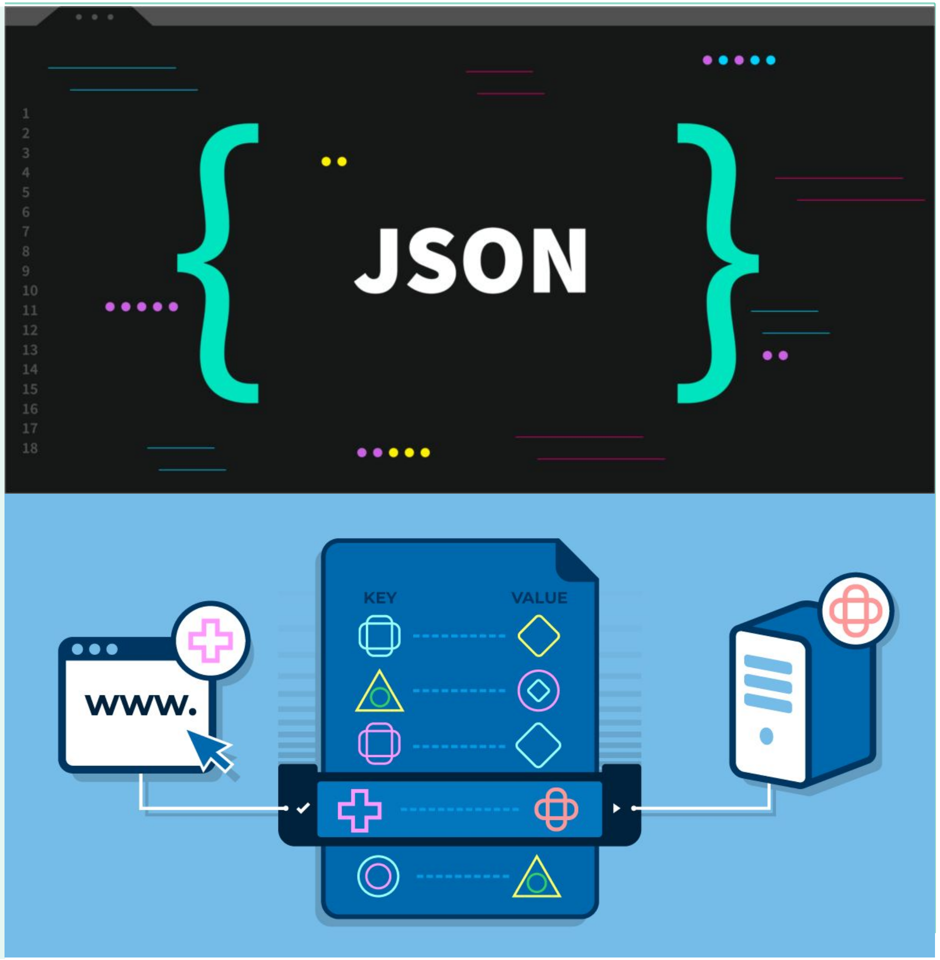
# JSON and Asynchronous JS

By Aquib Ajani



# Introduction to JSON

JSON stands for JavaScript Object Notation.

To transfer data between servers and browsers, the data needs to be in text format only.

JSON is a text format, which can be converted to object notation; hence, it is called JavaScript Object Notation.

Therefore, JSON is used for storing and exchanging data. It can be stored as JavaScript (JS) objects. JSON can be converted to text while being sent across the network, and they can be converted back to objects when they are received.

→ Key always in " "

```
{
  "name": "Ranchoddas Shamaldas Chanchad",
  "age": 30,
  "nationality": "Indian"
}
```

JSON

Sending b/w networks in Text format

After Receiving Convert into obj for using data

# Introduction to JSON

## JSON Data Types $\rightarrow$ $\boxed{6}$

| | Data Type | Description |
|---|---|---|
| 1 | number | This data type is used for all numeric types, be it integers or floating-point numbers. |
| 2 | string | This data type is used for a string of Unicode characters. You need to enclose the value in double-quotes. |
| 3 | boolean | This data type is used for a true/false value. |
| 4 | array | This data type contains an ordered list of 0 or more values. The values are wrapped in square brackets. |
| 5 | object | This is an unordered collection of key-value pairs. You can have multiple objects embedded in a JSON object. |
| 6 | null | This data type is used for null values. |

~~undefined~~

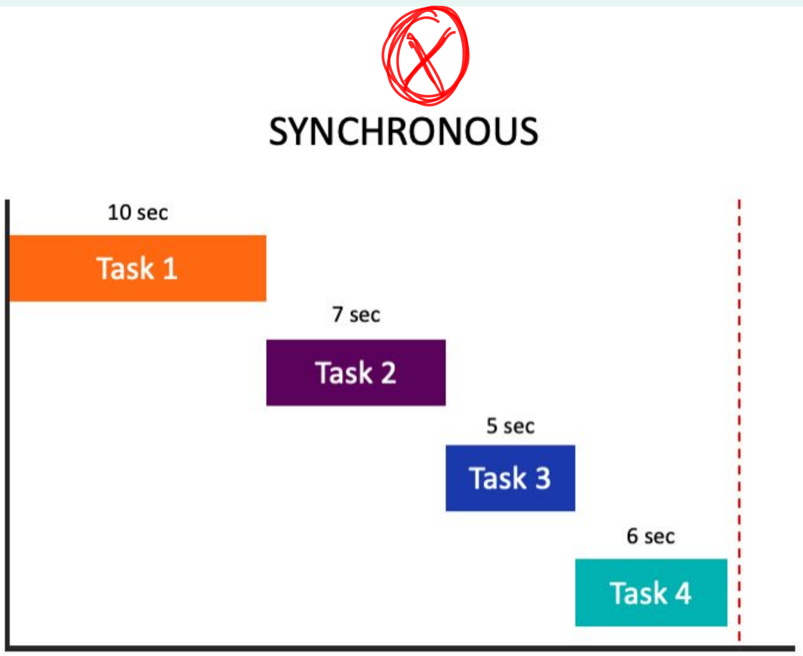
## Introduction to JSON

### JSON Parsing and Converting JSON to Strings

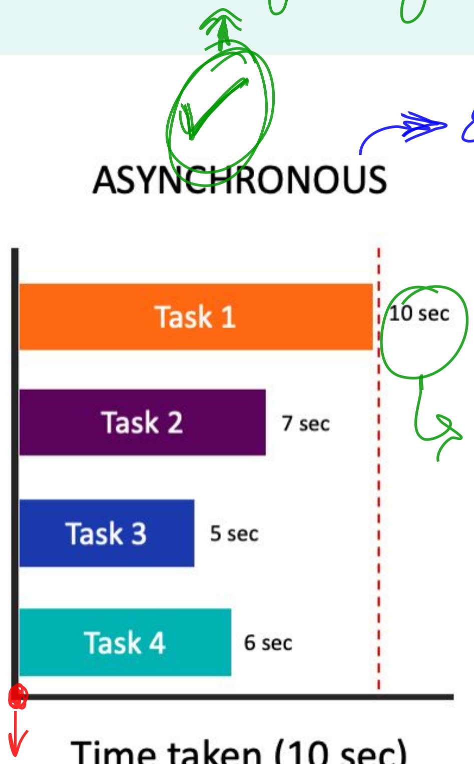
```
let myJson = '{"name": "John Doe", "age":
27,"nationality": "English", "cars": ["BMW", "Audi"]}’;

let obj1 = JSON.parse(myJson);
console.log(obj1);

let obj2 = {
   name: “John Doe”,
   age: 27
}
let jsonText = JSON.stringify(obj2);
console.log(jsonText);
```

# Asynchronous Programming

Favoured by everyone



Time taken (28 sec)



Time taken (10 sec)

Starting Point

Everything start Loading Parallely takes own specific time

Total time taken equal to Longest task

## Asynchronous Programming in JS

### setTimeout()

In setTimeout() function, the first parameter is the function whose code will be executed and the second parameter will be the time in milliseconds.

```
let showAlert = () => {
    alert("Hello World");
}

let myTimeout = setTimeout(showAlert, 3000);

alert("Goodbye World");

clearTimeout(myTimeout);
```

In the example given above, although the alert "Goodbye World" is placed after the setTimeout() function, it will be executed first. Then, after 3,000 milliseconds or 3 seconds, the showAlert() function will be executed and "Hello World" will be displayed in an alert box.

We can clear this timeout using the clearTimeout() function.

## Asynchronous Programming in JS

**setInterval()**

In setTimeout() function, the first parameter is the function whose code will be executed and the second parameter will be the time in milliseconds.

```
let counter = 1;

let incrementCounter = () => {
    counter = counter + 1;
    console.log(counter);
}

let myInterval = setInterval(incrementCounter, 1000);

clearInterval();
```

In the example given above, the counter will keep increasing by one every second until we stop it using the clearInterval() method.