**Web Version VS Desktop Version IDE**

When comes it comes to desktop version – smooth access, offline access and local libraries and need to install and configure . And hands on suggestions.

In Web version it woks online and no local libraries required, chances of server mal-functions. Collaborative work offering.

**studio.code.org – play around algorithms**

**API :**

**API** just a piece of software code to talk to another, example **REST API** is used to serve the client requests and server responses.

API that confirms to the **REST architectural style** is called REST API. **REST over HTTP**

**HTTP** is a protocol used to communicate through web server.

**HTTPs** is set of rules used to communicate securely.

**Request Method: GET, PUT, POST and DELETE**

**Status code: 200 ok, 500 series is server-side errors 400 series for client side error**

**Content-Type: -- the tech used in current web site – ex: text/html**

Structure: a process used to build something

Structure Data - **JSON** - key and value a better way for computer tom understand.

To get html pages or we pages mostly preferred protocol is **HTTP** (Content-Type**: text/html**). Either side when web site needs a data from database then it better to go for **REST** due to it uses JSON structure (**Content-Type: application/JSON**) ex: posting comment.

REST APIs commonly use the HTTP protocol to send requests & receive responses.

It is in terms of the data returned that an API request differs from a usual HTTP request for a webpage.

* HTTP requests for webpages return HTML, CSS & JavaScript files which are rendered by the browser and displayed to the user.
* But, in the case of APIs, the request can be for any data (not just webpage) and the response is read by the requesting program which interprets the data.

JSON (JavaScript Object Notation - how cryptic :| ) is a standard data format that is easily “understandable” by applications

* It can be handled well in most languages
* So the data format in REST is usually JSON

Curl – client URL – CLI which works for non GUI client requests.

Copy a payload from network tab try POST method.

**Micro Experience:** developing code testing code and making production ready code

**REST API calls using Browser:**

* **Modifying the end point of URl**

**REST API calls using Programs**

* import java.io.BufferedReader;
* import java.io.IOException;
* import java.io.InputStreamReader;
* import java.net.HttpURLConnection;
* import java.net.MalformedURLException;
* import java.net.URL;
* class restAPI {
* public static void main(String[] args) throws MalformedURLException, IOException {
* // create url
* URL url = new URL("https://crio-xflix.herokuapp.com/v1/videos/602d228e672f010020e5e95d");
* // Send Get request and fetch data
* HttpURLConnection conn = (HttpURLConnection) url.openConnection();
* conn.setRequestMethod("GET");
* BufferedReader br = new BufferedReader(new InputStreamReader(
* (conn.getInputStream())));
* // Read data line-by-line from buffer & print it out
* String output;
* while ((output = br.readLine()) != null) {
* System.out.println(output);
* }
* conn.disconnect();
* }
* }

**Data Structure:[DSA]**

This is terminology/ formulae to store, retrieve and use the data effectively.

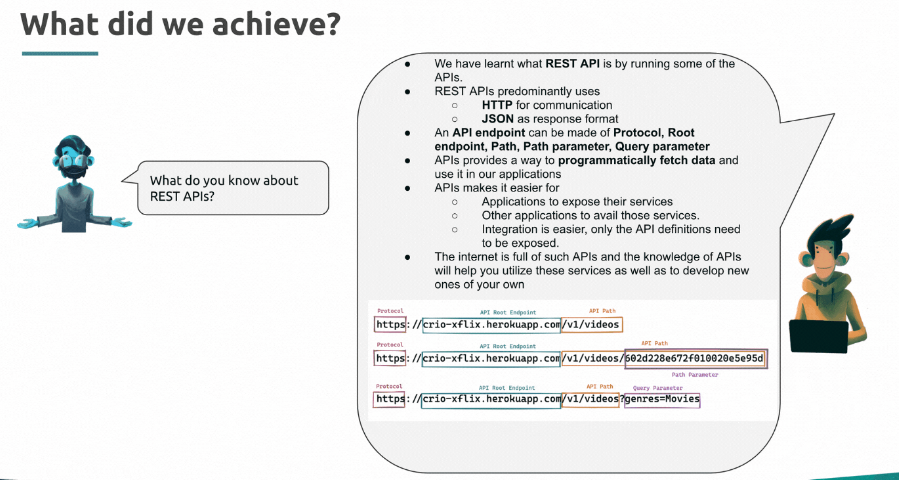
**Algorithm :**

Is a set of instructions to perform a task to given to given to computer via program based on certain requirements.

**Pattern:**

Two pointer – using two variables – time complexity: O(n)

Brute force – two nested loops – Time complexity: O(n\*n)



Query parameter vs Path parameter

Query start with “?” after the end point to request – to retrieve specific ID

/users?id=123 # Fetch a user who has id of 123

and where Path parameter start with “/” after endpoint to navigate. – to retrieve what ever data is present at other side

/users/123 # Fetch a user who has id 123

 If you want to identify a resource, you should use Path Variable. But if you want to sort or filter items, then you should use query parameter.

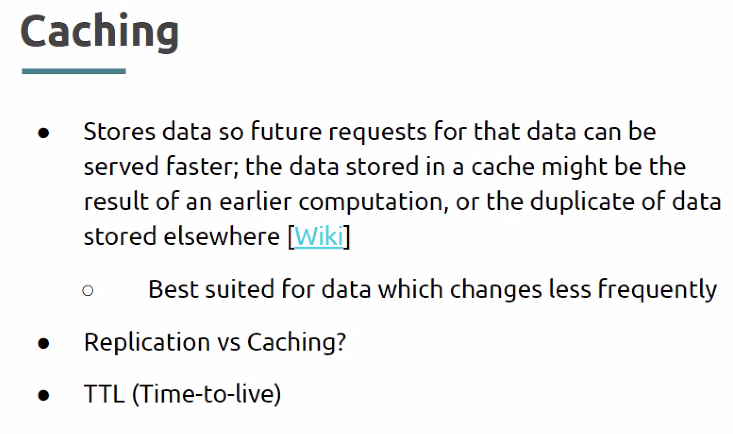
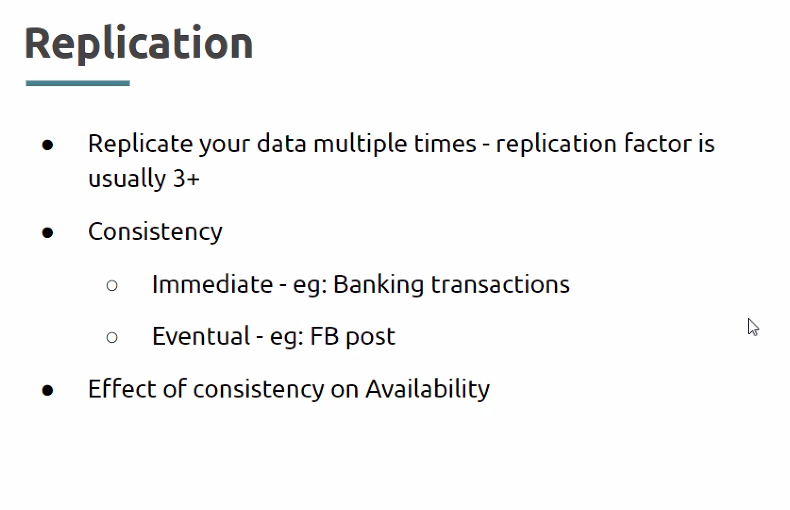
#### System Design

#### Distributed system:

1. Multiple computers2. Network Connectivity3.Work together for a common goal

Connecting multiple computers to shared network and working on same goal.

3-Tier Architecture > frontend (presentation layer) + backend (business logic) + (backend) Database



Load Balancer

# **JAVA SCRIPT**

#### SESSION 1

/\*\* -------Session 1 -----\*/

//console.log("Why do we use JS?")

//console.log("ANS: To build interactive web applications")

/\*\*---- Primitive -----\*/

//console.log("What are primitives?")

//console.log("values can not be changed\nPrimitives datatypes are numbers, Str..ings, Booleans, Null, Undefined, Symbols")

// when a variable is declared, it is `NOT assigned a value , so the default value is undefined` uintentional missing assignment

//where null is assigned, it is intentional missing assignment

// Symbols are unique and immutable primitive values

/\*\* ------Typeof------\*/

//console.log("I am "+ typeof(123))

//console.log("I am "+ typeof(typeof(123)))

// three ways of declaring variables

//var x = 123// declaration with initiazation

//let y;// declaration without initiazation

/\*----var----\*/

//x = 234; // re-assignment is allowed incase of var declaration

//console.log(x)

//var x = 123 //redecalration is allowed incase of var declaration

var greeter = "hey hi"

console.log(greeter)

var greeter = "say Hello instead"

//console.log(x)

// the worst case of var isbbecasue when coming over a large code of program var allows to user to declare a same variable any times, without any error.

/\*----const----\*/

//const marks = 456// when const is can't be change

// marks = 789 // re-assignment is not allowed incase of const declaration

//console.log(marks)

/\*----let----\*/

let para = "I am let"

console.log(para)

//let para = "I can'r redeclare so i am let" // re-declaration is not allowed incase of let declaration

para = "I am let because, my variable can be " // re-assignment is allowed incase of let declaration

//console.log(para)

// let is replace ment of var in case of use cases, let is block scoped, var is function scoped

// var is globally scoped, let and const are block scoped

//varible is pointer to value, let and const are immutable

// must follow naming constraints smallcase uppercse camelcase

#### SESSION 2

/\*\* ---Arthimatic operators------ \*/

/\*\* ---Relational operators------ \*/

/\*\* ---Logical operators------ \*/

/\*\* ---Conditional operators------ \*/

//== compres value where === compare data type

//strict(!==) comparing with datatype&value and where non-strict (!=) only value.

/\* ---Mathimatical operators and Precedure ------ \*/

// BODMAS BASMDR

// + , - , \* , / , % , \*\* , ++ , -- , += , -= , \*= , /= , %= , \*\*=

//console.log(2+3%1)

//console.log((2+3)%1)

//\*\* ---NAN operators------ \*/

//when inavlid mathimatical opertators are used, the result is NaN (not a number)

//console.log(1/"a") but the type is Number

//var age =16;

//if(age>0){

// console.log("age is a possitive number, can be considered");

//}

/\* ----FUnctions------ \*/

// Function Declaration

// function add(a,b){ // PARAMETERS

// return a+b;

// }

// add(10,20); // ARGUMENTS)

/\*\* ---Activity --- \*/

function multiple(x,y){

return x\*y;

}

console.log(multiple(5,2));

console.log(multiple(5,1));

///https://replit.com/@KomalSharma26/Crio-Foundations-JS-1-B2-Nov#Session2/Session2.js