**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 :**

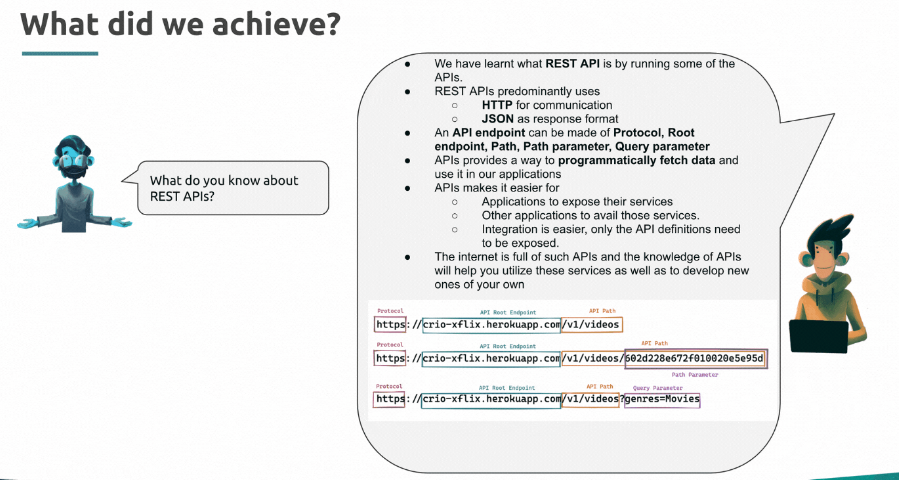
Is a set of instructions to perform a task to given to given to computer via program based on certain requirements.

**Pattern:**

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

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

Sliding window technique – O(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.

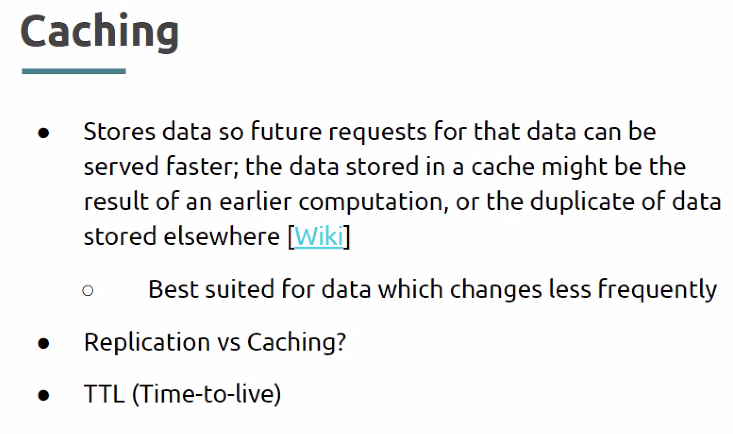
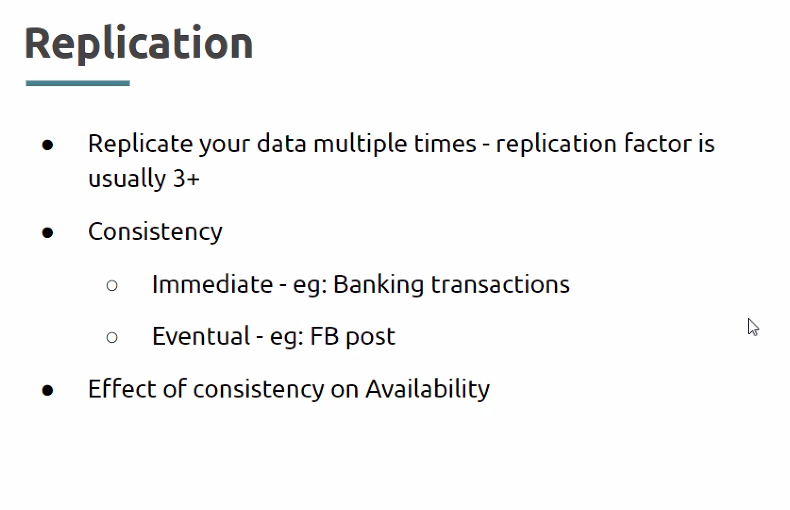
#### System Design

#### Distributed system:

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

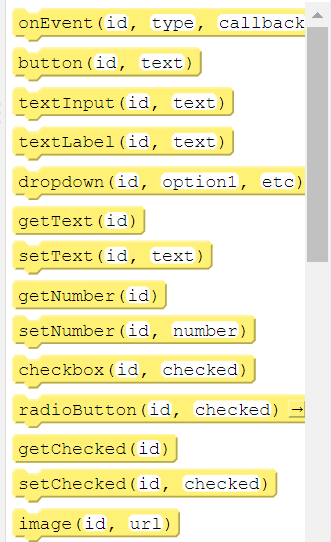
Connecting multiple computers to shared network and working on same goal.

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



Load Balancer

# **JAVA SCRIPT**



Local scope – function scope and block scope

Global scope -

Lexical scope - the scope which used by a child/ nested functions , the nested function can have access to variables declared in parent scope, lexical scope or lexical environment forms the closures.

#### All ABOUT VAR

* var variable can be redeclared and reinitialized.
* var global scoped. if it declared in function, it has function scope
* if var declared in block also can be accessed outside too.
* var allows variable hoisting, it causes possible errors.
* var declarations are either globally scoped or function-scoped

#### All ABOUT LET

* let can’t redeclared, it throws error, can be reinitialized.
* let have local scoped, If it is declared inside block , can’t accessed outside. Block scope
* let won’t support hoisting.
* let declarations are either block scope or function scoped

#### All ABOUT CONST

* const can’t redeclared and can’t reinitialized. Must assign at first declaration.
* const local scoped, If it is declared inside block , can’t accessed. Block scope.
* Local scope – block scope, function scope

#### All About primitive and non-primitive data types

* **Primitive data types are** number, Boolean, strings, null, undefined and NAN. Inbuilt classes.
* **Non - Primitive data types are** arrays, objects and functions. Even these data types declared with const can be modified. non-primitive data types reference to same memory even they assigned. Obj a = b;

typeof – is used to check the data types of variable

**NAN –** if a devision is invalid like 2/”a”

#### All ABOUT Arithmetical Operators

To compare the variable value with its data type a ===b or a !== b(Strict mode).

Truthy and falsy operators -if(a) returns true if it is value other wise false.

#### All About Strings

immuatable

Template Strings `I am template string ${varibale}`

Str.length;

Str.indexOf()

Str.substring(start index, end index)k

Str.substr(startindex, length )

Str.split(“”)

Str.slice(from, to)

#### All About Arrays

Arrays are objects, hetrogeonous, index access

Two find a variable is array or not use var.isArray()

Arr.length, arr.push(), arr.unshift(), arr.pop(), arr.shift(), arr.slice(), ar.splice(at, no.of, add ele..)

Arr.reverse(), Arr.joins(“”)

Arr.sort() – it works for string array

For number array – Arr.sort((x,y) => x-y))

#### All About Objects

Objects are non primitive data types

Object properties are retrieved using dot or bracket

Object.name = “snb” or Object[“name”] = “snb”to add

Object [1] or Object[“1”]

Delete Object.age

#### All About De-structuring and spread operator

A process to retrieve the object or arrays values to targeted variable

const [ index0, , index2] = arr;

for arrays must declare equal number of array elements

const {name , age} = obj; // when obj properties have same name

**spread operator**

const arr1 = {…arr} -> to copy the array to another array

const obj1 = {…obj}

#### All About Math Functions

Math.floor() – to round of the integer to smaller integer side

Math.ceil() – to round of the integer to larger side

Math.round() – it rounds the number based on >=0,5 or < 0.5

Math.abs() – to return positive integer

Math.trunc() – to cut the integer to non decimal value.

Math.sign() - returns the -1 or 1 based on negative or positive number

Math.max() – to return the max value

Math.min() – to return the min value

To retrieve constants

Math.PI and Math.E

#### All About pass by reference and pass by value

**Pass by reference** is achieved by non primitive data types, where these data types are refers to same memory location even when modified or assigned to another variable.

**Pass by value** is achieved by primitive data types, because every new assign will create new reference in memory.

#### All About Arrow functions

* Must use the () – open and closing brackets , If a arrow function don’t have parameter. If a single argument no need to use brackets.
* More than one argument must use brackets (a,b)=>{}
* If a function have only one statement , can return it without a return keyword

#### All About Hoisting, Closure and Callback

**Hoisting** special behaviour of var keyword, even using of a variable can works when it declared after that. It called hoisting

**Closures** is a process of returning a function inside another function without calling, and assigning it to a variabale and calling the variable can do the same process. It is achievable because of lexical environment scope.

**Callback**  is a process to handling the function call to another function. Where a function is passed a an argument to function, and calling that callback function inside it at target place.