□ HTML Notes (Part 1):

□ Topic 1: <!DOCTYPE html>

- **Definition:** Declares the document type as HTML5 to ensure the browser renders it in standard mode.
- **Code Snippet:**

```
<!DOCTYPE html>
```

- **Output:**

  No visual output on browser.

  Ensures HTML5 standards and prevents quirks mode rendering.

□ Topic 2: <html>, <head>, <meta>, <title>

- **Definition:**
  - `<html>`: Root element.
  - `<head>`: Metadata container.
  - `<meta charset="UTF-8">`: Character encoding as UTF-8.
  - `<meta name="viewport">`: Responsive design.
  - `<title>`: Page title shown on browser tab.
- **Code Snippet:**

```
<html lang="en">
<head><!--contains website information-->
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
</html>
```

- **Output:**

  Page title: Document (displayed on browser tab).

  Sets language as English, encoding as UTF-8, responsive viewport.

□ Topic 3: Headings <h1> to <h6>

- **Definition:** Headings create structured titles from largest (`<h1>`) to smallest (`<h6>`).
- **Code Snippet:**

```
<h1>Hello!</h1>
<h2>Hello!</h2>
<h3>Hello!</h3>
<h4>Hello!</h4>
<h5>Hello!</h5>
<h6>Hello!</h6>
```

- **Output:**

  Displays:

  Hello! (largest)

  Hello!

  Hello!

  Hello!

  Hello!

  Hello! (smallest)

  Each heading decreases in size and importance, used for semantic structure.

□ Topic 4: Paragraphs <p>, Line Breaks

- **Definition:**
  - `<p>` defines paragraphs.
  - `<br>` creates a line break within elements.
- **Code Snippet:**

```
<p>Hello! My name is Deven Malla.</p>
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. <br>Nulla nam aperiam quam magni, debitis incidunt perspiciatis. <br>Cum a maxime iste.</p>
```

- **Output:**

  Displays two paragraphs. The second has line breaks creating new lines within the paragraph text:

  Lorem ipsum dolor sit amet consectetur adipisicing elit.

  Nulla nam aperiam quam magni, debitis incidunt perspiciatis.

  Cum a maxime iste.

□ Topic 5: Horizontal Rule <hr>

- **Definition:** Inserts a horizontal line to separate content.
- **Code Snippet:**

```
<hr>
```

- **Output:**
  Draws a straight horizontal line across the page, visually separating sections.

□ Topic 6: Ordered Lists <ol>, Unordered Lists <ul>, List Items

- **Definition:**
  - : Ordered list (numbered).
  - : Unordered list (bulleted).
  - : List item.
- **Code Snippet:**

```
<ol>
  <li>Red</li>
  <li>Blue</li>
  <li>Green</li>
</ol>
<ul>
  <li>Red</li>
  <li>Blue</li>
  <li>Green</li>
</ul>
```

- **Output:**
  Red
  Blue
  Green
  Red
  Blue
  Green

□ Topic 7: Image Tag <img>

- **Definition:** Embeds an image in the webpage.

- **Code Snippet:**

```
<img src="nagaland-university-logo.png" alt="Nagaland University logo">
```

- **Output:**

  Displays the Nagaland University logo if nagaland-university-logo.png exists in the same folder.

  If not found, shows alt text: “Nagaland University logo”.

□ Topic 8: Hyperlink <a>

- **Definition:** Creates clickable links to other resources.
- **Code Snippet:**

```
<a href="https://www.youtube.com/watch?v=kkOuRJ69BRY&list=PLSUlCIUmG02WfnUvc4hWPrJPvtH2mn37o">Learn HTML, CSS and JavaScript in Single Video</a>
```

- **Output:**

  Shows clickable text:

  Learn HTML, CSS and JavaScript in Single Video

  On clicking, redirects to the provided YouTube link.

□ Topic 9: Form Tag <form> and Input Types

- **Definition:**
  - `<form>`: Container for user inputs.
  - `<input>`: Form fields of various types.
- **Code Snippet:**

```
<form><!--container-->
   Enter name:<input type="text" placeholder="enter your name"><br>
   Enter password:<input type="password" placeholder="enter your password"><br>
   Enter date:<input type="date"><br>
   Select gender:<input type="checkbox">Male<input type="checkbox">Female<br>
   Select gender:<input type="radio">Male<input type="radio">Female<br>
   Select colour:<input type="color">
</form>
```

- **Output:**

  Text input with placeholder.

Password input with placeholder.

Date picker.

Two checkboxes (Male, Female).

Two radio buttons (Male, Female).

Colour picker to select any colour.

□ Topic 10: Video Tag <video>

- **Definition:** Embeds videos with controls.
- **Code Snippet:**

```
<video autoplay loop muted controls src=""></video>
```

- **Output:**

  Renders a video player with controls.

  Because src="" is empty, no video plays.

  Attributes: autoplay, loop, muted (starts silent in loop if video is provided).

□ Topic 11: Div Tag

- **Definition:** Generic container for grouping and styling elements.
- **Code Snippet:**

```
<div><!--used as a container to create divisions-->
   <h1>Hello!</h1>
</div>
```

- **Output:**

  Displays "Hello!" within a div block, allowing CSS styling or layout grouping.

□ Topic 12: Semantic Tags (<main>, <header>, <nav>, <article>, <aside>)

- **Definition:** HTML5 semantic elements for meaningful page structure.
- **Code Snippet (Commented):**

```
<!--
<main></main>
<header></header>
<nav></nav>
<article></article>
```

```
<aside></aside>
-->
```

- **Output:**

  Currently none because they’re commented out.

  When used, they structure pages for accessibility and SEO.

□ HTML Notes (Part 2):

□ Topic 13: Linking External CSS with `<link>`

- **Definition:** Connects an external CSS stylesheet to the HTML document for styling.
- **Code Snippet:**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="../CSS/style_01.css">
</head>
<body>
  <div id="parent">
    <div id="child1" class="child"></div>
    <div id="child2" class="child"></div>
    <div id="child3" class="child"></div>
  </div>
</body>
</html>
```

- **Output:**

  Loads styles from style_01.css.

  Displays a parent div containing three child divs styled via CSS (full CSS explanation will be in CSS topics).

  This structure is used for layouts, grids, flexbox demonstrations.

□ Topic 14: External Icon Libraries

- **Definition:** Adds icons from external libraries like Remix Icon via CDN.
- **Code Snippet:**

```
<link
href="https://cdn.jsdelivr.net/npm/remixicon@4.5.0/fonts/remixicon.css"
rel="stylesheet"/>
```

- **Output:**

  Allows you to use icons like:

```
<i class="ri-arrow-right-up-line"></i>
```

Which renders as an arrow icon styled by Remix Icon CSS.

□ Topic 15: <nav>, <section>, and Website Layout

- **Definition:**
  - `<nav>`: Navigation bar container.
  - `<section>`: Groups thematic content.
- **Code Snippet:**

```
<nav>
  <h2>The Green One</h2>
  <div class="part2">
    <a href="#">About Us</a>
    <a href="#">Services</a>
    <a href="#">Projects</a>
    <a href="#">Let's talk</a>
  </div>
</nav>
<section>
  <div class="hero-text1">
    <h1>Digitize</h1>
    <div class="video">
      <div class="play">
        <img height="32" width="32" src="play-circle-fill.svg" />
      </div>
    </div>
  </div>
  <div class="hero-text2">
    <h1>Ideas</h1>
    <p>The art of visual communication, creatively<br>
    impacting the world around us-one good<br>
    design at a time, design like you mean it!<br>
    </p>
```

```
  </div>
</section>
<div class="image">
  <div class="arrow">
    <i class="ri-arrow-right-up-line"></i>
  </div>
</div>
```

- **Output:**

  Displays a navigation bar titled The Green One with links.

  Hero section with "Digitize" and "Ideas" as headings, plus descriptive text.

  Video play button icon (if play-circle-fill.svg exists).

  Image section with an arrow icon styled as per CSS.

□ Topic 16: Using <script> to Link External JavaScript

- **Definition:** Embeds or links external JavaScript files for functionality.
- **Code Snippet:**

```
<script src="../JavaScript/script_01.js"></script>
```

- **Output:**

  Runs the linked JavaScript file when the HTML loads.

  Example from your file displays:

```
<h1>Study JS friends!!!</h1>
```

Plus executes whatever is inside script_01.js (which we will dissect fully in JavaScript topics).

□ Topic 17: DOM Manipulation Example – Bulb Switch

- **Definition:** Uses JavaScript to manipulate HTML elements and styles based on user actions.
- **Code Snippet:**

```
<div id="bulb"></div>
<button>on</button>
<script src="../JavaScript/script_01.js"></script>
```

- **Output:**

  Displays a circular bulb div and a button labeled "on".

When clicked, JavaScript toggles bulb colour to yellow (on) or transparent (off).

Dynamic interactivity using DOM (Document Object Model) manipulation.

□ Topic 18: Empty <body> for JavaScript Practice Files

- **Definition:** Sometimes kept empty to test scripts without page clutter.
- **Code Snippet:**

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <script src="../JavaScript/script_03.js"></script>
   <title>Document</title>
</head>
<body>
</body>
</html>
```

- **Output:**

  Renders an empty page.

  Executes script_03.js code behind the scenes for alerts, logs, or DOM manipulation.

□ Topic 19: HTML Comments

- **Definition:** Notes within code ignored by browsers.
- **Code Snippet:**

```
<!--used as a container to create divisions-->
```

- **Output:**

  No output on the page.

  Helps document your code for yourself or others.

□ CSS Notes:

□ Topic 1: CSS Comments

- **Definition:** Used to annotate code. Ignored by browsers.

- **Code Snippet:**

```
/*cascading style sheets*/
```

- **Output:**

  No output.

  Good for explaining sections within CSS files.

□ Topic 2: Universal Selector *

- **Definition:** Targets all elements on the page.
- **Code Snippet:**

```
*{
   margin: 0;
   padding: 0;
   box-sizing: border-box;
}
```

- **Output Explanation:**

  Removes default margins and paddings of all HTML elements.

  Sets box-sizing: border-box so width/height include padding and borders, making layouts consistent.

□ Topic 3: html, body Selectors

- **Definition:** Targets html and body tags together to define base height and width.
- **Code Snippet:**

```
html, body{
   height: 100%;
   width: 100%;
}
```

- **Output Explanation:**

  Sets both to occupy full viewport height and width.

  Essential for full-page layouts.

□ Topic 4: ID Selector #real

- **Definition:** Targets element with id="real".

- **Code Snippet:**

```
#real{
    background-color: blue;
}
```

- **Output:**

  Any element with id="real" will have a blue background.

□ Topic 5: Class Selector .a

- **Definition:** Targets elements with class="a".
- **Code Snippet:**

```
.a{
    background-color: royalblue;
}
```

- **Output:**

  Changes background colour to royal blue for these elements.

□ Topic 6: Div Styling Example

- **Definition:** Styling a div by id.
- **Code Snippet (commented but important):**

```
#box{
    height: 200px;
    width: 200px;
    background-color: crimson;
    margin-left: 100px;
    margin-top: 50px;
    padding-left: 20px;
    padding-top: 20px;
}
```

- **Output:**

  A crimson box of size 200x200px.

  Positioned 100px from left and 50px from top.

  Content inside the div has 20px padding from left and top.

□ Topic 7: Nested Div Styling Example

- **Definition:** Styling a parent and child div.
- **Code Snippet (commented but present):**

```
#parent{
   height: 200px;
   width: 200px;
   background-color: crimson;
}
#child{
   height: 50%;
   width: 50%;
   background-color: royalblue;
}
```

- **Output:**

  Parent: crimson box of 200x200px.

  Child: royal blue box of 100x100px inside parent.

□ Topic 8: Button Styling with Hover Effect

- **Definition:** Styles a button and changes its look on hover.
- **Code Snippet (commented but present):**

```
button{
   margin: 40px;
   font-size: 20px;
   padding: 8px;
   background-color: lightgreen;
   color: white;
   font-weight: 300px;
   /*border: 5px solid red;*/
   /*border: 5px dashed red;*/
   /*border: 5px dotted red;*/
   border: none;
   border-radius: 10px;
}
button:hover{
   background-color: grey;
   color: black;
}
```

- **Output:**

  Button with:

  Light green background

  White text

  Rounded corners (10px radius)

  On hover:

  Background turns grey

  Text turns black

□ Topic 9: Positioning (position: absolute)

- **Definition:** Positions an element absolutely within its parent or the page.
- **Code Snippet (commented but valuable):**

```
#box1{
   height: 200px;
   width: 200px;
   background-color: crimson;
   border: 2px solid white;
   margin-top: 20px;
   position: absolute;
   top: 50%;
   left: 50%;
   transform: translate(-50%, -50%);
   z-index: 9;
}
#box2{
   height: 200px;
   width: 200px;
   background-color: royalblue;
   border: 2px solid white;
   top: 50%;
   left: 50%;
   position: absolute;
   z-index: 10;
}
```

- **Output:**

  #box1: Crimson square centered on the page. z-index: 9 (behind #box2).

  #box2: Royal blue square centered and overlapping #box1. z-index: 10 (on top).

□ Topic 10: Flexbox Layout

- **Definition:** CSS layout model for arranging elements in rows or columns with flexibility.
- **Code Snippet:**

```
#parent{
   height: 100%;
   width: 100%;
   background-color: black;
   display: flex; /* activates flexbox */
   align-items: center; /* vertically centers children */
   justify-content: space-around; /* spaces out children evenly with space around them */
}
```

- **Output:**

  A black full-page container with its children arranged horizontally, evenly spaced and vertically centered.

□ Topic 11: Child Div Styling with Background Images

- **Definition:** Styles children with specific images as background.
- **Code Snippet:**

```
.child{
   height: 200px;
   width: 200px;
   background-color: crimson;
   border: 3px solid white;
   background-image: url(https://plus.unsplash.com/premium_photo-1675337267945-
3b2fff5344a0?q=80&w=764&auto=format&fit=crop);
   background-size: cover;
   background-position: center;
}
#child1{
   background-image: url(https://images.unsplash.com/photo-1743445888873-
7b989699663d?w=1000&auto=format&fit=crop);
   background-size: cover;
   background-position: center;
}
#child3{
   background-image: url(https://images.unsplash.com/photo-1745487954749-
a33270b757de?w=1000&auto=format&fit=crop);
   background-size: cover;
   background-position: center;
```

```
}
```

- **Output:**
  Each .child div: 200x200px with a default background image.
  #child1 and #child3 override the image with their specific URLs.
  Images cover the div and are centered.

□ Topic 12: Navigation Bar Styling

- **Definition:** Styles navigation bars and links.
- **Code Snippet:**

```
nav{
    padding: 60px 112px;
    width: 100%;
    display: flex;
    align-items: center;
    justify-content: space-between;
}
nav .part2{
    display: flex;
    align-items: center;
    justify-content: center;
    gap: 100px;
}
nav h2{
    font-size: 20px;
    font-weight: 700;
}
nav a{
    font-size: 15px;
    color: black;
    text-decoration: none;
    font-weight: 500;
}
a:hover{
    background-color: grey;
    padding: 4px;
    border-radius: 10px;
    font-size: 15px;
    color: black;
    text-decoration: none;
    font-weight: 500;
```

```
}
```

- **Output:**

  Navigation bar with:

  Title (h2) on left, links spaced out on right.

  Links change background to grey with padding and rounded corners on hover.

□ Topic 13: Hero Text Styling

- **Definition:** Styles large heading sections with images or background colours.
- **Code Snippet (partial):**

```
.hero-text1 h1{
    font-size: 180px;
    text-transform: uppercase;
    font-weight: 900;
    line-height: 200px;
}
.hero-text2 h1{
    font-size: 180px;
    text-transform: uppercase;
    font-weight: 900;
    line-height: 2px;
}
```

- **Output:**

  Large bold headings ("Digitize", "Ideas") appear on the page in uppercase, styled as hero text.

□ Topic 14: Button and Div Hover Effects

- **Definition:** Changes button or div styles on mouse hover.
- **Code Snippet:** Already covered under Topic 8.
- **Output:**

  Improves interactivity and user feedback.

□ JavaScript Notes:

□ Topic 1: Console Statements

- **Definition:** Outputs messages to the browser console for debugging.

- **Code Snippet:**

```
console.log("hello")
console.warn("this is warning")
console.error("this is error")
```

- **Output:**

  console.log: normal log – hello

  console.warn: yellow warning – this is warning

  console.error: red error – this is error

□ Topic 2: Browser Popups – alert, confirm, prompt

- **Definition:**
  - alert: shows a popup with OK.
  - confirm: shows OK/Cancel popup, returns true/false.
  - prompt: asks user input, returns it as string.
- **Code Snippet:**

```
alert("this is alert")
confirm("are you an adult?")
prompt("enter your name")
```

- **Output:**

  alert: popup with “this is alert”

  confirm: popup with “are you an adult?” and returns true/false

  prompt: popup with “enter your name” and returns user input string

□ Topic 3: Variables (var, let, const)

- **Definition:** Stores data in memory.
- **Code Snippet:**

```
var a = 10
let b = 20
const c = 30
```

- **Output:**

  Declares variables:

var: function/global scoped, re-declarable.

let: block scoped, not re-declarable.

const: block scoped, immutable after assignment.

□ Topic 4: Arithmetic Operators

- **Definition:** Perform mathematical operations.
- **Code Snippet:**

```
var a = 10
var b = 20
console.log(a + b) // 30
console.log(a - b) // -10
console.log(a * b) // 200
console.log(a / b) // 0.5
console.log(a % b) // 10
```

- **Output:**

  Performs addition, subtraction, multiplication, division, modulus on a and b.

□ Topic 5: Comparison Operators (==, ===)

- **Definition:**
  - ==: compares values only.
  - ===: compares value and type.
- **Code Snippet:**

```
var a = 10
var b = '10'
if(a==b){
 console.log("correct")
}else{
 console.log("incorrect")
}
if(a===b){
 console.log("correct")
}else{
 console.log("incorrect")
}
```

- **Output:**

  First if prints “correct” because 10 == '10' (value matches).

  Second if prints “incorrect” because types differ (number !== string).

□ Topic 6: Data Types

- **Definition:**
  - Primitive: number, string, boolean, undefined, null, symbol, NaN
  - Reference: array, object, function
- **Code Snippet:**

```
var a = 10
var b = "string"
var c = true
var d
var e = null
console.log(a, b, c, d, e)
```

- **Output:**

  10 "string" true undefined null

□ Topic 7: Conditional Statements (if-else)

- **Definition:** Executes code blocks based on conditions.
- **Code Snippet:**

```
var age = prompt("enter your age")
if(age > 18){
 console.log("you can vote")
}else{
 console.log("you cannot vote")
}
```

- **Output:**

  If input is above 18: “you can vote”

  Else: “you cannot vote”

□ Topic 8: Loops (while, for)

- **Definition:** Repeats code blocks multiple times.

- **Code Snippet:**

```
var a = 0
while(a < 5){
 console.log("run")
 a++
}
for(var i = 0; i < 5; i++){
 console.log(i)
}
```

- **Output:**

  while: prints "run" five times.

  for: prints numbers 0 to 4.

□ Topic 9: Functions (Normal, First-Class, Arrow)

- **Definition:** Blocks of reusable code.
- **Code Snippet:**

```
function greet(a){
 console.log("good morning", a)
}
greet("deven") // good morning deven
var abc = function(){
 console.log("hello")
}
abc() // hello
var abc = () => {
 console.log("hello")
}
abc() // hello
```

- **Output:**

  Normal function prints greeting with parameter.

  First-class function stored in variable prints “hello”.

  Arrow function also prints “hello”.

□ Topic 10: Arrays and Methods (push, pop, forEach)

- **Definition:** Stores multiple values. Methods manipulate them.

- **Code Snippet:**

```
var arr = [10, 20, 30]
arr.push(40)
console.log(arr) // [10,20,30,40]
arr.pop()
console.log(arr) // [10,20,30]
arr.forEach(function(e){
  console.log('hey', e)
})
```

- **Output:**

  push: adds 40 to array.

  pop: removes last element.

  forEach: prints each value prefixed with “hey”.

□ Topic 11: Objects and Methods

- **Definition:** Key-value pairs. Functions inside objects are called methods.
- **Code Snippet:**

```
var user = {
  userName: 'john',
  age: 20,
  greet: function(){
    console.log("good morning")
    return 10
  }
}
console.log(user.userName) // john
console.log(user.age) // 20
console.log(user.greet()) // good morning 10
```

- **Output:**

  Prints username, age, runs greet method printing “good morning” and returns 10.

□ Topic 12: DOM Selection and Manipulation

- **Definition:** Access and modify HTML elements using JavaScript.
- **Code Snippet:**

```
var a = document.querySelector("h1")
a.innerHTML = "changed"
a.style.color = "brown"
a.style.backgroundColor = "royalblue"
```

- **Output:**

  Selects first `<h1>` tag.

  Changes its text to “changed”.

  Changes colour to brown, background to royalblue.

□ Topic 13: Event Listeners

- **Definition:** Executes code in response to user actions.
- **Code Snippet:**

```
var a = document.querySelector("h1")
a.addEventListener("click", function(){
 a.innerHTML = "changed"
 a.style.color = "black"
 console.log("hello")
})
```

- **Output:**

  On clicking h1:

  Text changes to “changed”.

  Colour changes to black.

  Logs “hello”.

□ Topic 14: Bulb Toggle Example (DOM + Event)

- **Definition:** Practical use of event listeners and DOM manipulation.
- **Code Snippet:**

```
var bulb = document.querySelector("#bulb")
var btn = document.querySelector("button")
var flag = 0
btn.addEventListener("click", function(){
 if(flag == 0){
   bulb.style.backgroundColor = "yellow"
   console.log("clicked")
   flag = 1
```

```
  btn.innerHTML = "on"
 }else{
  bulb.style.backgroundColor = "transparent"
  console.log("clicked again")
  flag = 0
  btn.innerHTML = "off"
 }
})
```

- **Output:**

  Toggles bulb div between yellow (on) and transparent (off) on button click.

□ Topic 15: Asynchronous JavaScript – setTimeout

- **Definition:** Runs code after a specified delay.
- **Code Snippet:**

```
console.log("hey1")
setTimeout(function(){
 console.log("hey2")
}, 2000)
console.log("hey3")
```

- **Output:**

  Prints “hey1”, “hey3” immediately.

  Prints “hey2” after 2 seconds.

□ Topic 16: Promises

- **Definition:** Handles asynchronous operations with resolve/reject.
- **Code Snippet:**

```
var ans = new Promise((res, rej)=>{
 var n = Math.floor(Math.random()*10)
 if(n<5){
  return res()
 }else{
  return rej()
 }
})
ans.then(function(){
 console.log("below")
```

```
})
.catch(function(){
 console.log("above")
})
```

- **Output:**

  Generates random number 0-9.

  If below 5: prints “below”.

  If 5 or above: prints “above”.

□ Topic 17: async/await

- **Definition:** Cleaner syntax for handling promises in async code.
- **Code Snippet:**

```
async function abcd(){
 let raw = await fetch('https://randomuser.me/api')
 let ans = await raw.json()
 console.log(ans)
}
abcd()
```

- **Output:**

  Fetches random user data from API.

  Converts response to JSON.

  Prints data to console.

□ Topic 18: Destructuring

- **Definition:** Extracts values from arrays or objects into variables.
- **Code Snippet:**

```
const user = {
 address: { zip: "90210" },
 roles: ["user", "admin"]
}
let { zip } = user.address
console.log(zip) // 90210
let [first, second] = user.roles
console.log(first, second) // user admin
```

- **Output:**

  Extracts zip from address object and roles from array.

□ Topic 19: Spread and Rest Operators

- **Definition:**
  - Spread (...): expands array/object.
  - Rest (...): collects remaining parameters.
- **Code Snippet:**

```
const names = ["alice", "john", "charlie"]
const copynames = [...names]
function abcd(a, b, c, ...random){
 console.log(a, b, c, random)
}
abcd(1,2,3,4,5,6) // 1 2 3 [4,5,6]
```

- **Output:**

  copynames copies names array.

  Rest operator collects remaining arguments into array.

□ Topic 20: Array Methods – map and filter

- **Definition:**
  - map: returns a new array by transforming each element.
  - filter: returns new array with elements passing a condition.
- **Code Snippet:**

```
const names = ["alice", "john", "charlie"]
let newarr = names.map(function(value){
 return value + " ji"
})
console.log(newarr) // ["alice ji", "john ji", "charlie ji"]
let filtered = names.filter(function(value){
 return value.startsWith("a")
})
console.log(filtered) // ["alice"]
```

- **Output:**

  map: appends “ ji” to each name.

  filter: keeps names starting with ‘a’.

□ Topic 21: Practice Questions

- **Definition:** Real-world applications of array/object methods.
- **Code Snippet Example:**

```
const users = [
  { id: 1, name: "Alice", age: 25 },
  { id: 2, name: "Bob", age: 30 },
  { id: 3, name: "Charlie", age: 35 },
]
let newarr = users.filter(function(user){
 return user.id !== 2
})
console.log(newarr)
```

- **Output:**

  Removes user with id 2.

  Prints:

  [

  { id: 1, name: "Alice", age: 25 },

  { id: 3, name: "Charlie", age: 35 }

  ]

□ JavaScript Practice Questions:

□ Question 1: Merging Two Arrays from Separate APIs into a Single List

- **Definition:** Combines two arrays using the spread operator.
- **Full Code Snippet:**

```
const names = ["alice", "john", "charlie", "david", "emma"]
const lastnames = ["al", "jo", "ch", "da", "em"]
const fullnames = [...names, ...lastnames]
console.log(fullnames)
```

- **Output Explanation:**

  Prints:

  ["alice", "john", "charlie", "david", "emma", "al", "jo", "ch", "da", "em"]

  Combines both arrays into a single merged array.

Question 2: Filtering an Array of Objects Based on a Search Query

- **Definition:** Uses filter() to select only objects matching a condition.
- **Full Code Snippet:**

```
const products = [
  { name: "Apple", type: "Electronics" },
  { name: "Banana", type: "Electronics" },
  { name: "Cherry", type: "Electronics" },
  { name: "Date", type: "Electronics" },
]
let newarr = products.filter(function(product){
 return product.type === "Electronics"
})
console.log(newarr)
```

- **Output Explanation:**

  Prints:

  [

  { name: "Apple", type: "Electronics" },

  { name: "Banana", type: "Electronics" },

  { name: "Cherry", type: "Electronics" },

  { name: "Date", type: "Electronics" }

  ]

  Filters all products with type “Electronics”.

Question 3: Mapping Over User Data to Create User Cards

- **Definition:** Uses map() to transform user data into HTML card format.
- **Full Code Snippet:**

```
const users = [
  { name: "Alice", age: 25 },
  { name: "Bob", age: 30 },
  { name: "Charlie", age: 35 },
]
```

```
let newarr = users.map(function(user){
 return `<div><h3>${user.name}</h3><h5>${user.age}</h5></div>`
})
console.log(newarr)
```

- **Output Explanation:**

  Prints:

  [

  "

Alice

25

",

"

Bob

30

",

"

Charlie

35

"

]

 Each user is converted into a card-like HTML string.

□ Question 4: Grouping an Array of Objects by a Specific Property

- **Definition:** Groups users by their role property into an object.
- **Full Code Snippet:**

```
const users = [
 { name: "Alice", age: 25, role: "admin" },
 { name: "Bob", age: 30, role: "admin"},
```

```
  { name: "Charlie", age: 35, role: "user"},
];
let obj = {}
users.forEach(function (user){
 if(obj[user.role]){
   obj[user.role].push(user)
 }
 else {
   obj[user.role] = []
   obj[user.role].push(user)
 }
})
console.log(obj)
```

- **Output Explanation:**

  Prints:

  {

  admin: [

  { name: "Alice", age: 25, role: "admin" },

  { name: "Bob", age: 30, role: "admin" }

  ],

  user: [

  { name: "Charlie", age: 35, role: "user" }

  ]

  }

  ✅ Groups users into arrays based on their roles (admin/user).

❓ Question 5: Removing or Updating a Specific Object Based on a Unique ID

- **Definition:** Uses filter() to remove an object with a particular ID.
- **Full Code Snippet:**

```
const users = [
 { id: 1, name: "Alice", age: 25 },
 { id: 2, name: "Bob", age: 30 },
 { id: 3, name: "Charlie", age: 35 },
]
let newarr = users.filter(function(user){
 return user.id !== 2
})
```

```
console.log(newarr)
```

- **Output Explanation:**

  Prints:

  [

  { id: 1, name: "Alice", age: 25 },

  { id: 3, name: "Charlie", age: 35 }

  ]

  Removes user with id: 2 (Bob) from the array.