Lec: Java Script.

Outline:

1. Fundamentals.
2. DOM Manipulation and Event Handling.
3. Form Handling.
4. Conclusion.

1. Fundamentals:

- Background:

---A lightweight scripting language.

--- Create in 1995 by Brendan Eich, originally called LiveScript.

--- Interpreted language.

--- Most modern JS interpreter uses technique called Just-In-Time(JIT) compiling to improve performance.

--- JS source code gets compiled into a faster binary format while the script is being used, so that it can run as quickly as possible.

--- Since compilation happens in runtime rather ahead-of-time, JS is considered as interpreted language.

--- Client-Side and Server-side code:

--- Client-side code runs on the user's computer, code is downloaded and run by the browser.

--- Server-side code runs on the server, then its results are downloaded and displayed in the browser.

1. Fundamentals:

- Background:

--- Ways to add JS to a document:

--- Internal JS:

---Within HTML file embed your JS code wrapping inside <script> and </script> tag.

--- External JS:

--- Write your JS code in a separate file with .js extension, and add the file using:

<script src="example.js" defer></script> in the head section.

** defer attribute tells browser to continue downloading html content and simultaneously load JS code.

** Don't forget Separation of Concern.

--- Comments in JS:

--- Single line comment: // and multi-line comment: /* */.

1. Fundamentals:

- Language Basics:

--- Declaring Variables:

--- Using keyword var or let followed by the name of variable.

ex. let name;

--- Declaration makes value of the variable “undefined”.

--- Initializing Variable:

name = “Your Name”;

--- Declaration and initialization may be done in same line.

--- Variable naming rule: /* SAME AS OTHER LANGUAGE YOU HAVE STUDIED.*/

--- Variable Type:

--- Numbers: Either integers or floating point number.

--- Not required to explicitly declare type.

--- Strings: Text wrapped in either single quotes or double quotes.

--- ex. let name = “Your Name”;

--- Boolean: true/false.

ex. let test = 6 < 3;

1. Fundamentals:

- Language Basics:

--- Variable Type:

--- Arrays: Single object with multiple values.

---ex. let marks = [70,80,87];

--- Accessing individual element: marks[0]; …

--- Object: In JS object are represented as:

let student = {Name: “Your Name”, RollNo: 42};

--- for retrieving information: person.Name;

*** Aside: JS is a “dynamically typed language”, meaning we are not required to explicitly mention the type of the variable. Browser will infer the type of the variable from the value itself.

---We can use “typeof” operator to find the type of the variable.

*** Constant in JS are declared using const keyword.

ex. const today = “Friday”;

1. Fundamentals:

- Language Basics:

--- Arithmetic Operators:

--- + , -, *, /, %, **.

--- Other operators: ++, --, =, +=,-=,*=,/=, === (Strict equality), !==(strict non-equality), <,>,<=, >=.

--- Concatenation in Strings: + is used as operator for concatenation.

--- The Number object converts a string into number and conversely every number has a method called toString(), which converts number to string.

--- Few important String methods: **ex. string, let name = "Java Script";**

--- Finding length of a string: name.length; → 11.

--- How will you retrieve the last character of the string?→

--- Finding substring: name.indexOf("Script"); → 5. (index location).

--- If string is not found it will return -1.

--- Extracting substring: name.slice(0,4); → Java.

name.slice(5); → Script.

--- Changing Case: name.toLowerCase(); and name.toUpperCase();

--- Updating parts of a string: name.replace("Java","Python"); → Python Script.

1. Fundamentals:

- Language Basics:

--- Two Special value for type: undefined and null.

--- undefined: Declared but has not yet been assigned a value.

--- null: Exists, but was specifically assigned an empty value or null. Expresses intentional a lack of identification.

--- More on Array object:

--- Creating arrays:

let books = [“Book 1”,”Book 2”, “Book 3”];

--- Array can store various data types: strings, numbers, objects and other array.

--- We can mix data types in array.

ex. let random = [“Book 4”, 42, [21,”78”,90]];

--- Adding and removing elements from end of the array: push() and pop().

--- Adding and removing elements from the beginning: unshift() and shift().

1. Fundamentals:

- Language Basics:

---Conditional statement: (Decision making statements)

--- if….else statement

```
if(conditon){
        //code;
    }
else{
      code;
 }
```

--- Can be nested.

--- logical operator: && → And, || → Or, ! →Not

--- switch…case statement:

```
switch(expression){
  case choice1:
     //code;
     break;
  case choice2:
   //code;
  break;
default:
   //code;
}
```

1. Fundamentals:

- Language Basics:

---Loop:

--- Standard For loop:

```
for(initializer; condition; final-expression){
    //code;
}
```

--- As with language, break and continue can be used with JS loops.

--- Standard while loop:

```
initializer;
while(condition){
  //code;
}
```

--- do…while loop:

```
initializer;
do{
   //code;
}while(condition);
```

1. Fundamentals:

- Language Basics:

--- Function:

Set of statements that performs a task/computation.

--- Declaration: starts with keyword function followed by name of the function, list of coma separated parameters and statements inside curly braces.

ex. function print_hello(name){

console.log("Hello " +name);

}

--- primitive parameters such as number are passed to function by value.

--- changes inside the function is not reflected globally.

--- Non-primitive value like Array or user-defined object when passed as parameter, changes in the properties done inside the function are visible globally.

--- Function Expression:

--- Can assign function to a variable anonymously or by giving it a name.

--- ex. const square = function(num){

return num ** 2;

} -→ square(4);

const factorial = function fac(num){

return num < 2 ? 1 : num * fac(num – 1);

} → console.log(factorial(3));

1. Fundamentals:

- Language Basics:

--- Function:

--- Function expression are convenient when passing a function as an argument to another function.

---ex. function map(f,a){

```
let result = [];
for(let i=0; i<a.length; i++){
    result[i] = f(a[i]);
}
return result;
}
```

--- Calling function: invoke with function name and required parameter.

** A method in JS is a function that is a property of an object.

--- Function Scope:

--- Variable defined inside the function can not be accessed from anywhere outside the function.

--- Function can access all variables and function defined inside the scope in which it is defined, i.e.

--- A function defined in global scope means:

--- Can access all variables defined in the global scope

--- Function defined inside another function can access all variables defined in its parent function, and any other variable to which parent function has access.

--- A function can be nested inside another function. The nested (inner) function is private to its containing (outer) function.

---- The inner function can be accessed only from statements in the outer function.

--- The inner function can use the arguments and variables of the outer function, while the outer function cannot use the arguments and variables of the inner function.

1. Fundamentals:

- Language Basics:

--- Function:

--- Name Conflict: Name conflict in arguments or variables are resolved using nested scope precedence i.e. inner most scope takes the highest precedence, while outer most scope takes the lowest.

--- This is called Scope-chain, i.e. first in the chain is the inner-most scope and last in the chain is the outer-most scope.

--- argument object:

--- Arguments in a function are maintained in an array-like object called arguments and can be accessed like array starting with ordinal no. 0.

--- Total number of argument is indicated by arguments.length.

--- ex. function myConCat(separator){

```
let result = '';
 for(let i=1;i<arguments.length; i++){
     result += arguments[i] + separator;
}
return result; }                                    myConCat(', ','book_1','book_2','book_3');
```

1. Fundamentals:

- Language Basics:

--- Object:

--- Collection of properties, and a property is an association between a name/key and a value.

--- A property value can be a function or any other type. It is same as any JS variable and attached to an object.

--- Property of an object can be accessed using dot notion, i.e. object_name.property_name;

--- object name and property name are case sensitive.

--- Creating Object:

--- using new keyword:

ex. let book = new Object();

book.name = “Book 1”;

book.author = “Author 1”;

book.publisher = “publisher 1”; -→ book.name;

1. Fundamentals:

- Language Basics:

--- Object:

--- Using Object Initializer:

ex. let book_1 = {

name: “Book 2”,

author: “Author 2”,

publisher: “Publisher 2”,

}

--- Properties of JS object can also be accessed or set using a bracket notation.

--- book_1[‘name’] = ‘Introduction to Java Script’;

--- Object in JS are also called Associative Array, since each property is associated with a string value that can be used to access it.

--- An Object property name can be any valid JS string or anything that can be converted to string.

--- Identical object initializers create distinct object that will not compare to each other as equal.

1. Fundamentals:

- Language Basics:

--- Object:

--- Using Constructor function:

--- Define the object type by writing a constructor function.

--- Create an instance of the object with new.

--- ex. function Book(name,author,publisher){

this.name = name;

this.author = author;

this.publisher = publisher;

}

book = new Book(“Java Script”, “Author”,”MIT Press”);

*** An object can have a property that itself is an object.

2. DOM Manipulation and Event Handling:

- Standard JS Way of handling Dynamics of a Web Application:
  1. Find the Element.
  2. Get the Dynamic behaviour and on What (Event Handling and Event).
  3. Add the behaviour to the element.

  --- In other words,

  1. Link a JS program to your HTML.
  2. Identify the elements you want to attach user/page events.
  3. Identify the events you want to respond to.
  4. Identify what each response function is.
  5. Assign to the Listener the identified response function as callback.

2. DOM Manipulation and Event Handling:

- Use cases of Events in JS:

  1. User hovers the cursor over a certain element.
  2. User Chooses a key on the keyboard.
  3. Page Loaded.
  4. Form Submitted.
  5. Video is played, paused or finished.
  6. Error occurs.

  …

- Each event has an event handler, which is a block of code, that executes when event occurs.

  --- When such a block of code is defined to run in response to an event, it is called registering an event handler.

2. DOM Manipulation and Event Handling:

- Ways to Adding Event Handler:

  1. Using Event Handler Property:

     -- ex. const btn = document.getElementById(“idButton”);

     btn.onclick = function(){

     //code;

     }

  2. Inline Event Handler: (Aside: Don’t use them):

     --- <button onclick = “doSomething()”>Click</Button>

     function doSomething(){

     //code;

     }

2. DOM Manipulation and Event Handling:

- Ways to Adding Event Handler:

  3. addEventListener() and removeEventListener():

  window.addEventListener(“load”,getDateAndTime);

  --- getDateAndTime is a function called Callbacks, i.e. we are passing function as value to another function.

  --- load is a kind of event that is associated with window global object.

  --- newest type of event handling mechanism, defined DOM level 2 specification.

  --- remomeEventListener(), removes a previously added listener.

  --- window.removeEventListener(“load”,getDateAndTime);

  → This is can improve efficiency for large programs by cleaning unused event handlers.

  → Allows to use same widget for performing different actions in different circumstances, by adding or removing event handler.

2. DOM Manipulation and Event Handling:

- Common Events:

  --- Mouse Events:

Event	Description
click (mousedown and mouseup)	Fires when the mouse is pressed and released on an element.
dbclick	Fires when an element is clicked twice.
mouseenter	Fires when a pointer enters an element.
mouseleave	Fires when a pointer leaves an element.
mousemove	Fires every time a pointer moves inside an element.

2. DOM Manipulation and Event Handling:

- Common Events:

--- Form Events:

Event	Description
submit	Fires when a form is submitted.
focus	Fires when an element (such as an input) receives focus.
Blur	Fires when an element loses focus.

--- Keyboard Event:

Event	Description
keydown	Fires once when a key is pressed.
Keyup	Fires once when a key is released.
keypress	Fires continuously while a key is pressed.

2. DOM Manipulation and Event Handling:

- Common Events:

--- Form Events:

Three important properties related to keyboard event can be accessed if additional event object is passed.

Property	Description	Ex.
keyCode	A number pertaining to the key.	65.
key	Represents the character name	A
code	Represents the physical key being pressed.	Key A.

--- ex. document.addEventListener("keydown",event => {

console.log('key: '+event.keyCode);

console.log('key: '+event.key);

console.log('code: '+event.code);

});

2. DOM Manipulation and Event Handling:

- Accessing DOM Element:

--- Can be accessed by ID, Class and Query Selector.

--- Accessing by ID: Access single element in the DOM by its unique ID with getElementById() method of the document object.

--- ex. var para = document.getElementById(“date”);

--- Drawback: Since ID is unique, so can able to access a single element at a time.

---Accessing by Class:

--- Method: document.getElementByClassName();

--- If a class is attached to multiple element then we have to access with index notation.

ex– var classVar = document.getElementByClassName();

for(var i=0;i<classVar.length;i++)

classVar[i].style.color = “red”;

2. DOM Manipulation and Event Handling:

- Accessing DOM Element:

  --- Query Selectors:

  --- Methods : document.querySelector(); → For accessing Single Element.

  document.querySelectorAll(); -→ For accessing Multiple elements.

  --- If selector is ID, then we prefix the query with # in parameter and if it is class we prefix it with dot.

  --- ex. var var1 = document.quearySelector(#demoQuery);

  var var2 = document.querySelectorAll(.demQueryAll);

- Traversing DOM Element:

  --- document object is the root of every node in DOM. It is a property of the window global object.

  --- Nodes in the DOM are refereed to as parents, children, and sibling depending their relationship with other element. (REMEMBER: parent-child relationship).

2. DOM Manipulation and Event Handling:

- Accessing DOM Element:

Property	Meaning
parentNode	Parent Node.
parentElement	Parent Element Node.
childNodes	Child Nodes.
firstChild	First Child Node.
lastChild.	Last Child Node.
children	Element Child Nodes.
firstElementChild	First Child Element Node
lastElementChild	Last Child Element Node.
previousSibling	Previous sibling Node.
nextSibling	Next Sibling Node.
previousElementSibling	Previous Sibling Element Node.
nextElementSibling	Next Sibling Element Node.

--- ex.

```
for(var i of p.childNodes){
    console.log(i);
 }
```

2. DOM Manipulation and Event Handling:

- Making Changes in DOM: (Adding, Removing, Replacing Nodes)

--- Creating New Nodes:

Methods	Description
createElement(“tag”)	Creates and returns a new empty DOM node representing an element. --- var newElem = document.createElement(“p”); newElem.textContent = “New Paragraph Component.”

--- Inserting Nodes into the DOM:

Methods	Description
node. appendChild()	Add a node as the last child of a parent element.
node.insertBefore()	Insert a node into the parent element before a specified sibling node.
Node.replaceChild()	Replace an existing node with a new node.

2. DOM Manipulation and Event Handling:

- Making Changes in DOM: (Adding, Removing, Replacing Nodes)

--- Removing Nodes:

Methods	Description
node.removeChild()	Remove child node.
node.remove()	Remove node.

*** Your Work: Read about Manipulating Classes.

2.DOM Manipulation and Event Handling:

2.1. Introduction to jQuery:

--- jQuery is a JS library, used for writing common JS tasks more concisely.

--- jQuery can be included in two ways in your project:

--- Downloading and including as a script just like regular JS file.

--- Link to a file via Content Delivery Network (CDN).

--- jQuery code are wrapped inside:

```
$(document).ready(function(){
    //Code;
});
```

--- jQuery will detect this state of readiness so that code included inside this function will run once the DOM is ready for JS code to execute.

--- Syntax:

```
$("selector").method();
```

--- ex. $(".jQtest").html("Hello from jQuery.");

2.DOM Manipulation and Event Handling:

2.1. Introduction to jQuery:

---jQuery Selector:

--- Selectors are how we tell jQuery which elements we want to work on. To access a selector we use $ followed by parenthesis.

--- Commonly used selectors:

--- $(“*”) : Wildcard, selects every elements on the page.

--- $(“this”): Current, selects the current element being operated on within a function.

--- $(“p”): Tag, selects every instance of the <p> tag.

--- $(“.example”): Class: selects every element that has the example class applied to it.

--- $(“#example”): Id, selects a single instance of the unique example id.

--- $(“[type=‘text’]”): Attribute, selects any element with text applied to the type.

--- $(“p:first-of-type”): Pseudo Element, selects the first <p>.

2.DOM Manipulation and Event Handling:

2.1. Introduction to jQuery:

--- Events in jQuery:

Ex. Case: $(document).ready(

```
$(“#btn”).click(function(){
    $(“#jQtest”).html(“Hello again.”);
}); );
```

--- Commonly used jQuery events are:

Event	Description
click()	Executes on a single mouse click.
hover()	Executes when the mouse is hovered over an element. mouseenter() and mouseleave() apply to the mouse entering or leaving an element, respectively.
submit()	Executes when a form is submitted.
scroll()	Executes when the screen is scrolled.
keydown()	Executes when you press down on a key on the keyboard.