HTTP	HTTPS
Full form:Hyper Text Transfer Protocol	Hyper Text Transfer Protocol Secure
Start with:http://	https://
Port no:80	443
Unsecure	Secure
Not encrypted	Encrypted
Runs:Appln layer	Transport layer
Doesnot require Certificate	SSL certificate
Faster	Slower

SSL	TLS
Full form:Secure Socket Layer	Transport Layer Security
Message Authentication Code protocol	Hashed Message Authentication Code protocol
Complex	Simple
Less Secured	More Secured
Less reliable and slower	More Reliable and less latency
uses port to set up explicit connection	uses protocol to set up implicit connection

JSON	XML
Full form:Javascript Object Notation	Extensible Markup Language
Doesnot support Namespace	Supports Namespace
Supports Array	Doesnot support Array
Easy to read	Difficult to Read
Doesnot use end tag	Uses Start tag and end tag
Less secured	More secured

URI, URL, and URN

URI (Uniform Resource Identifier):

- A URI is a generic term used to identify any resource, whether it is located on the internet or elsewhere. It is a string of characters that provides a way to uniquely identify a resource.
- URIs serve as a fundamental concept in web architecture and are used to identify resources such as web pages, images, files, and more.
- URIs can be further divided into two subcategories: URLs and URNs.

URL (Uniform Resource Locator):

- A URL is a specific type of URI that provides not only a means of identifying a resource but also a way to locate it on the internet. It includes information about how to access the resource, such as the protocol (e.g., http:// or ftp://), the domain name (or IP address), and the path to the resource on a server.
- Example URL: https://www.example.com/index.html
- URLs are commonly used when you want to specify the location of a web resource, such as a web page or an image.

URN (Uniform Resource Name):

- A URN is another type of URI that is used to provide a unique and persistent name for a resource without specifying its location or how to access it. URNs are meant to serve as a long-lasting and stable identifier.
- Unlike URLs, URNs do not change even if the resource is moved to a different location or if the method of access changes.
- Example URN: urn:isbn:0451450523 (used to uniquely identify a book by its ISBN)

REST API

-Stateless

-Client-server

-layered system

1. **Explain alert(), confirm() and prompt() method of window object:**

- `alert()`: Displays a dialog box with a message and an "OK" button. It is used to provide information to the user.

- `confirm()`: Displays a dialog box with a message and "OK" and "Cancel" buttons. It is used to get a yes/no response from the user.

- `prompt()`: Displays a dialog box with a message, an input field, and "OK" and "Cancel" buttons. It is used to get input from the user.

```
// Alert
alert("This is an alert message.");


// Confirm
var result = confirm("Do you want to proceed?");
if (result) {
    console.log("User clicked OK.");
} else {
    console.log("User clicked Cancel.");
}


// Prompt
var name = prompt("Please enter your name:");
console.log("Hello, " + name);
```

2. **Explain output and input methods in JavaScript:**

- Output methods are used to display information to the user, such as `console.log()` for logging to the console or `alert()` for displaying pop-up messages.

- Input methods are used to get input from the user, such as reading user input from forms or using `prompt()` to request input.

3. **What are functions? Explain how to create and call functions:**

- Functions in JavaScript are reusable blocks of code that perform a specific task.

- To create a function, use the `function` keyword followed by a name, parameters, and a code block. Example: `function add(a, b) { return a + b; }`

- To call a function, use its name followed by parentheses and pass any required arguments. Example: `var result = add(2, 3);`

4. **Explain various data types used in JavaScript:**

- JavaScript has several data types, including numbers, strings, booleans, objects, arrays, functions, null, and undefined.

5. **Different ways of declaring variables in JavaScript (let, var, const):**

- `var`: Used for declaring variables with function or global scope.

- `let`: Used for declaring block-scoped variables.

- `const`: Used for declaring constants, which cannot be reassigned.

6. **What is an event handler? Explain any three mouse events:**

- An event handler is a function that responds to specific events triggered by user actions or the browser.

- **Mouse events:**

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

- Three mouse events:

  - `click`: Occurs when the mouse is clicked on an element.

  - `mouseover`: Occurs when the mouse pointer enters an element.

  - `mouseout`: Occurs when the mouse pointer leaves an element.

```
<button id="myButton">Click Me</button>
<script>
  // Event Handler for Click
  document.getElementById("myButton").addEventListener("click", function() {
    alert("Button clicked!");
  });
  // Event Handlers for Mouse Over and Mouse Out
  var element = document.getElementById("myElement");
  element.addEventListener("mouseover", function() {
    console.log("Mouse over element.");
  });
  element.addEventListener("mouseout", function() {
    console.log("Mouse out of element.");
  });
</script>
```

7. **Different types of events and event handlers**:**

  - Events include keyboard events, form events, window events, and more.

  - Event handlers are functions that respond to specific events. Examples include `onclick`, `onsubmit`, and `onload`.

8. **What do you mean by an object in JavaScript? Different ways of creating objects:**

  - An object is a collection of key-value pairs, where each key is a string (or symbol) and each value can be any data type.

  - Objects can be created using object literals, constructors (e.g., `new Object()`), or with classes (ES6).

9. **Explain Object-Oriented concepts in JavaScript**:**

- JavaScript is a prototype-based language, not a class-based one. It uses prototypes for inheritance.

- Objects can inherit properties and methods from other objects through their prototypes.

10. **Explain advantages of using Arrow functions:**

- Arrow functions have a shorter syntax.

- They capture the value of `this` from the surrounding context.

- They are often used for simple, concise function expressions.

11. **Explain generator function and how it is different from a normal function:**

- A generator function is a special type of function in JavaScript that can pause its execution and resume it later.

- It is created using the `function*` syntax and uses `yield` to pause and return values.

- Unlike regular functions, generator functions can be paused and resumed, allowing for asynchronous-like behavior without callbacks or promises.

12. **Synchronous and Asynchronous Programming:**

- Synchronous programming executes code line by line in a sequential manner.

```
console.log("Start");
for (let i = 0; i < 3; i++) {
  console.log(i);
}
console.log("End");
```

- **Asynchronous programming** allows tasks to run independently and concurrently, often using callbacks, promises, or async/await to manage asynchronous operations.

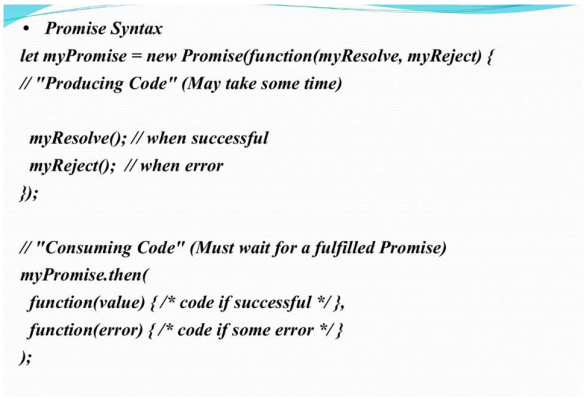
```
console.log("Start");
setTimeout(function() {
  console.log("Timeout");
}, 1000);
console.log("End");
```

A Promise is in one of these states:

pending: initial state, neither fulfilled nor rejected.

fulfilled: meaning that the operation was completed successfully.

rejected: meaning that the operation failed.



- *Promise Syntax*

```
let myPromise = new Promise(function(myResolve, myReject) {
// "Producing Code" (May take some time)

  myResolve(); // when successful
  myReject();  // when error
});

// "Consuming Code" (Must wait for a fulfilled Promise)
myPromise.then(
  function(value) { /* code if successful */ },
  function(error) { /* code if some error */ }
);
```

Data validation in JavaScript is a crucial aspect of web development to ensure that the data entered or processed by users is correct, secure, and adheres to specific rules or patterns. Regular expressions (regex or RegExp) are a powerful tool in JavaScript for data validation because they allow you to **define patterns that data must match.**

Creating a Regular Expression:

```
const emailPattern = /^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$/;
```

Testing for a Match:

```
const isValidEmail = emailPattern.test("example@email.com");
```

Matching and Extracting Data:

```
const text = "This is a phone number: 123-456-7890";

const phoneNumbers = text.match(/\d{3}-\d{3}-\d{4}/g);

console.log(phoneNumbers); // ["123-456-7890"]
```

let	var	const
Block scoped	Global or local scoped	Block scoped
Reassignment of value	yes	no
no	Variables can be hoisted	no
Eg Let a =10;	Eg var a=20	Eg Const pi=3.14;

Generator vs. Iterator:

Generator:

A generator is a special type of function in JavaScript defined using the **function* syntax.** It can yield multiple values over time using the **yield** keyword.

Generators are iterators themselves and provide a convenient way to create iterators.

They can be **paused and resumed**, allowing you to control the flow of execution.

Generators are often used for **asynchronous programming** and working with data streams.

```
function* generatorFunction() {
  yield 1;
  yield 2;
  yield 3;
}
const generator = generatorFunction();
```

Iterator:

An iterator is an object with a **next()** method that returns an object with two properties: value (the next value in the sequence) and done (a boolean indicating whether the iteration is complete).

Iterators are typically used to **traverse data structures** or sequences, such as **arrays** or custom collections.

You can create custom iterators by implementing the iterator protocol (defining a next() method).

Write Javascript code that will change the background color of the page when user clicks on the particular button.

```
<head>
   <title>Change Background Color</title>
</head>
<body>
   <button id="changeColorButton">Change Color</button>
<script>
      var changeColorButton = document.getElementById("changeColorButton");
     // Define a function to change the background color
      function changeBackgroundColor() {
            document.body.style.backgroundColor = red;
      }
 // Attach a click event listener to the button
      changeColorButton.addEventListener("click", changeBackgroundColor);
   </script>
</body>
</html>
```

Write Javascript code to calculate gross salary whare salary details has been accepted using form.

```
<html>
	<head>
		<title>Salary App</title>
		<style>body{font-size:20px;}</style>
		<script>
		function grossSalary(){
			event.preventDefault();
			let b=document.getElementById("basic");
			let a=document.getElementById("allowances");
			let d=document.getElementById("deductions");
			let ans=document.getElementById("ans");
			gross=parseFloat(b.value)+parseFloat(a.value)-parseFloat(d.value);
			ans.innerHTML="Gross Salary = "+gross;
		}
		</script>
	</head>
	<body>
	<center>
		<h1>Enter your salary details</h1>
		<form onsubmit="grossSalary()">
		<input type="number" id="basic" placeholder="enter salary"/>
		<br/><br/>
		<input type="number" id="allowances" placeholder="enter allowance"/>
		<br/><br/>
		<input type="number" id="deductions" placeholder="enter deductions"/>
		<br/><br/>
		<input type="submit" value="calculate gross salary"/>
		<h1 id="ans"></h1>
		</form>
	</center>
	</body>
</html>
```

Write a javascript code to find sum of N natural numbers using user defined function.

```
let num=prompt("enter a number")

let ans=0

for (let i=1;i<=num;i++){

   ans+=i;

}

console.log(ans);
```

REACT:

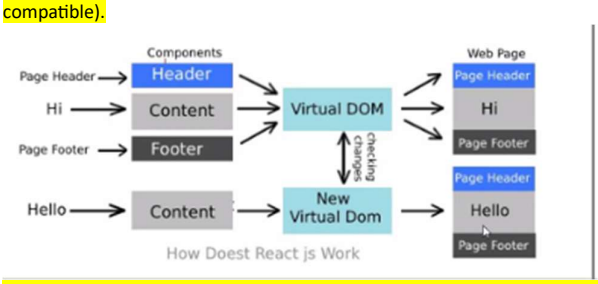
1. **What is react? Explain the Features of React.**
   ⇨ Features of React:
   JSX (JavaScript Syntax Extension)
   Virtual DOM
   One-way data binding
   Performance
   Extensions
   Conditional statements
   Components
   Simplicity

2. **What is JSX? Explain how JSX is processed by Browser.**
   JSX(JavaScript Syntax Extension): JSX is a combination of HTML and JavaScript. You can embed JavaScript objects inside the HTML elements. JSX is not supported by the browsers, as a result Babel compiler transcompile the code into JavaScript code. JSX makes codes easy and understandable. It is easy to learn if you know HTML and JavaScript.

   ```
   const name="GeekforGeeks";
   const ele = <h1>Welcome to {name}</h1>;
   ```

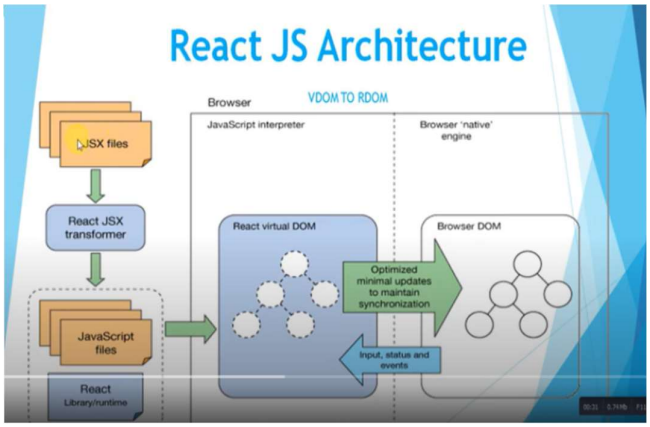
3. **What is Virtual DOM? How it is different from Real DOM?**
   In simple words, virtual DOM is just a copy of the original DOM kept in the memory and synced with the real DOM by libraries such as ReactDOM.

   This process is called Reconciliation(the process of making consistent or compatible).

   We have seen that the re-rendering of the UI is the most expensive part and React manages to do this most efficiently by ensuring that the Real DOM receives batch updates to re-render the UI.

4. **The architecture of React.**



5. **Folder Structure of React application.**

- ***Below is how the project will be structured:***

```
my-sample-app
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── build
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── serviceWorker.js
```

⇨ **README.md** is a markdown file that includes a lot of helpful tips and links that can help you while learning to use Create React App.

⇨ **node_modules** is a folder that includes all of the dependency-related code that

⇨ **package.json** that manages our app dependencies and what is included in our node_modules folder for our project, plus the scripts we need to run our app.

⇨ **.gitignore** is a file that is used to exclude files and folders from being tracked by Git. We don't want to include large folders such as the node_modules folder

⇨ **public** is a folder that we can use to store our **static asset**s, such as images, svgs, and fonts for our React app.

⇨ **src** is a folder that contains our source code.
It is where all of our React-related code will live and is what we will primarily work in to build our app.

⇨ **Create React App**

⇨ **npx create-react-app my-react-app**

⇨ create-react-app will set up everything you need to run a React application.

⇨ **Run the React Application**

⇨ Run this command to move to the my-react-app directory:

⇨ **cd my-react-app**

⇨ Run this command to execute the React application my-react-app:

⇨ **npm start**

⇨ A new browser window will pop up with your newly created React App

6. **What are the components in React? Explain different types of Components?**

React is a popular JavaScript library for building user interfaces, and it's based on the concept of components. Components are the building blocks of a React application, and they encapsulate a part of the user interface's functionality. React components can be categorized into two main types: functional components and class components.

Functional Components:

- Functional components are also known as stateless or presentational components.
- They are simple JavaScript functions that return JSX (JavaScript XML) to describe what should be rendered.
- These components are primarily responsible for the UI and presentation of data.

Class Components:

- Class components are also known as stateful components.
- They are defined as JavaScript classes that extend the React.Component class.
- Class components have their own internal state, allowing them to manage and react to changes in state.

They don't have their own internal state (stateless), making them more straightforward and easier to test and maintain.

7. **What is the use of render() in React?**

In React, the render() method is a fundamental and required method in class components. It plays a crucial role in determining what should be displayed on the user interface. Here's the main purpose and use of the render() method:

Eg:

```
function save(){
        return(
        <>
        <h1>Hello</h1>
        </>
        );
}
```

UI Rendering:

The primary purpose of the **render() method is to return JSX (JavaScript XML)** that describes the UI elements that the component should render.

JSX is a syntax extension for JavaScript that allows you to **write HTML-like code within your JavaScript code.** React uses JSX to define the structure and appearance of components.

The render() method should return a single root element (e.g., a div) that contains all the elements you want to render. This root element can have multiple child elements.

n summary, the render() method is a crucial part of React class components, responsible for defining what the component should display based on its current state and props.