1. Difference between HTTP and HTTPS.

HTTP: An application-layer protocol introduced to enable communication between web browsers (clients) and web servers. It transmits data as plain text, making it visible to anyone who intercepts the connection.
HTTPS: A more secure version of HTTP that uses SSL/TLS (Secure Sockets Layer/Transport Layer Security) to encrypt the connection between the browser and the server. It is now the modern standard for all websites to protect user privacy.

Primary Differences:

Security and Encryption:
HTTP transmits data in plain text. This means any information sent, such as passwords or credit card numbers, can be intercepted and read easily by anyone monitoring the network.
HTTPS uses SSL/TLS encryption to scramble data into unreadable ciphertext before it is sent. Even if intercepted, the data is useless to hackers without the private decryption key.

Authentication and Trust:
HTTP does not verify the identity of the server. You have no guarantee that the website you are visiting is the real one and not a malicious spoof.
HTTPS requires an SSL/TLS certificate issued by a trusted Certificate Authority. This certificate confirms the website's authenticity, preventing impersonation attacks.

Browser Indicators and Warnings:
HTTP websites are now explicitly marked as "Not Secure" by modern browsers like Chrome and Firefox, often accompanied by a red warning icon or a "!" symbol.
HTTPS websites display a padlock icon in the address bar, providing a visual signal that the connection is private and secure.

Technical Communication (Ports):
HTTP uses Port 80 by default for all data transfers.
HTTPS uses Port 443, which is specifically designated for secure, encrypted traffic.

Performance and Speed:
HTTP was traditionally faster because it skipped the "handshake" process required for encryption.
HTTPS is faster in 2026. This is because modern, high-performance protocols like HTTP/2 and HTTP/3 (which support features like multiplexing) generally require an encrypted HTTPS connection to work.

Search Engine Optimization (SEO):
HTTP provides no benefit for search visibility and can even lead to lower rankings.
HTTPS is a ranking factor for search engines like Google, meaning secure sites are prioritized and more likely to appear higher in search results.

2. Define web architecture and DNS.

Web architecture:
Web Architecture is the conceptual framework and logical structure of a web application. It defines how the various software components—such as the browser, the server, and the database—interact with one another to deliver a website to a user. It's about organizing information and systems for seamless user experience and performance, encompassing.
everything from design to underlying infrastructure.

DNS (Domain Name System):
DNS is a hierarchical, decentralized naming system that serves as the "phonebook of the internet." Its sole purpose is to translate human-friendly domain names (like `google.com`) into numerical IP addresses (like `142.250.190.46`)
Because computers and routers communicate using numbers, they cannot understand "google.com" on their own. DNS acts as a massive global database that looks up the name you typed and provides your browser with the specific IP address of the server where that website's architecture lives.

3. Difference between ARIA and WAI ARIA.

There is no difference between ARIA and WAI-ARIA,both are the same.WAI-ARIA is the full name of ARIA. ARIA stands for Accessible Rich Internet Applications and WAI-ARIA stands for Web Accessibility Initiative – Accessible Rich Internet Applications.

4. How arithmetic operators and concatenation works in the browser?

When you use arithmetic operators in the browser:

The browser reads your JavaScript code inside `<script>` or linked `.js` file.

It evaluates the expressions (+,-,*,/,%,++,--)

The result is computed in memory.

If you use `console.log()` or update HTML with the result, the browser displays it on the screen.

Example:

```
let a = 10;
let b = 5;
console.log(a + b); // Addition
console.log(a - b); // Subtraction
console.log(a * b); // Multiplication
console.log(a / b); // Division
```

Concatenation:

When + is used with strings, the browser:

Detects that one or both operands are strings.

Joins the values as text instead of doing math.

Creates a new string in memory.

If displayed in HTML or console, it shows the combined string.

Example:

```
<script>
let firstName = "Nishal";
let lastName = "Reddy";
let age = 20;
console.log(firstName + " " + lastName); // Nishal Reddy
console.log("Age: " + age);              // Age: 20
</script>
```

5. Define CDN.

CDN (Content Delivery Network) is a system of multiple servers distributed across different locations worldwide.
It delivers website content from the server closest to the user.
CDN improves website loading speed and performance.
It reduces latency and network congestion.
CDN decreases the load on the main server.
It helps handle high traffic efficiently.
CDN improves website reliability and uptime.
It also provides basic security like DDoS protection.

6. Explain what is the smallest js file which you can execute on the web.

JavaScript does not require code to produce output to be executable.
Technically the smallest executable file is just a semi colon (;).

7. Behind the screen how the web works.

You enter a URL in the browser (e.g., `www.google.com`).

DNS lookup happens: the browser asks DNS servers to convert the domain name into an IP address.

Browser sends an HTTP/HTTPS request to the server using that IP address.

The server processes the request (runs backend code, accesses databases if needed).

The server sends a response (HTML, CSS, JavaScript, images).

The browser receives the files and starts parsing them.

HTML builds the structure of the page (DOM).

CSS styles the page (layout, colors, fonts).

JavaScript runs to add interactivity (events, animations, API calls).

The page is rendered on the screen, and user interactions trigger new requests.

8.Types of DOMs in React

1. Real DOM
● The browser's native representation of HTML elements.
● Direct manipulation is slow because the entire tree may need re-rendering.

2. Virtual DOM
● A lightweight JavaScript object tree that mirrors the Real DOM.
● React updates this first, then efficiently reconciles changes into the Real DOM.
● Improves performance by minimizing costly reflows.

3. Shadow DOM
● A web standard used in Web Components, not unique to React
● Encapsulates styles and markup so they don't leak into or out of a component.
● Example: or elements internally use Shadow DOM to isolate their implementation.
● In React, you rarely interact with it directly, but it's important when integrating with Web Components.

4. Shallow DOM (Shallow Rendering)
● Not a DOM type per se, but a testing technique.
● With libraries like Enzyme, shallow rendering creates a lightweight representation of a component without rendering its children.
● Useful for unit testing a component in isolation.
● Example: Testing without worrying about the inside.

5. Fiber DOM (React Fiber)
● React's internal implementation of the Virtual DOM.
● Fiber is a tree-like structure that enables React to: ○ Break rendering work into chunks. ○ Prioritize updates (e.g., animations vs. data fetching). ○ Pause and resume rendering for smoother performance.
● This is why React can handle complex UIs without blocking the browser.

9.How does asynchronous work in node.js?

Node.js is single-threaded, but it can handle many tasks at the same time using asynchronous (non-blocking) execution.

How it actually works:

1. Call Stack – runs your normal JS code.
2. Async task starts (file read, timer, API call).
3. That task is sent to the background / OS.
4. When it finishes, its callback / promise goes to the Event Queue.
5. Event Loop pushes it back to the Call Stack when JS is free.

Simple example:

```
console.log("Start");
setTimeout(() => {
  console.log("Async task done");
}, 2000);
console.log("End");
```

10.Identify the differ between spread and rest operator along with the use case

Spread Operator:

Purpose: *Expands* element

Used when: You want to copy, merge, or pass values

Use cases:

- Copy arrays/objects
- Merge arrays/objects
- Pass array elements as function arguments

Example:

```
const arr1 = [1, 2];

const arr2 = [...arr1, 3, 4];

console.log(arr2); // [1, 2, 3, 4]

const user = { name: "johnl" };

const updatedUser = { ...user, age: 21 };
```

Rest Operator:

Purpose: *Collects* elements
Used when: You want to group remaining values

Use cases:

- Handle multiple function arguments
- Destructure remaining values

Example:

```
function sum(...numbers) {
  return numbers.reduce((a, b) => a + b);
}
console.log(sum(1, 2, 3)); // 6
const [first, ...rest] = [10, 20, 30, 40];
console.log(rest); // [20, 30, 40]
```

11.Arc of v8 engine in nodejs explain briefly

Architecture (ARC) of V8 Engine in Node.js:

The V8 engine is JavaScript’s runtime engine used by Node.js to execute JS code

Main Components of V8:

1. Parser

- Reads JavaScript code
- Converts it into Abstract Syntax Tree (AST)

2. Ignition (Interpreter)

- Converts AST into bytecode
- Starts executing the code quickly

3. TurboFan (Compiler)

- Optimizes frequently used code
- Converts bytecode into machine code
- Makes execution faster

4. Heap

- Memory area where objects & variables are stored
- Managed by Garbage Collector

5. Call Stack

- Keeps track of function calls
- Executes code line by line

Execution Flow (Easy to remember):

```
JavaScript Code
   ↓
Parser → AST
   ↓
Ignition → Bytecode
   ↓
TurboFan → Optimized Machine Code
   ↓
Execution (Call Stack + Heap)
```