React is library build on top of javascript with javascript code

React works only in place where the tag is marked as root so react is library which can load any where or any peace of code standalone

Framework … it works with all the structure with it already

**React can be used in existing applications where it is need by adding the CDN but framework already have all set of features like routing etc with it.**

**All framework need to build entire application with that framework but react can work anywhere in any code**

**Incuding react through CDN**

What is CDN?

CND(content delivery network) is system of distributed servers that are rendered to user based on geographical location of user.The main purpose of CDN is fast delivery, content availability and performance

 <!-- core of react -->

<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>

<!-- react dom which we need to modifiy dom -->

<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

<!-- why 2 fiels .. there are differrent places where react is used like mobile(react native) so first file have core react functionaltiies and second file is used to manuplate DOM in website or browser -->

Advantages:

* Fast access to resources
* Performance speed as files are not included in application
* Bandwidth saving 🡪 once the script is loaded to browser it will be stored in cache so each time when user try to access it … the cache gets loaded instead of seprate call to cdn
* **Even if main server is down CdN can still deliver the content**
* Offloading static resources to a CDN reduces the burden on your server, allowing it to focus on dynamic content.
* Easy integration just one line of code

Disadvantage

* No ofline access
* When u need to bundle the application for performance optimization using vite or any other packages it wont work for cdn

How cdn works even if main server is down

* Cdn cache static content like

 JavaScript files (e.g., jQuery, React)

 CSS files (e.g., Bootstrap)

 Fonts

 Images

 Videos

These are stored across **multiple edge servers** around the world.

**What happens if your main web server goes down?**

* Your dynamic content (e.g., database-driven pages, user-specific data) won’t work.
* But any assets served through a **CDN link** (like a JavaScript library or stylesheet) will still **load just fine** — because they're **not hosted on your server**.

✅ So your **site might partially work**, or **error pages will look better/styled** because CDN resources like CSS or fonts still load.

**Bonus: You can even host your own assets on a CDN (like Cloudflare, AWS CloudFront), so they stay online even if your server goes down. –how?**

## **How a CDN Works Behind the Scenes**

### 1. ****You upload or link your files****

* JS, CSS, images, fonts, videos—anything static.
* These files are stored in a **central origin server** (can be your server or a cloud bucket like AWS S3).

### 2. ****CDN distributes copies to edge servers****

* The CDN **caches** your assets across **global edge locations** (also called PoPs – Points of Presence).
* Example CDNs: Cloudflare, AWS CloudFront, Akamai, Fastly, jsDelivr.

### 3. ****User requests your site****

* Let’s say your site uses a CDN URL like:

html

CopyEdit

<img src="https://cdn.yoursite.com/assets/logo.png" />

### 4. ****The CDN finds the closest edge server****

* Based on the user’s **geographic location**.
* The nearest CDN edge server responds (much faster than a central server).-->graph algorithm 🡪faster near location

### 5. ****Cache hit or miss****

* ✅ **Cache Hit**: File is already in that edge server → served instantly.
* ❌ **Cache Miss**: File not there yet → pulled from origin → cached for future users.

### 6. ****If your origin server goes down****

* The **cached files** in the edge servers **still work** for users for a limited time (as long as TTL allows).

## 📦 Hosting Your Own Assets on a CDN (Cloudflare / AWS CloudFront)

### 🚀 ****Option 1: Cloudflare (Free & Easy)****

**Steps:**

1. Point your domain to Cloudflare (change your DNS nameservers to Cloudflare’s).
2. Cloudflare acts as a reverse proxy (sits between your users and your server).
3. Enable **"Cache Everything"** or just static asset caching.
4. Now, when users visit:

https://yourdomain.com/images/banner.jpg

Cloudflare caches and serves it from their global edge network.

✅ **Bonus**: Cloudflare offers **free CDN**, DDoS protection, SSL, and performance optimization.

### ☁️ ****Option 2: AWS CloudFront (More control & scale)****

**Steps:**

1. Upload your static assets to **Amazon S3** bucket.
2. Create a **CloudFront distribution** and point it to your S3 bucket or origin server.
3. CloudFront assigns you a CDN URL like:

bash

CopyEdit

https://d1234abcd.cloudfront.net/images/logo.png

1. You can also configure a custom domain:

arduino

CopyEdit

https://cdn.yoursite.com/logo.png

1. Set caching policies (how long assets stay cached, invalidation rules, etc.)

✅ Use CloudFront for high-scale apps, video delivery, or fine-grained control.

## 🔒 What happens if your origin server crashes?

* Assets already cached in the CDN **still serve users**.
* Users won't even notice, as long as they're not requesting new files or expired cache.
* This improves **uptime and performance** massively.

### 🧠 TL;DR

| **Topic** | **Summary** |
| --- | --- |
| **CDN** | Globally distributes your static files for speed and reliability |
| **How it works** | Requests are routed to closest server; cache hit/miss decides source |
| **Cloudflare** | Simple, free, domain-based CDN with cache rules |
| **CloudFront** | Enterprise-level, customizable, S3-compatible CDN |
| **Why it matters** | Makes your site fast, scalable, and resilient—even if your main server fails |

Let me know if you want a walkthrough of setting up your own asset CDN with Cloudflare or CloudFront (step-by-step with example files).

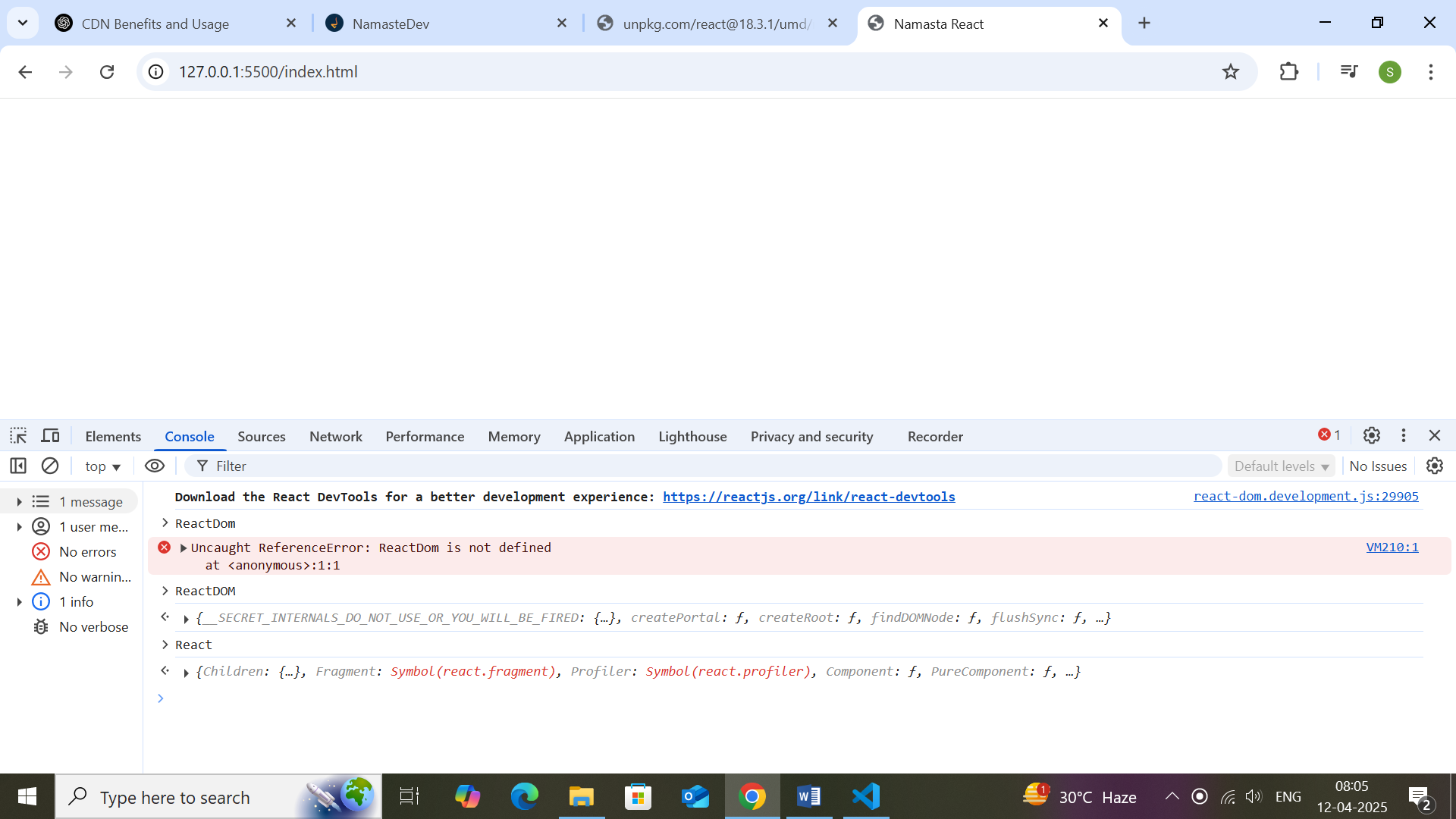
**Why do we use cross origin?**

To control how browsers handle cross-origin requests (requests to a different domain), and to comply with CORS (Cross-Origin Resource Sharing) rules for security and error handling.

* **When there is error with CDN link loaded the error message wont be shown properly in error when user use CDN without CORS**
* When we need to access other CDN using credentials without CORS we wont be able to access it as it throws error
* **There are 3 values for CORS based on which it works**
  + Corsorigin 🡪 anonymous or cors
  + **Use-credentails**
  + **None ->disable cors**

| **Use Case** | **Why Use crossorigin** |
| --- | --- |
| Loading scripts from CDN | For full error stack traces |
| Using fonts/images from other domains | To avoid CORS errors |
| Drawing images on canvas from another domain | To access or export canvas data |
| Securely loading cross-origin resources | To obey browser’s CORS policy |

**After adding cdn to html page react is into html page**



**Above shows all the react and react related DOM manuplation script are loaded and user can use it**

**First react code**

 <script>

    // it is core thing in react and no need react dom

    const h1 = React.createElement('h1',{ className: 'heading' },'Hello world');

    //create need root to do all manuplation..when we ned to h1 into root ..we need to create root

    // everything will render in the root

    const root = ReactDOM.createRoot(document.getElementById('root'));

    root.render(h1);

 </script>

**Note about order of code execution**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Namasta React</title>

    <style>

        .heading{

            color: red;

        }

    </style>

    <!-- added react in project in header throws error as when the html is parsed and reaches script tag it tried to run before dom is loaded -->

 <!-- core of react -->

  <!-- we can use defer which tells user to download script now and use later -->

<script crossorigin  defer src="https://unpkg.com/react@18/umd/react.development.js"></script>

<!-- react dom which we need to modifiy dom -->

<script crossorigin defer src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

<!-- why 2 fiels .. there are differrent places where react is used like mobile(react native) so first file have core react functionaltiies and second file is used to manuplate DOM in website or browser -->

 <script defer src="./app.js"></script>

</head>

<body>

    <div id="root">

    </div>

</body>

<!-- added react in project -->

 <!-- core of react -->

</html>

When we include script tag in header .. before body is loaded it throws error we can either use defer to tell browser to run src later or add event listener it custom js scripts like app.js with addEventListerner(‘DomLoaded’)

React works only inside root element

<body>

    <!-- react is working only inside root and other part of html remains same -->

    <div>

        <h1>test</h1>

    </div>

    <div id="root">

        <!-- the content in the root tag get replaced by render method -->

    </div>

</body>