Know about
Render, Reactivity

Rendering

What is Render in react?

- Render technically means to provide service.
- In react, render is a method that tell react what to display.
- React schedules a render every time the state of a component changes.

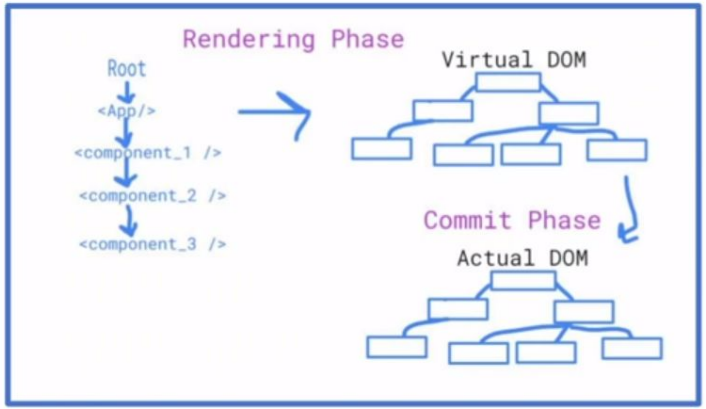
How does rendering happen?

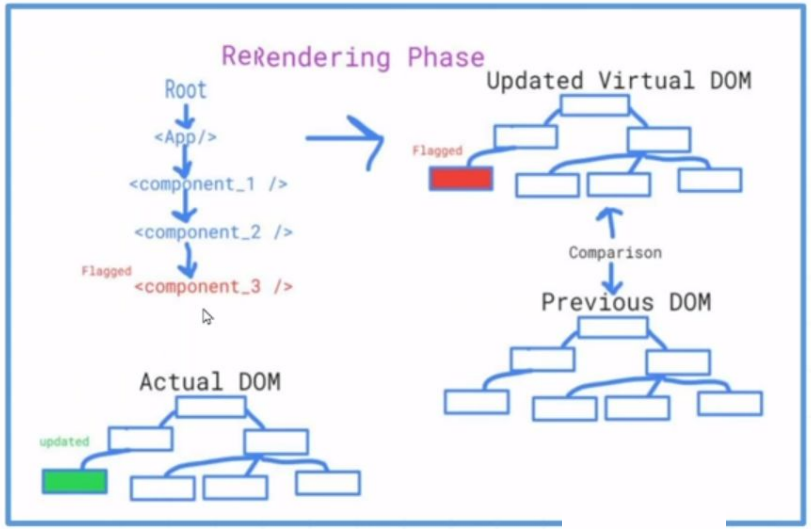
```
import React from "react";
import ReactDOM from "react-dom";

const App =( )=>{
   return <div>
        Hello World!
       </div>
}

ReactDOM.render(<App/>,document.querySelector("#root");
```

The render function requires two parameters -

- What is to be displayed
- Where is to be displayed

Rendering vs Re-rendering

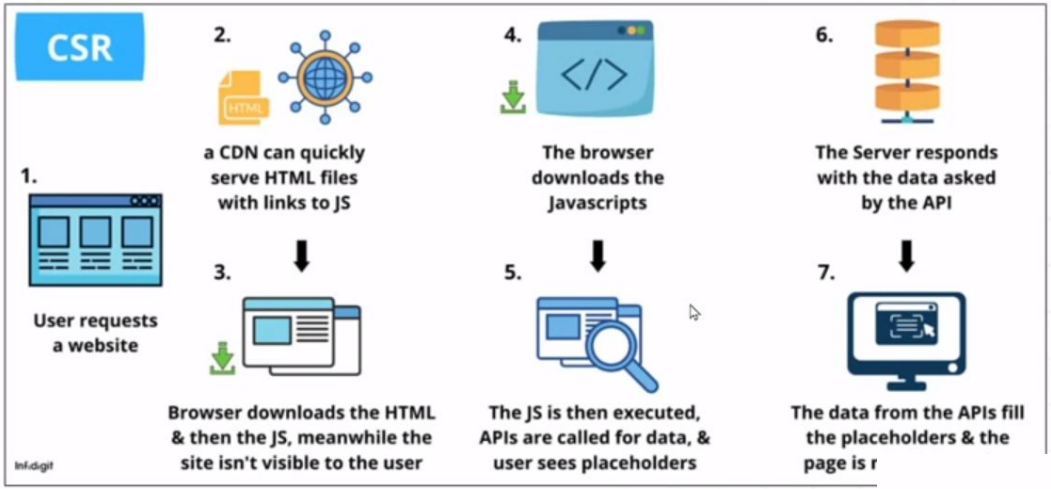




SSR vs CSR

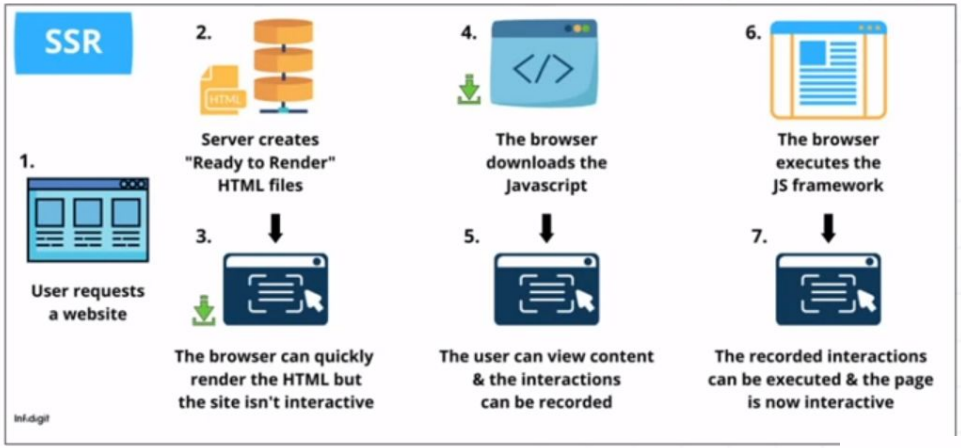
Client Side Rendering

CSR is a technique where all the page resources are rendered on the client's browser rather than the server.



Server Side Rendering

In SSR, all page resources are rendered on the server rather than the browser. The server sends fully rendered HTML, CSS and JS to the client when a request is made.



SSR? CSR? When to use what?

use SSR

- if SEO is your priority, typically when you are building a blog site and you want everyone who searching on google go to your website, then SSR is your choice.
- if your website needs a faster initial loading.
- if the content of your website doesn't need much user interaction.

use CSR

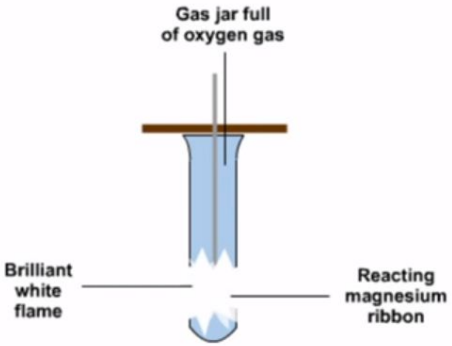
- when SEO is not your priority
- if your site has rich interactions
- if you are building a web application

Reactivity

Meaning of Reactivity

The meaning of **REACTIVITY** is the quality or state of being reactive.

In **chemistry**, Reactivity is a measure of how easily an element will combine with other elements to form compounds (a new element). => Reactivity হল একটি পরিমাপ যে একটি উপাদান কত সহজে অন্যান্য উপাদানের সাথে মিলিত হয়ে যৌগ গঠন করবে (একটি নতুন উপাদান)



What is Reactivity/ Reactive programming?

Reactivity is an **advanced overwhelming paradigm/ model**.

Reactive Programming allows us to set **Dynamic Behavior only at Declaration Time**.

Reactive Programming is different from Imperative programming because as the name suggests, it reacts when something in our application is occurring or done.

It relies on **Asynchronous Nature** that happens in time.

It **stays responsive** in the both case of **Failure and Success.**

In the Reactive world, we can not wait on something that will happen to th . We have to continue the execution, and if something happens, we need to R That is what the Async Pattern is.

Is React Reactive?

React is **not react**. With a combination of RxJx, React becomes reactive.

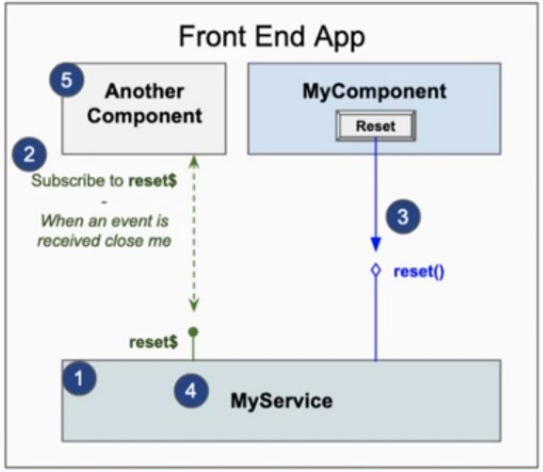
RxJS (**Reactive Extensions for JavaScript**) is a library for reactive programming using observables that makes it easier to compose asynchronous or callback-based code.

Is RxJS the same as react?

- React is a javascript library for building user interfaces, RxJS is a javascript library for reactive programming using Observables.
- Both these javascript libraries are not comparable to each other since it serve different purposes.
- Both these can be used together to create better single page applications.

Let’s understand reactivity in front end

Example - 1



1. **MyService** is instantiated and **reset$** stream is created
2. **AnotherComponent** subscribes to **reset$** stream and specifies that, when an event is received on this stream, it will clear any data it may have rendered
3. **Reset** Is clicked and **MyComponent** invokes the method **reset** of **MyService**
4. The method **reset** generates a new event on the **reset$** stream
5. **AnotherComponent** is notified of the new event on the **reset$** stream and reacts closing itself clearing any data it mah rendered

Let’s understand reactivity in programming language

Example - 2

In general programming paradigm, if **sum = number1 + number2** at the time of declaration then **sum** will contain the result of addition of **number1** and **number2** at that moment of time.

However if after that statement, we **change the value of number1 and number2** then value of **sum will not get changed**.

Let’s understand reactivity in programming language

Example - 2 (imperative programming behaviour)

```
var number1 = 1;
var number2 = 2;
let sum = number1 + number2;
console.log(sum);
number1 = 3;
number2 = 4;
console.log(sum);

// Ouput -->
// 3
// 3
```

value of sum is same every time the values of a & b are changing

In this example, the value of **number1** and **number2** does not affect variable sum, so in **Imperative Programming,** The concept of **Dynamic Change** is missing.

Let’s understand reactivity in programming language

Example - 2 (dynamic programming behaviour)

```
import { Observable } from 'rxjs';

let a=1;
let b=2;
let sum=0;

let observable = new Observable((subscriber)=>{
  subscriber.next(a);
  subscriber.next(b);
  console.log("Sum = ",a+b);
})

observable.subscribe((x) => {
  console.log(x);
});

observable.subscribe(()=>{
  a=3;
  b=4;
})

observable.subscribe((x) => {
  console.log(x);
});
```

Ignore the code syntax,
focus on output
& clear concept.

```
Preview (local)
1
2
Sum = 3
3
4
Sum = 7
```

value of sum changes automatically every time the values of a & b are chan

Thank you!