**Next.js Data Fetching (SSR,SSG,ISR,CSR)**

**Client Side Rendering**

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with low confidenceFirst Server send the markup to client then fetch the data from database after that. Mostly something with useEffect.

**🡺**

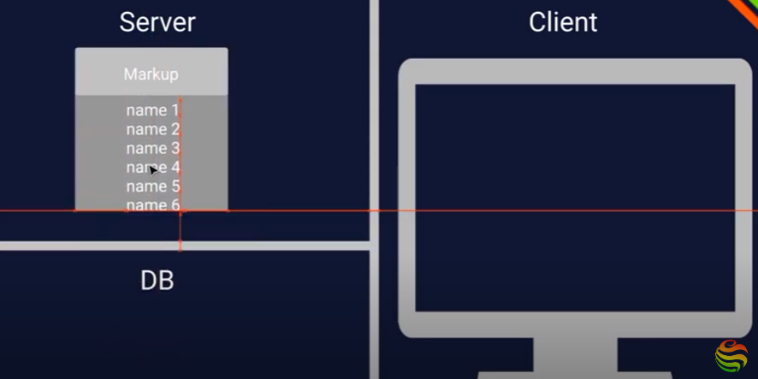
Graphical user interface

Description automatically generated

**🡺**

**Server-Side Rendering (SSR)**

Before we actually send the markup to client, we fetch the data at server-side bundle into markup then send to client. In the end first thing client will see is markup with fetched data. İf the client refreshes the page server will do that again. Fetch the data into markup and send the data fetched markup to client.

Graphical user interface

Description automatically generated

**🡺**

**Static Site Generation**

İt creates data fetched markups (it creates data json files and markups seperately) at the build time.

In every rebuild next.js will fetch the data from database and creates static markups with static data.

İf client refresh the page server will not fetch data again, it will return the static page that been already created by server. This is probably the fastest way the serve page because file stays in the server even before the client asks for page. Disadvantage of it that when data in the db changes, pages will not refresh it. It need rebuild.

A picture containing graphical user interface

Description automatically generatedA screenshot of a phone

Description automatically generated with low confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceShape

Description automatically generated with low confidenceShape

Description automatically generated with low confidenceA screenshot of a computer

Description automatically generated with medium confidence **=>**

**İncremental Static Generation**

It has common life cycle with SSG. Server still fetches statically data to statically generated markup but additionally we can set interval time for refetch the data and generate the markup statically.

**Notes:**

* In client site rendering we don’t have our data on html so its not SEO friendly

Graphical user interface

Description automatically generated

* In the other hand **server-side rendering** data fetching will look like this and data will be in the markup

|  |  |
| --- | --- |
|  | export default function **ServerSideRendered**({ data }) { |
|  | return ( |
|  | <> |
|  | {data.map((e) => ( |
|  | <h2 key={e.id}>{e.name}</h2> |
|  | ))} |
|  | </> |
|  | ); |
|  | } |
|  |  |
|  | export async function **getServerSideProps**() { |
|  | const res = await fetch(YOUR\_API\_URL); |
|  | const data = await res.json(); |
|  |  |
|  | return { |
|  | props: { |
|  | data, // will be passed to the page component as props |
|  | }, |
|  | }; |
|  | } |

Graphical user interface

Description automatically generated

* In the Sitatic Site Generation we almost same code with SSR. The only difference is we use getStaticrops over GetServerSideProps.

|  |  |
| --- | --- |
|  | export default function **ServerSideRendered**({ data }) { |
|  | return ( |
|  | <> |
|  | {data.map((e) => ( |
|  | <h2 key={e.id}>{e.name}</h2> |
|  | ))} |
|  | </> |
|  | ); |
|  | } |
|  |  |
|  | export async function getStaticProps () { |
|  | const res = await fetch(YOUR\_API\_URL); |
|  | const data = await res.json(); |
|  |  |
|  | return { |
|  | props: { |
|  | data, // will be passed to the page component as props |
|  | }, |
|  | }; |
|  | } |

Graphical user interface

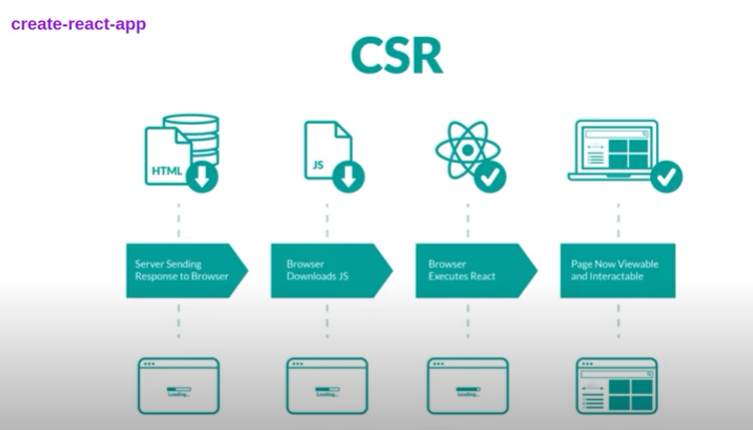
Description automatically generated

* The only differtence between incremental site generation ssg is revalidate interval.

|  |  |
| --- | --- |
| export async function getStaticProps () { |  |
| const res = await fetch(YOUR\_API\_URL); |  |
| const data = await res.json(); |  |
|  |  |
| return { |  |
| props: { |  |
| data, // will be passed to the page component as props |  |
| },  Revalidate:10 // in seconds |  |
| }; |  |
| } |  |
| When the use which.  CSR  Very very dynamic pages like dashboard, crm, accounting apps..  SSG  Cahanges not often. Blog pages, documentation apps, servers pages(quickiest load time as possible )  ISG  If pages are dynamic but changes are not often. Like product pages but products changes once a day once a week.  SSR  Quite dynamic data, data changes very very often and need fast upload like reddit, ebay.  Life Cycles: |  |

Diagram

Description automatically generated



[When should I use getServerSideProps](https://nextjs.org/docs/basic-features/data-fetching/get-server-side-props#when-should-i-use-getserversideprops)

You should use getServerSideProps only if you need to render a page whose data must be fetched at request time. This could be due to the nature of the data or properties of the request (such as authorization headers or geo location). Pages using getServerSideProps will be server side rendered at request time and only be cached if [cache-control headers are configured](https://nextjs.org/docs/going-to-production#caching).

If you do not need to render the data during the request, then you should consider fetching data on the [client side](https://nextjs.org/docs/basic-features/data-fetching/get-server-side-props#fetching-data-on-the-client-side) or [getStaticProps](https://nextjs.org/docs/basic-features/data-fetching/get-static-props).

## [When should I use getStaticPaths?](https://nextjs.org/docs/basic-features/data-fetching/get-static-paths#when-should-i-use-getstaticpaths)

You should use getStaticPaths if you’re statically pre-rendering pages that use dynamic routes and:

* The data comes from a headless CMS
* The data comes from a database
* The data comes from the filesystem
* The data can be publicly cached (not user-specific)
* The page must be pre-rendered (for SEO) and be very fast — getStaticProps generates HTML and JSON files, both of which can be cached by a CDN for performance

[When should I use getStaticProps?](https://nextjs.org/docs/basic-features/data-fetching/get-static-props#when-should-i-use-getstaticprops)

You should use getStaticProps if:

* The data required to render the page is available at build time ahead of a user’s request
* The data comes from a headless CMS
* The page must be pre-rendered (for SEO) and be very fast — getStaticProps generates HTML and JSON files, both of which can be cached by a CDN for performance
* The data can be publicly cached (not user-specific). This condition can be bypassed in certain specific situation by using a Middleware to rewrite the path.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text, letter

Description automatically generated

Graphical user interface, text, application

Description automatically generated

If user scroll down server will keep pre render more posts.

Graphical user interface, text, application, email

Description automatically generated

Text, letter

Description automatically generated

Graphical user interface, text, application

Description automatically generated

A picture containing timeline

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated