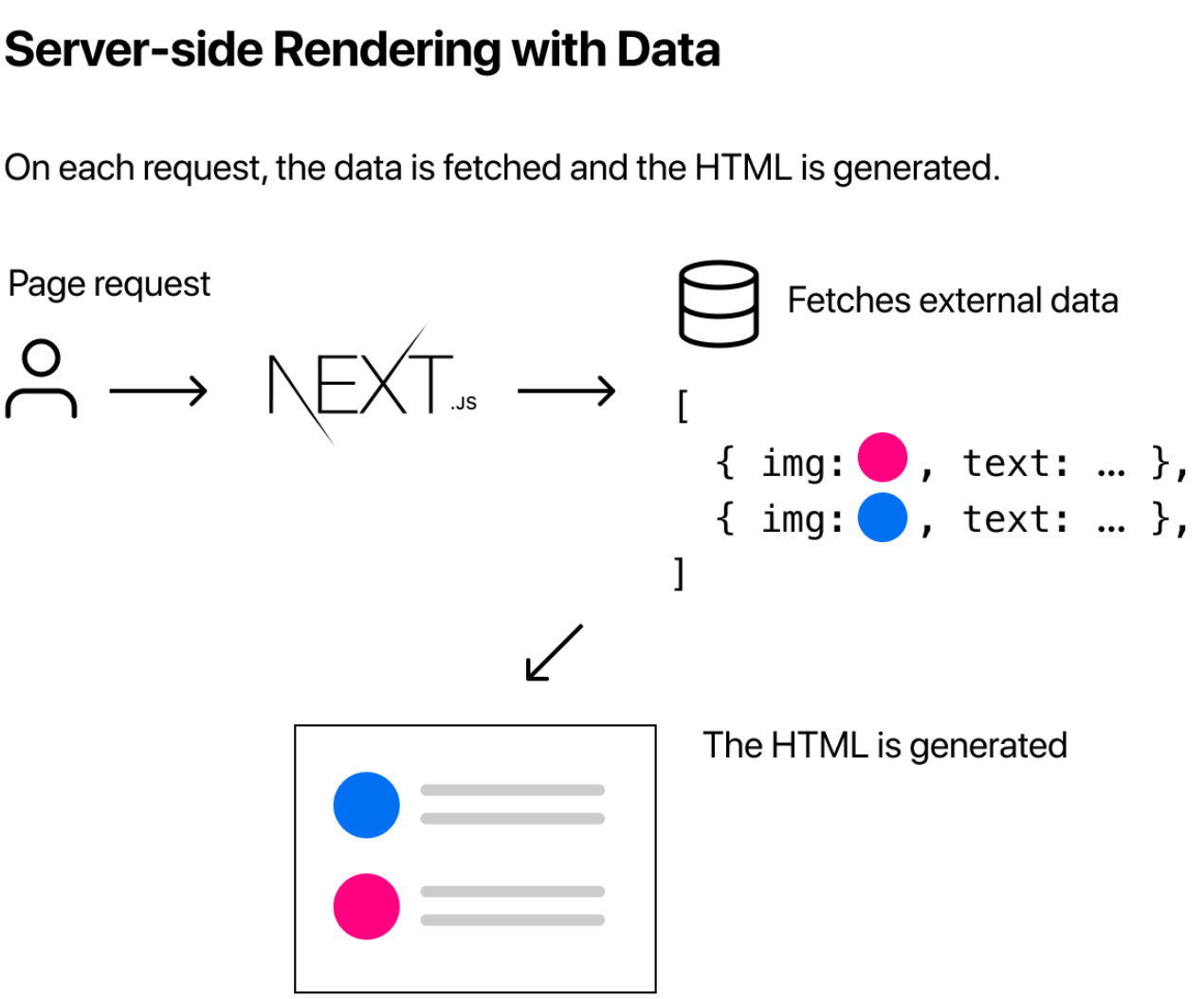
SSR - Server Side Rendering



Source: https://nextjs.org/learn/basics/data-fetching/request-time

AGENDA

1. getServerSideProps
2. Using cache
3. Potential issues

1. getServerSideProps

- Run on server-side only
- Run per page request
- **TTFB** (Time To First Byte) will be slower than `getStaticProps`
- Export `getServerSideProps` from your Page to enable SSR

```
export async function getServerSideProps(context) {
  return {
    props: {}, // will be passed to the page component as props
  }
}
```

context

- params: path/route params
- req: HTTP IncomingMessage object
- res: HTTP response object
- query: an object representing the query string
- ...

> More details: https://nextjs.org/docs/basic-features/data-fetching#getserversideprops-server-side-rendering

In this example, each page request will always take 3 seconds to query data before return it to client.

```
export async function getServerSideProps(context) {
  // fake slow query
    await new Promise((resolve) => setTimeout(resolve, 3000))

    return {
        props: {},
    }
}
```

2. Using cache

Using s-maxage=5

- Keep the page **FRESH** in 5 seconds, after that call getServerSideProps again on page request.

```
export async function getServerSideProps(context) {
    context.res.setHeader('Cache-Control', 's-maxage=5')

  // ...
}
```

timeline	s-maxage=5
first request	call `getServerSideProps`() and cache in CDN
next 0-5s	return from cache immediately
after that	call `getServerSideProps`() and cache in CDN
next 0-5s	return from cache immediately
after that	...

Using s-maxage=5 and stale-while-revalidate

- Keep the page FRESH for 5 seconds.
- Then on page request, return the stale data immediately and call getServerSideProps() to have new cache.

```
export async function getServerSideProps(context) {
    context.res.setHeader(
    'Cache-Control',
    's-maxage=5, stale-while-revalidate'
  )

  // ...
}
```

timeline	s-maxage=5, stale-while-revalidate
first request	call `getServerSideProps`() and cache in CDN
next 0-5s (s-maxage)	return from cache immediately
after that (swr)	return from cache immediately (stale), then call `getServerSideProps`()
once new cache is set	start a new cycle (cache 5s and start the swr on the next request)

Using s-maxage=5 and stale-while-revalidate=5

```
export async function getServerSideProps(context) {
    context.res.setHeader(
    'Cache-Control',
    's-maxage=5, stale-while-revalidate=5'
  )

  // ...
}
```

timeline	s-maxage=5, stale-while-revalidate
first request	call `getServerSideProps`() and cache in CDN
next 0-5s (s-maxage)	return from cache immediately
next 0-5s (swr)	return from cache immediately (stale), then call `getServerSideProps`()
once new cache is set	start a new life cycle (0 -> 10s)

Demo: https://learn-nextjs-g3cq5v13a-paulnguyen-mn.vercel.app/

> Read more about https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control

3. Potential issues

- Clear CDN cache for new deployment --> Vercel do it for us
- Be careful with page that render data for specific users.
    - Different promotions
    - Different results based on user's age
    - ...

Series - NextJS + Typescript 🎉

- Tác giả: **Hậu Nguyễn**
- Được phát hành trên kênh youtube **Easy Frontend**.
- Tài liệu pdf và videos đều có bản quyền thuộc về Easy Frontend.
- Videos được phát hành cho fan cứng trước, public sau.
- Đăng ký fan cứng để xem series này đầy đủ và sớm nhất nhé.

☎️ Kết nối với mình

- ✅ Follow Facebook: https://www.facebook.com/nvhauesmn/
- ✅ Like Fanpage: https://www.facebook.com/learn.easyfrontend
- ✅ Youtube Channel: https://www.youtube.com/easyfrontend