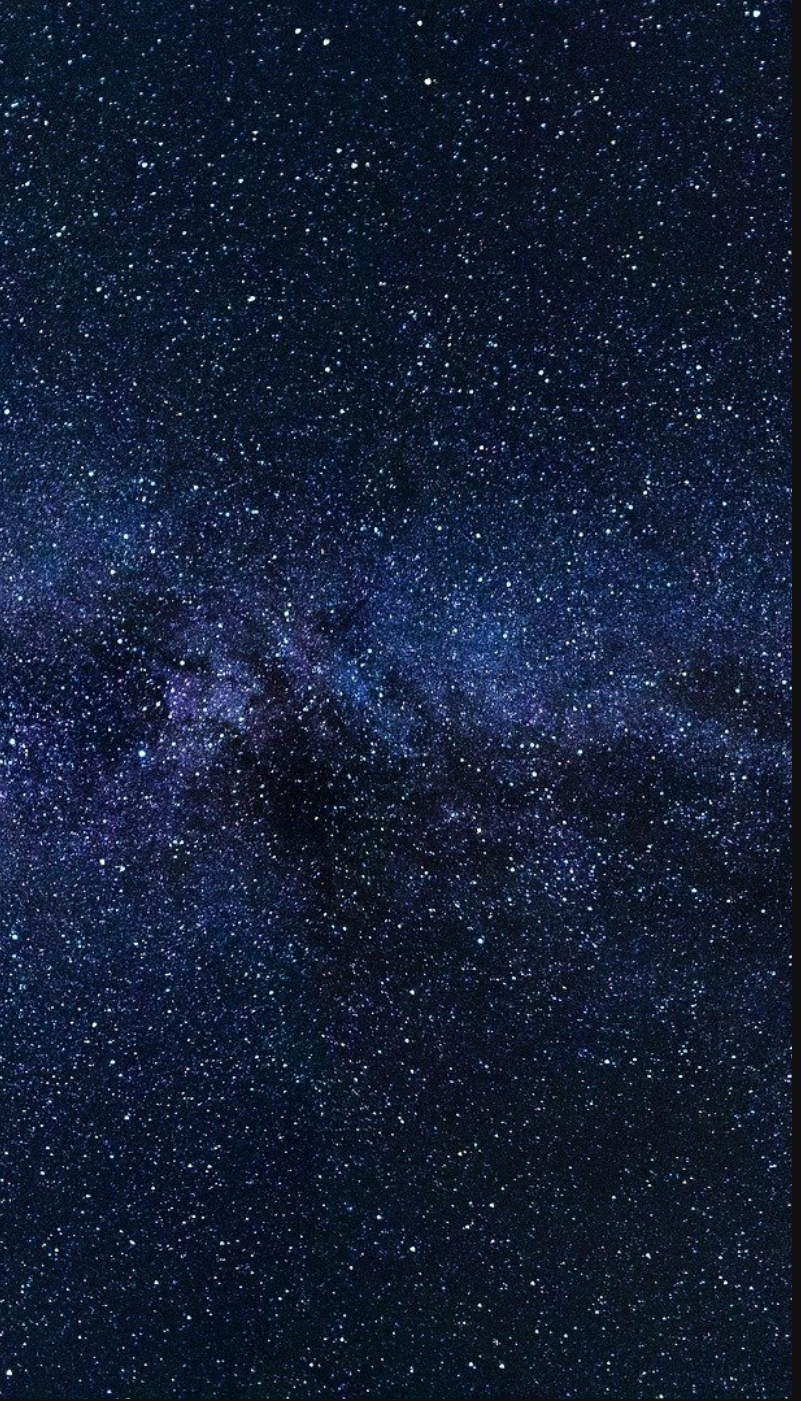
Astro and the Island Architecture



Agenda

- TLDR
  - What is Astro
  - What is the Island Architecture
- The confusing state of the modern web
- Where the Islands Architecture fits in
- How Astro Helps

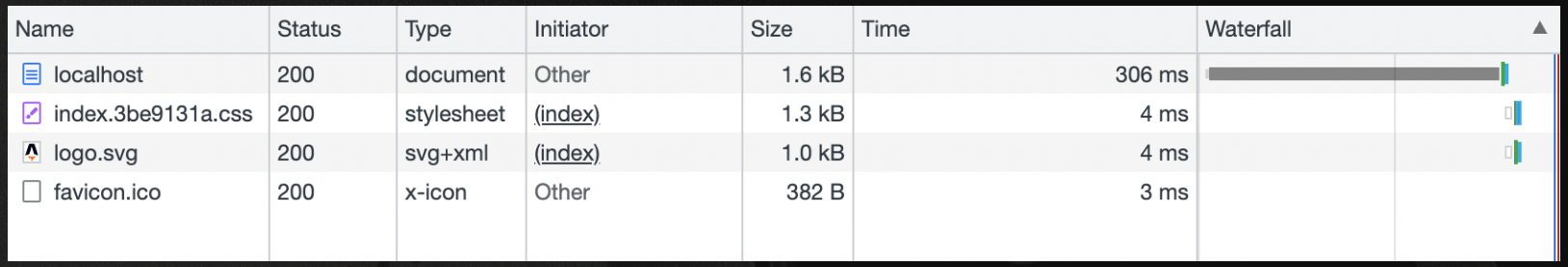
What is Astro?

Astro

Astro is a new kind of static site builder for the modern web

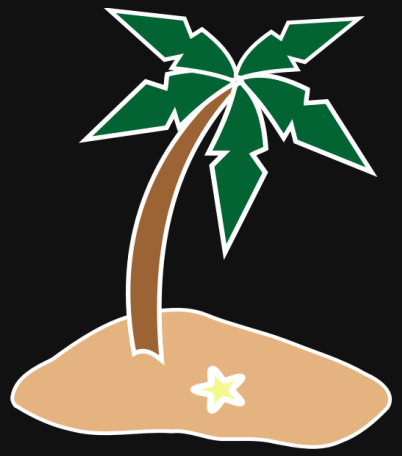
Killer Feature

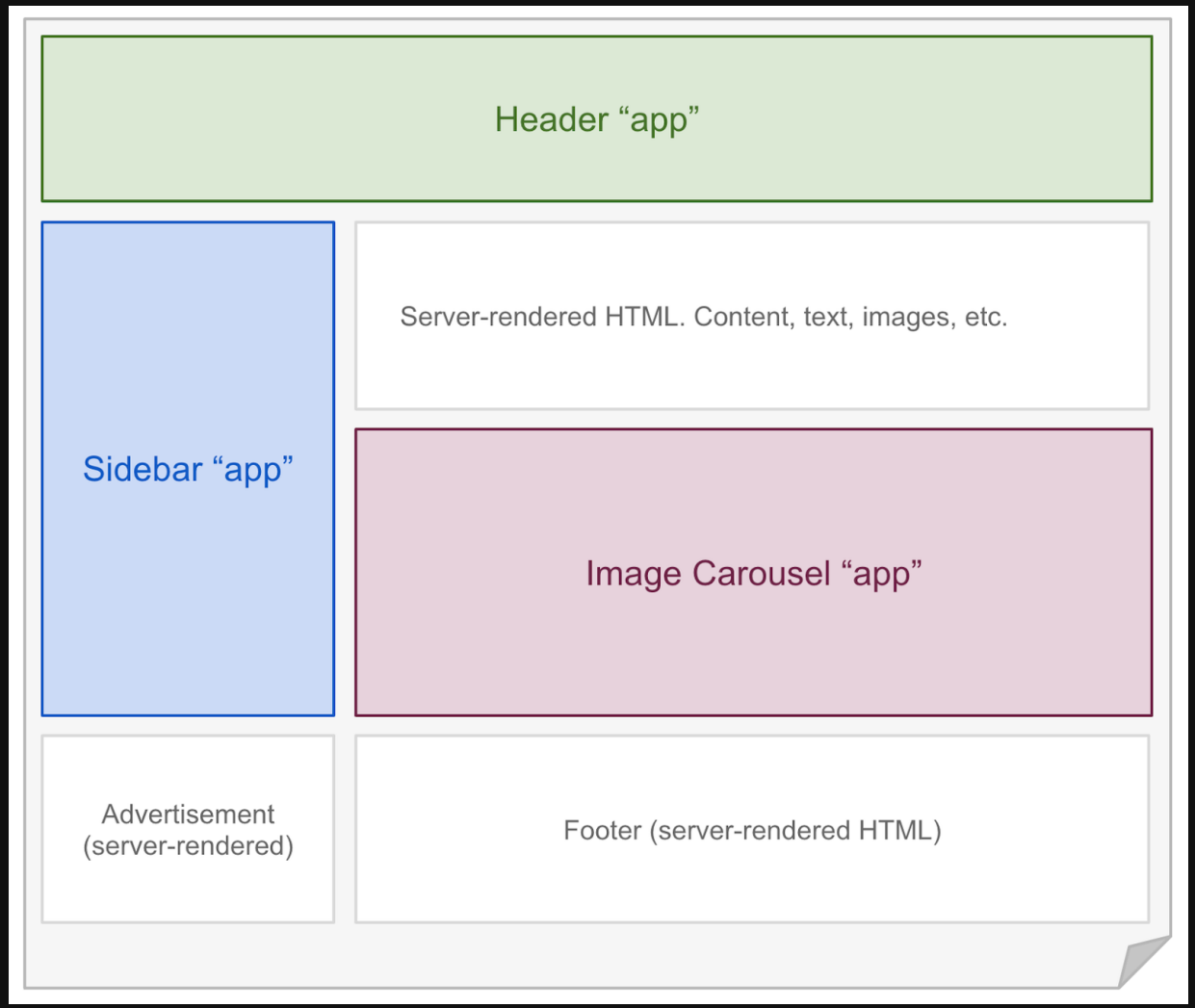
Best Feature



Name	Status	Type	Initiator	Size	Time	Waterfall
localhost	200	document	Other	1.6 kB	306 ms	
index.3be9131a.css	200	stylesheet	(index)	1.3 kB	4 ms	
logo.svg	200	svg+xml	(index)	1.0 kB	4 ms	
favicon.ico	200	x-icon	Other	382 B	3 ms	

What is the Islands Architecture?




Header “app”
Sidebar “app”
Server-rendered HTML. Content, text, images, etc.
Image Carousel “app”
Advertisement (server-rendered)
Footer (server-rendered HTML)

“...All of the other content is secondary to this information, and its inclusion in the HTML becomes a product decision. How vital is this bit of information to a user visiting the page? How important is that widget to the business model? A "buy now" button that directly relates to revenue should be easily prioritized over a site feedback survey button that relates to information gathering.”

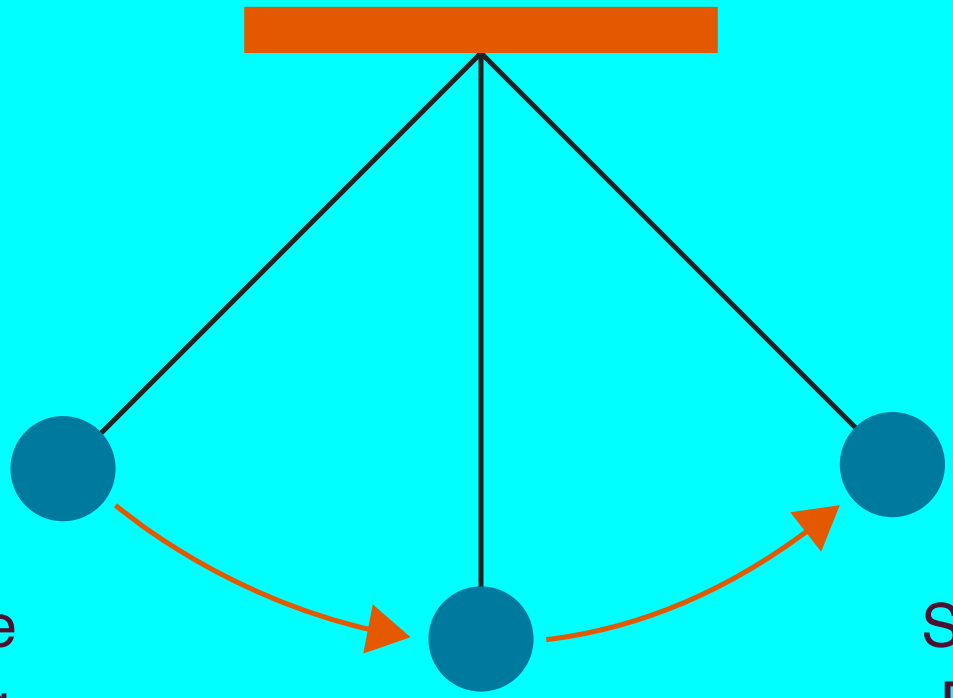
- Jason Miller

https://jasonformat.com/islands-architecture/

The current messy state of rendering **HTML** on the web

Unpopular Opinion

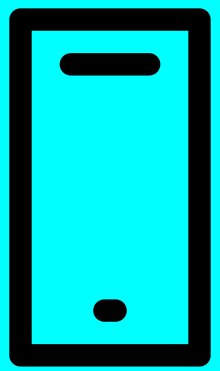
SSR is overused

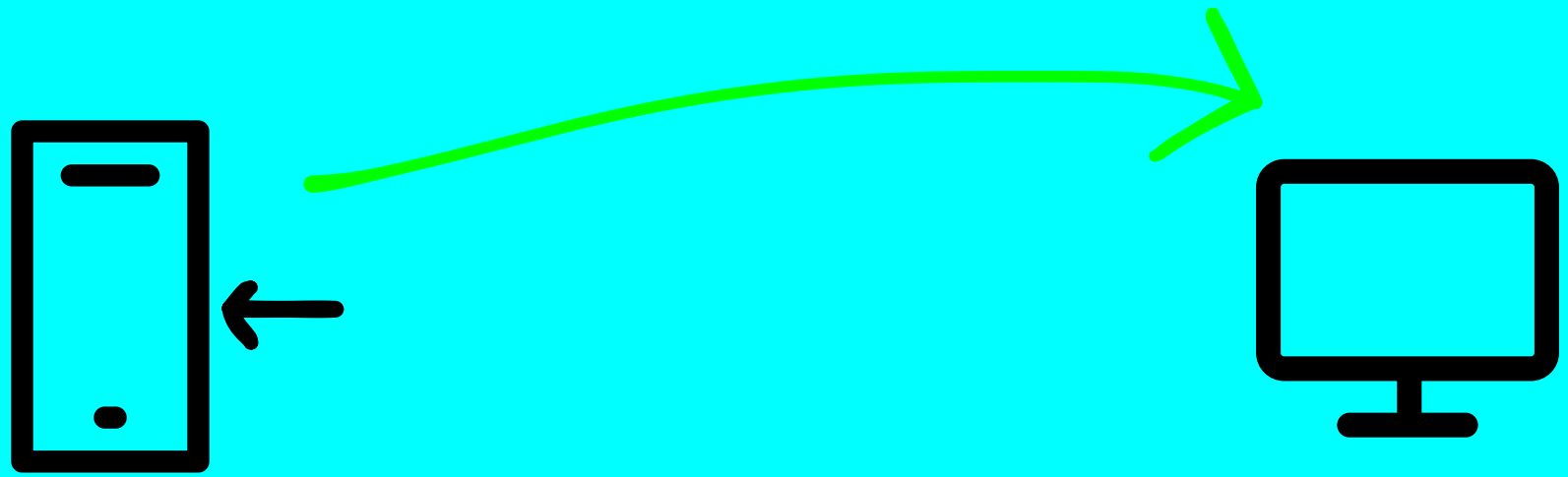


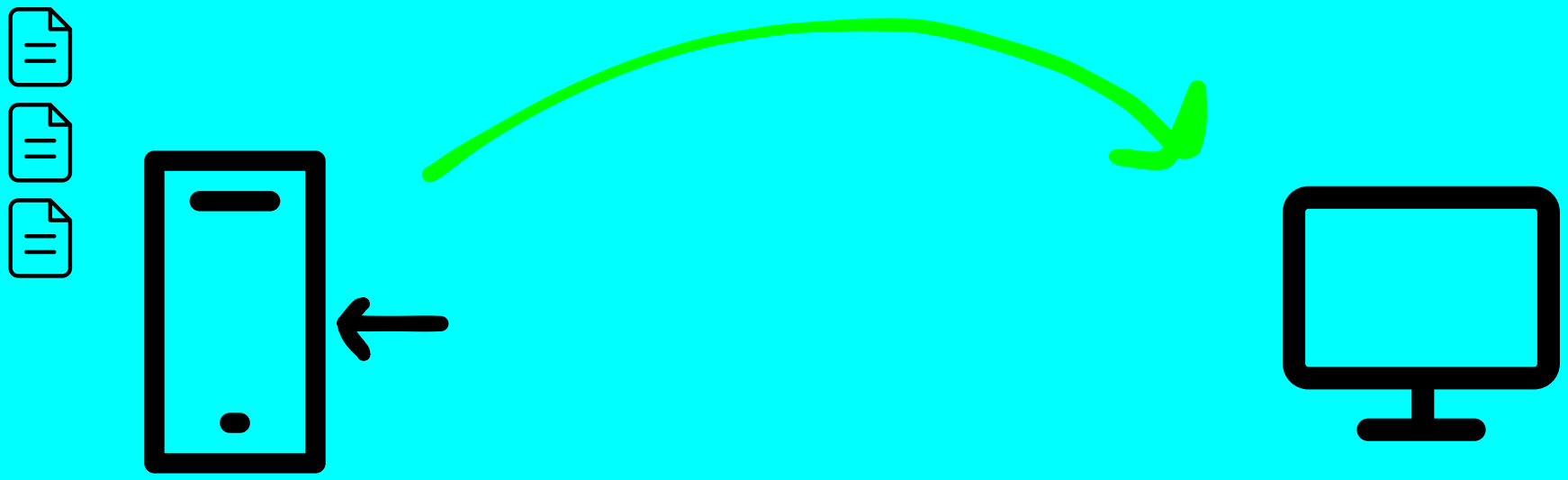
Client Side
Rendering

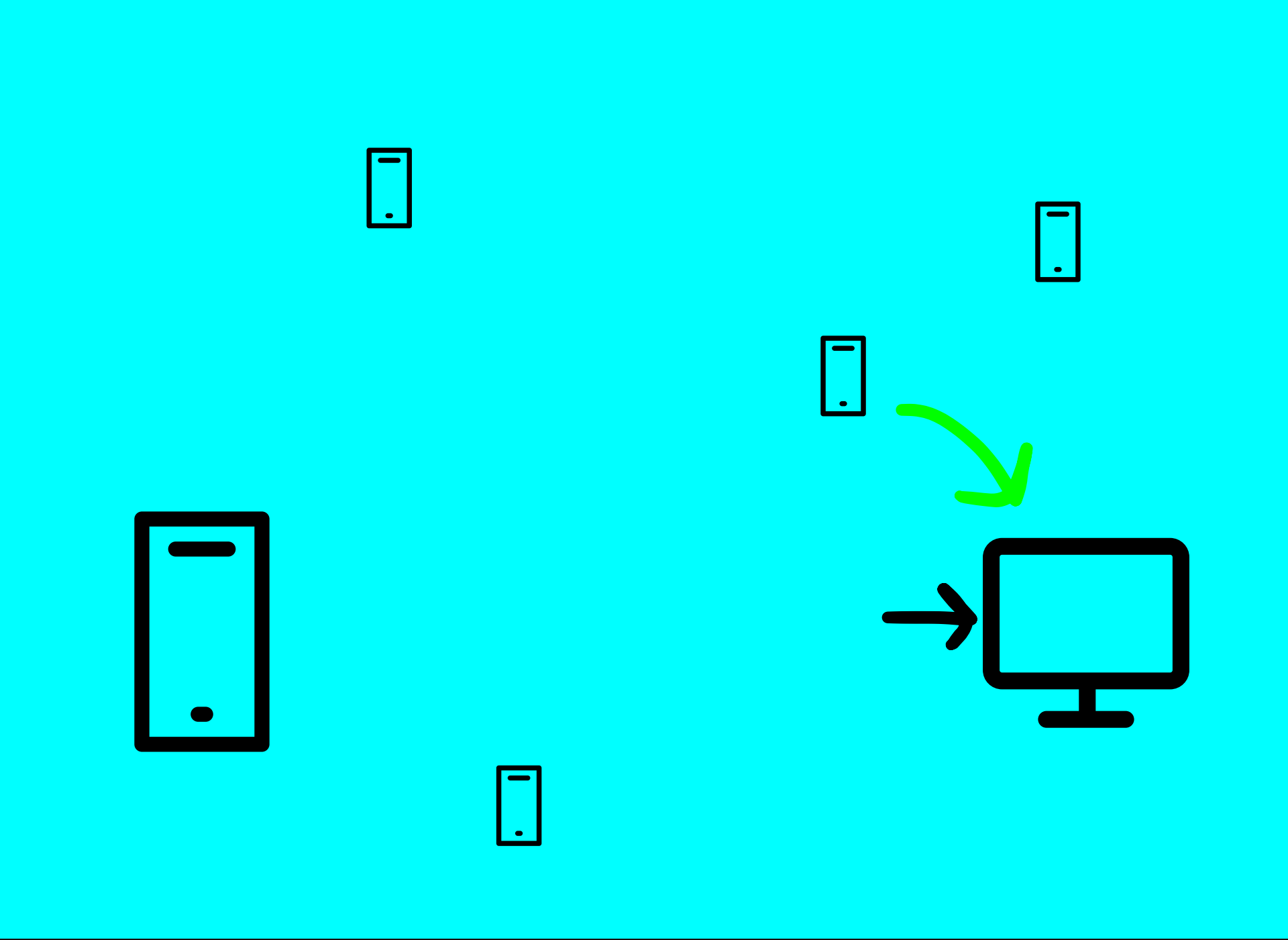
Server Side
Rendering

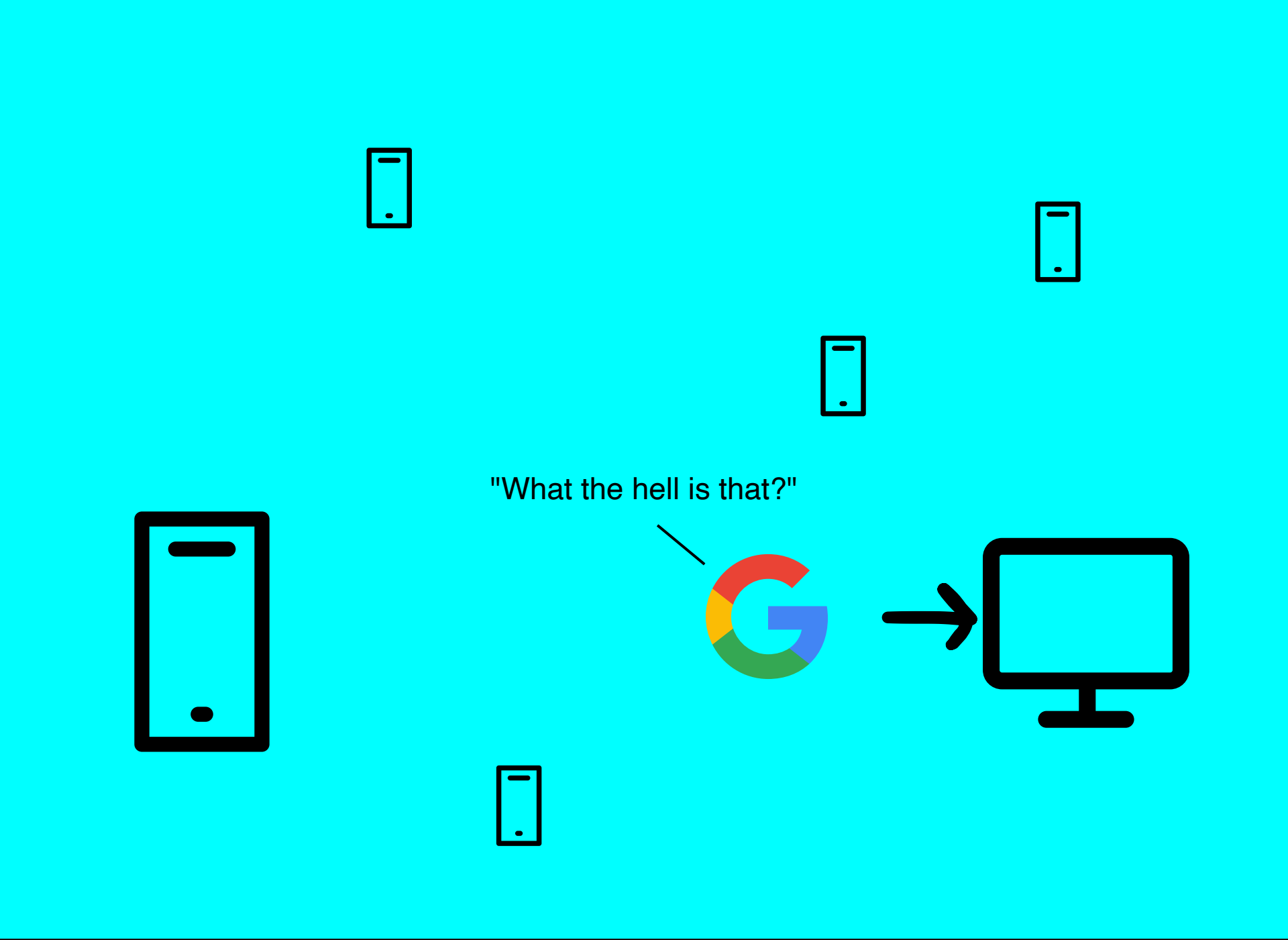
SSR vs SPA?

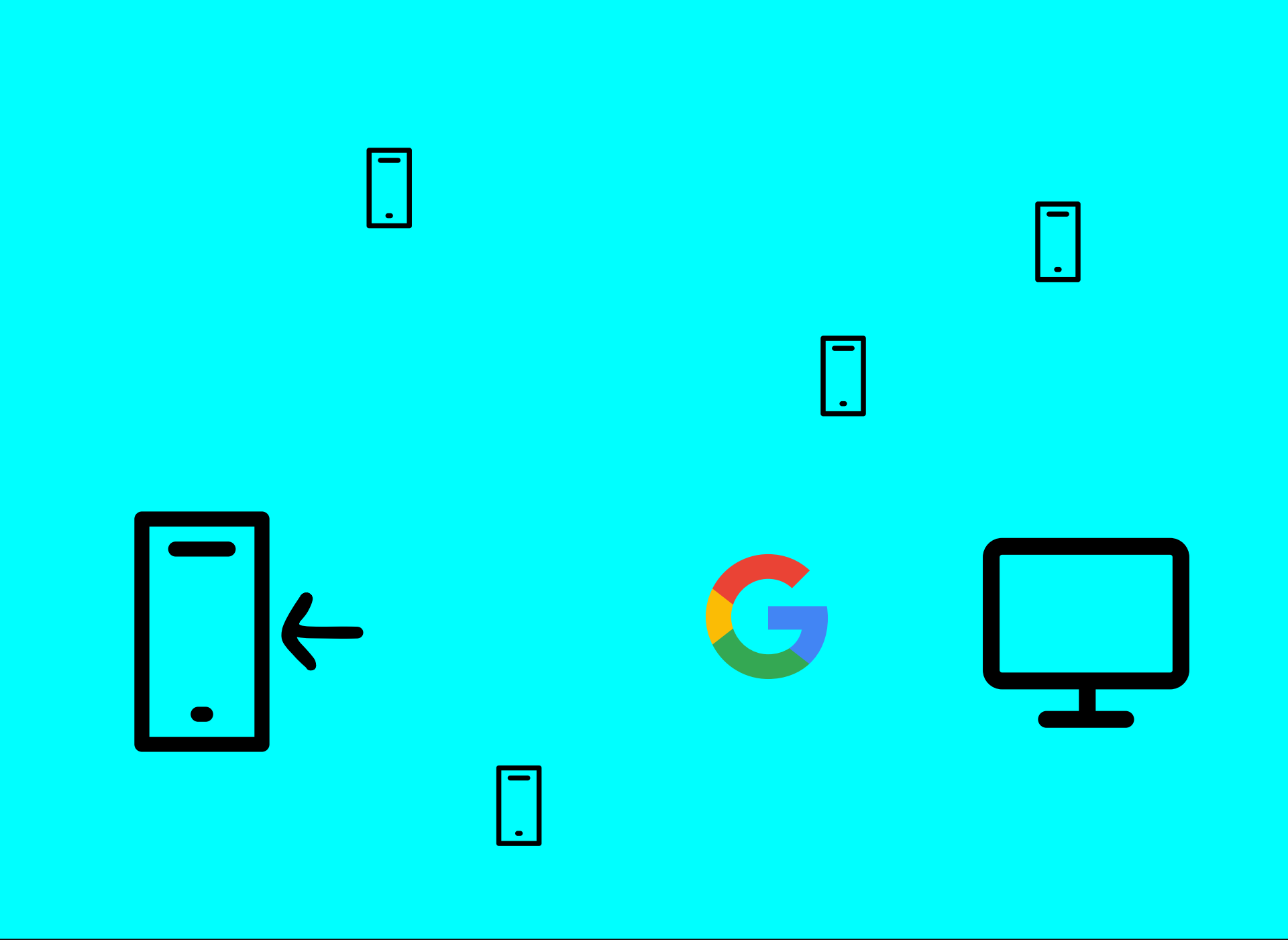


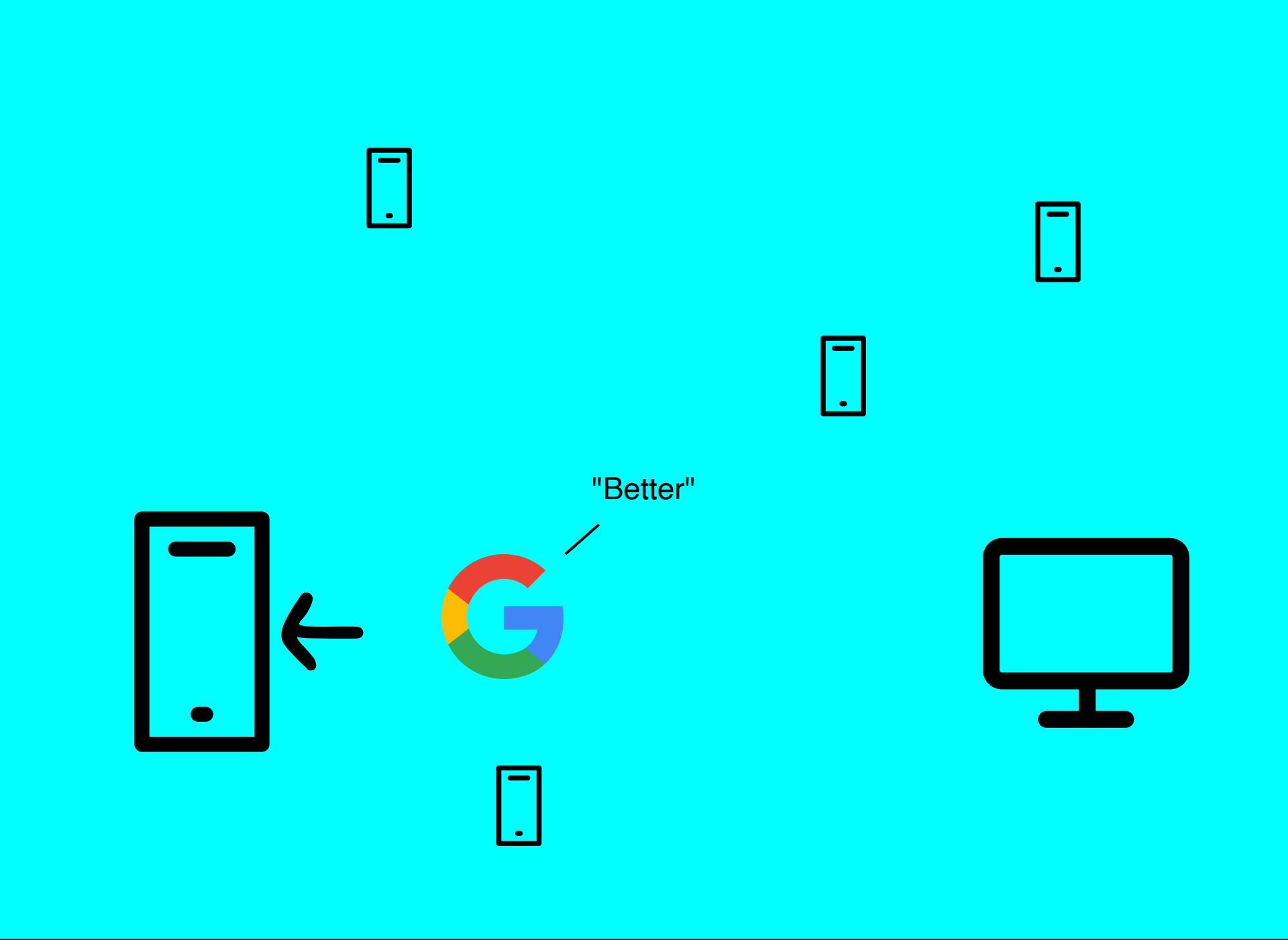


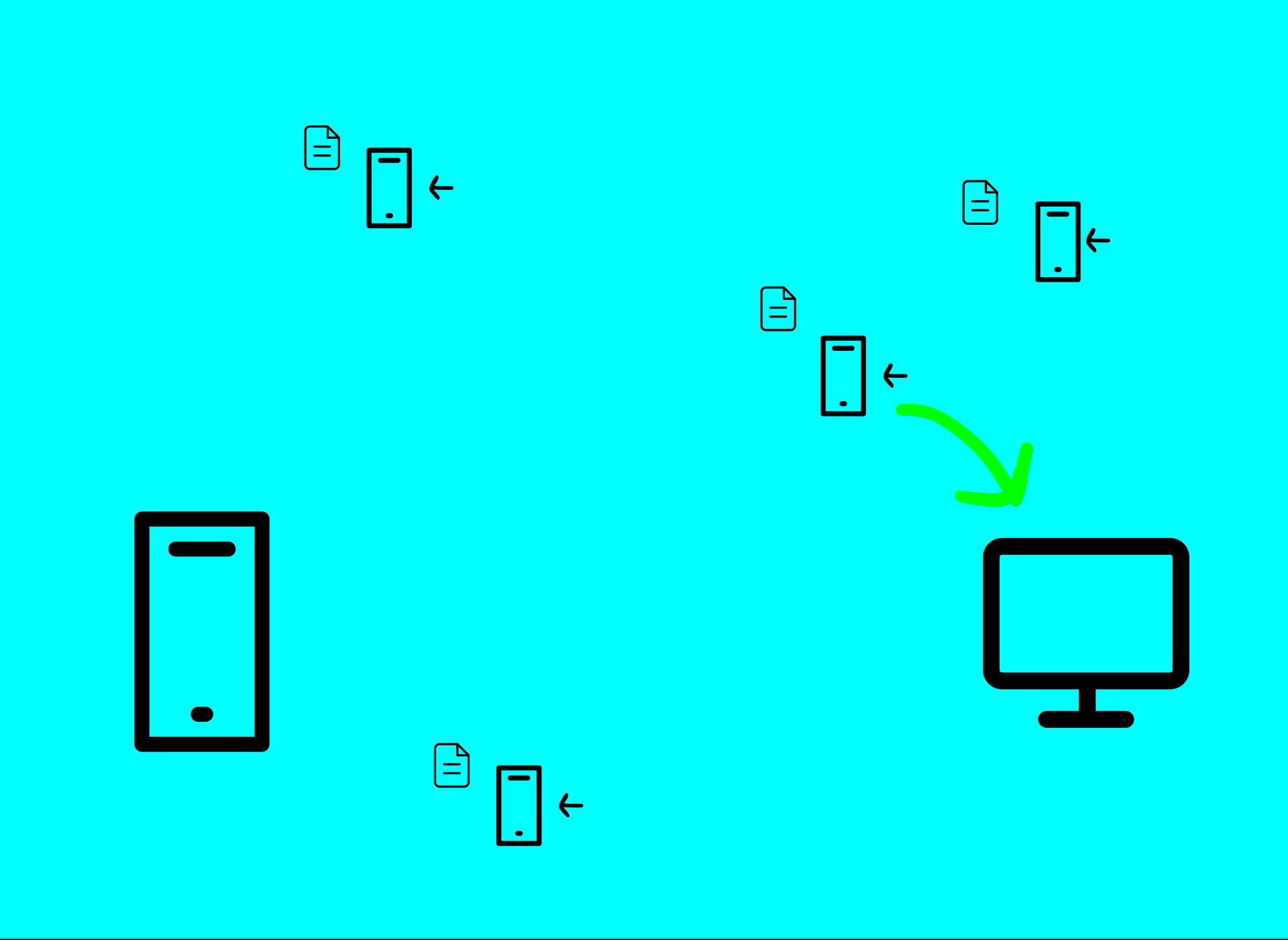





"What the hell is that?"



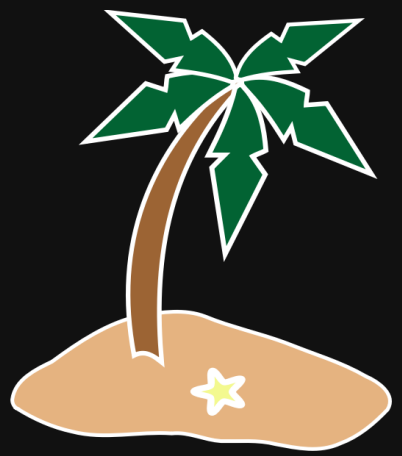

"Better"



So SSR vs SPA becomes...

- Are we doing work on request or ahead of time
- where do we do the work
- do we have to access a database
- do we care about SEO?
- How often do we update which files?
- etc...

Where the
Islands Architecture
fits in

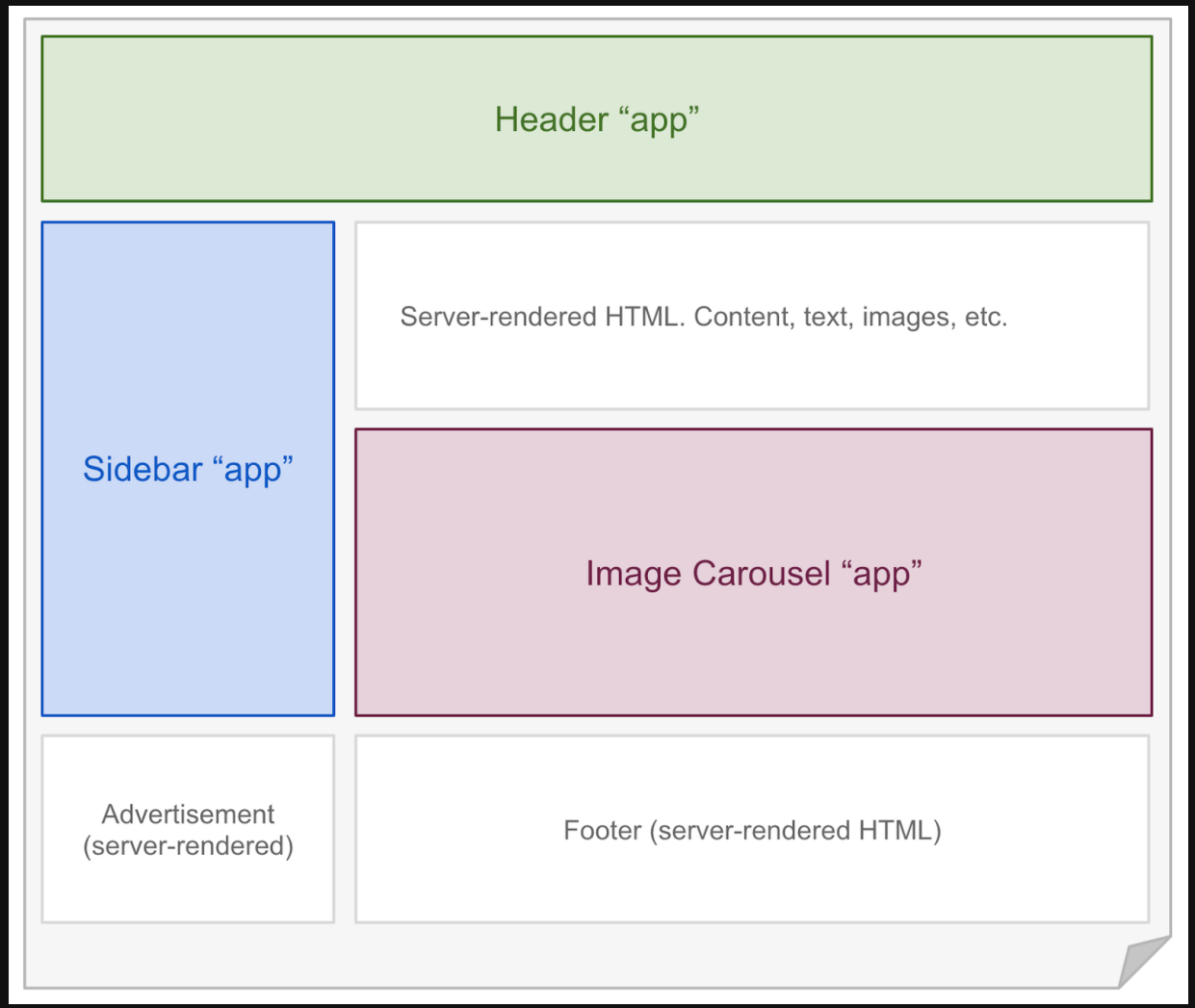


How Astro can help

astro

DOCS

astro.build


Header “app”
Sidebar “app”
Server-rendered HTML. Content, text, images, etc.
Image Carousel “app”
Advertisement (server-rendered)
Footer (server-rendered HTML)


The Astro
Component files
...and file based routing

index.astro

```astro
---
import {format} from 'date-fns';
const builtAt: Date = new Date();
const builtAtFormatted = format(builtAt, 'MMMM dd, yyyy -- H:mm:ss.SSS');
---
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Astro Playground</title>
    <style>
      header {
        display: flex;
        flex-direction: column;
        align-items: center;
        text-align: center;
        margin-top: 15vh;
        font-family: Arial;
      }
    </style>
  </head>
  <body>
    <header>
      <img width="60" height="80" src="https://bestofjs.org/logos/astro.svg" alt="Astro log
      <h1>Hello, Astro!</h1>
    </header>
```

MyComponent.astro

```
---
import { Markdown } from 'astro/components';
---
<Markdown>
  # Markdown syntax is now supported! **Yay!**
</Markdown>
```

Movies.astro

```
---
import { MovieList } from ':components/MovieList.jsx'
const response = await fetch('https://example.com/movies.json');
const data = await response.json();
---
<!-- Output the result to the page -->
<MovieList movies={data} />
```


Opt-In to interactivity
...and use any framework

Movies.astro

```
---
import { MovieList } from ':components/MovieList.jsx'
const response = await fetch('https://example.com/movies.json');
const data = await response.json();
---
<!-- Output the result to the page -->
<MovieList movies={data} />
```

Movies.astro

```
---
import { MovieList } from ':components/MovieList.jsx'
const response = await fetch('https://example.com/movies.json');
const data = await response.json();
---
<!-- Output the result to the page -->
<MovieList client:load movies={data} />
```

client:*

<MyComponent client:load />

Start importing the component JS at page load.
Hydrate the component when import completes

<MyComponent client:idle />

Start importing the component JS as soon as main thread is free. Hydrate the component when import completes.

<MyComponent client:visible />

Start importing the component JS as soon as the element enters the viewport. Hydrate the component when import completes. Useful for content lower down on the page

<MyComponent client:media={QUERY} />

Start importing the component JS as soon as the browser matches the given media query. Hydrate the component when import completes

<MyComponent client:only />

Start importing the component JS at page load and hydrate when the import completes, similar to client:load. The component will be skipped at build time

Other Cool Features

...Markdown, Code Blocks, RSS and more

Astro and the Island Architecture

@adamlbarrett

Anedot