SERVER-SIDE RENDERING ISN'T ENOUGH

MATTHEW PHILLIPS

donejs

GITHUB.COM/CANJS/CAN-SSR

TERMS

- Shared codebase
- Isomorphic
- Universal

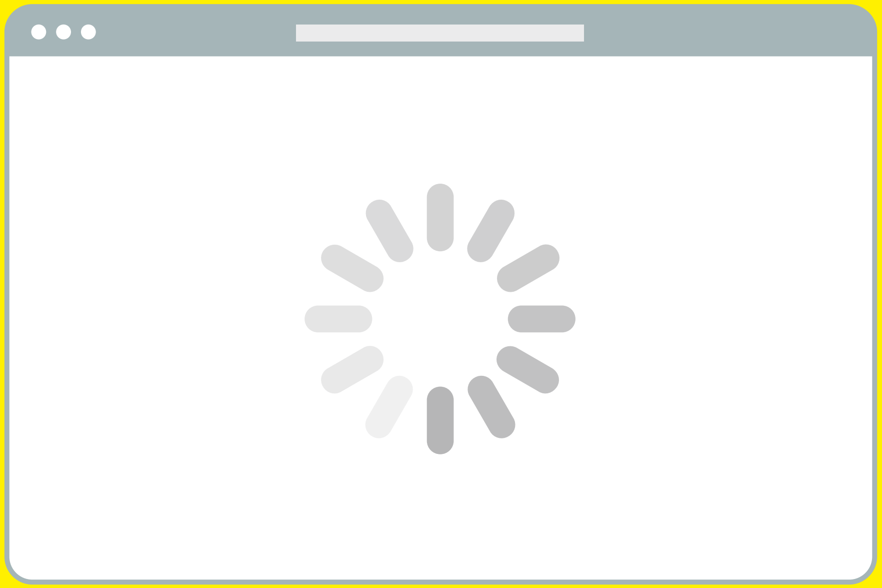
WHY BOTHER

Perceived performance: no one likes staring at a spinner.

SEO: if you care about that sort of thing, it helps. Not every bot is Googlebot.

BLING BLING: Amazon reports that conversion increased by 1% for every 100ms improvement.

This sucks



THE STATE OF SERVER RENDERING

EVERYTHING SHOULD BE SERVER RENDERED

REQUIREMENTS

PERFORMANCE

- Rendering speed
- Only includes the assets needed (CSS and JavaScript)
- Prevents unnecessary requests in the client

MAINTAINABILITY

- Shared router
- Asynchronous rendering
- Fast development experience with hot module swapping

RENDERING PERFORMANCE

HEADLESS BROWSER

PhantomJS

- Consumed a lot of memory
- Needed pooling
- Very fast

VIRTUAL DOMS

- Run the same code on the client and server
- Run within a single Node context
- Rendering is usually synchronous

CAN-SSR'S VDOM

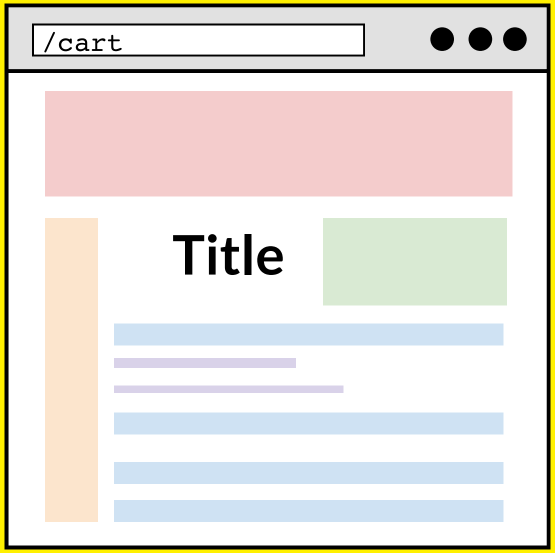
Looks like a real DOM, only the basics

DEMO COMPATIBILITY

MINIMIZING REQUEST SIZE

TRADITIONAL METHOD

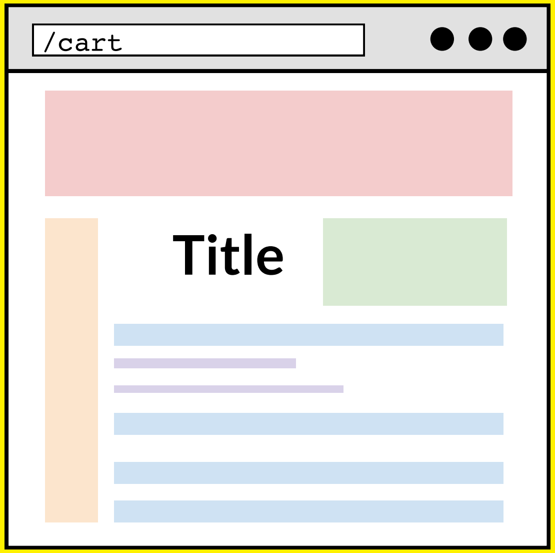
CSS LOADED IN JAVASCRIPT



1. Initially unstyled
2. Main, site-wide style is loaded
3. Page specific style is loaded progressively.

WITH SERVER TEMPLATE

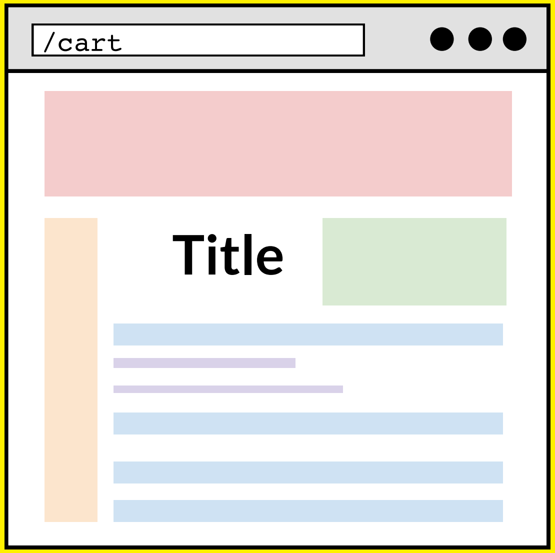
ADDING CSS MANUALLY



1. Initially partially styled; main CSS is included, most of the page-specific CSS.
2. Rest of page-specific styles are added.

WITH DONEJS

CAN-SSR DOES IT FOR YOU



1. All styles needed for the page are included directly in the head.
   - And only the styles needed for the page.

COMPONENT-BASED ARCHITECTURE

```
import Framework from 'fancy-framework';
import './styles.scss';

...
```

MINIMIZING THE NUMBER OF REQUESTS

PREVENT REDUNDANT REQUESTS

- Embed responses into the rendered page.
- Can be reused on the client to do initial rendering.

```
<script>
  INLINE_CACHE = {"users": [{ ... } ] };
</script>
```

SHARED CODE-BASE

HOW MUCH CODE IS SHARED?

MINIMIZING DIFFERENCES FOR EASIER MAINTAINENCE

- The "main" can run on both client and in Node.
- A shared router, not adding new routes in separate places

EXAMPLE

Middleware

```
var ssr = require("can-ssr/middleware");
var app = require("express")();

app.use(ssr());
```

Core API

```
var ssr = require("can-ssr");

var render = ssr();

render("/cart").then(function(result){
  console.log(result.html);
});
```

ASYNCHRONOUS RENDERING

SYNCHRONOUS RENDERING

- Forces all data to be present before rendering.
- Cannot use component-based architecture.
- Pushes application logic into another layer.
- Makes writing reusable components harder.

```
// server.js
import render from "framework-dom";

app.get("/cart", function(req, res){

    fetchCart().then(function(data){
        res.send(
            render(data)
        );
    });

});
```

```
// cart.js
import Component from "fancy-framework";

class Cart extends Component {
    render() {
        let data = this.props.data;

        return <div> ... </div>
    }
}
```

DEMO ASYNCHRONOUS REACT

HTTPS://GITHUB.COM/CANJS/CAN-WAIT

INSTANT DEV WORKFLOW

HOT MODULE REPLACEMENT

DEMO DONEJS LIVE-RELOAD

THE END

BY MATTHEW PHILLIPS