Next.js for Mobile

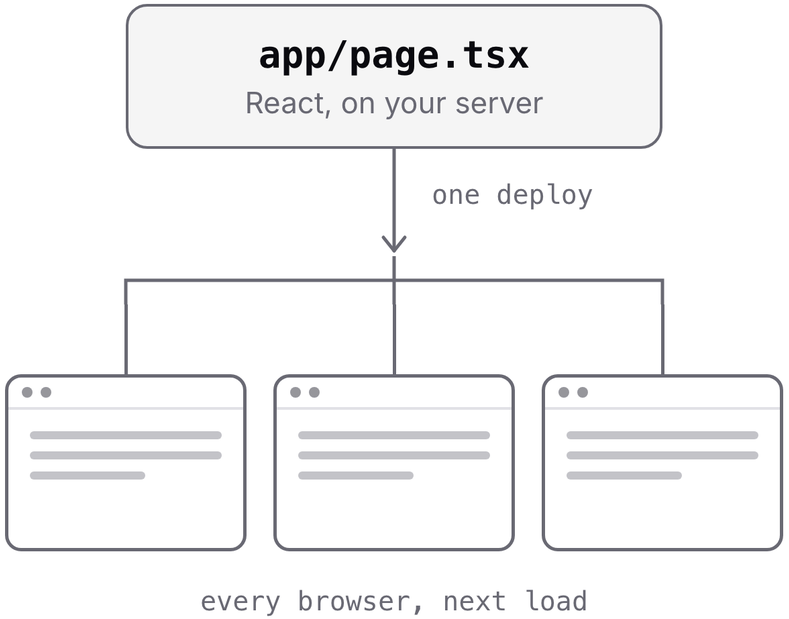
Server-Driven React Native

Jean-François Bisson · React Summit 2026

Backtick: Next.js for mobile

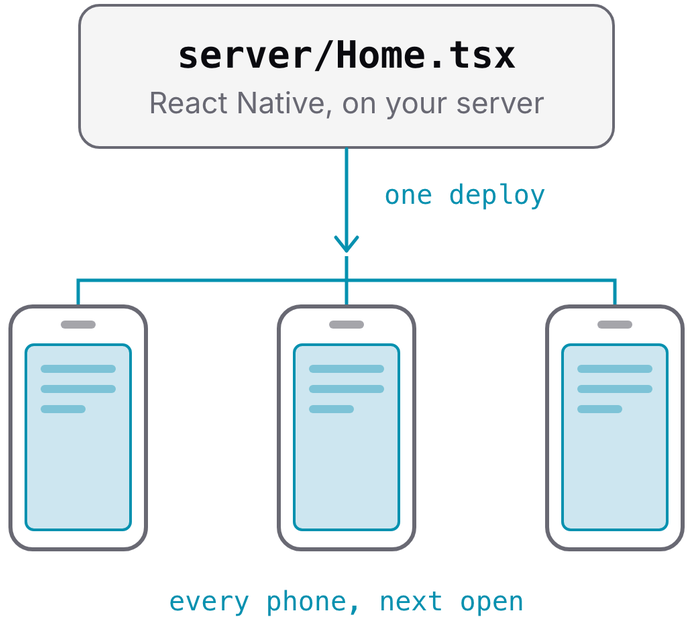
Open source, MIT licensed.

Next.js



Edit. Deploy. Everyone has it.

Backtick

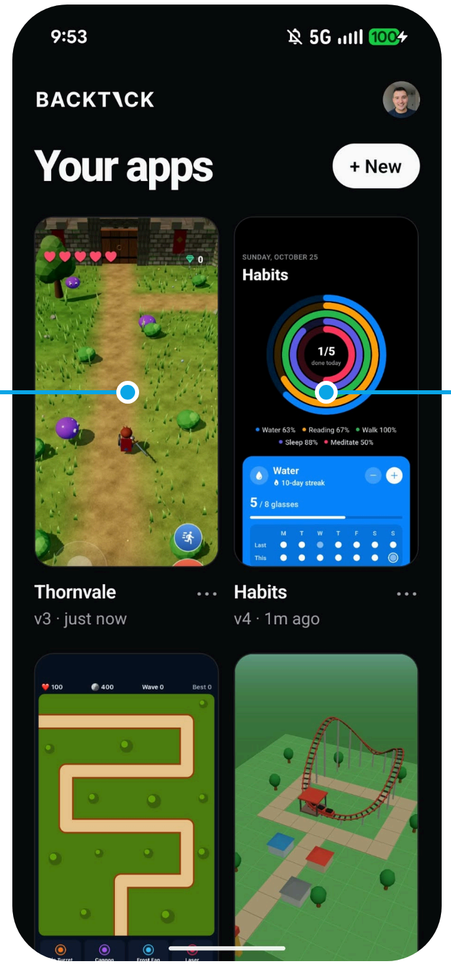


Edit. Deploy. Everyone has it.

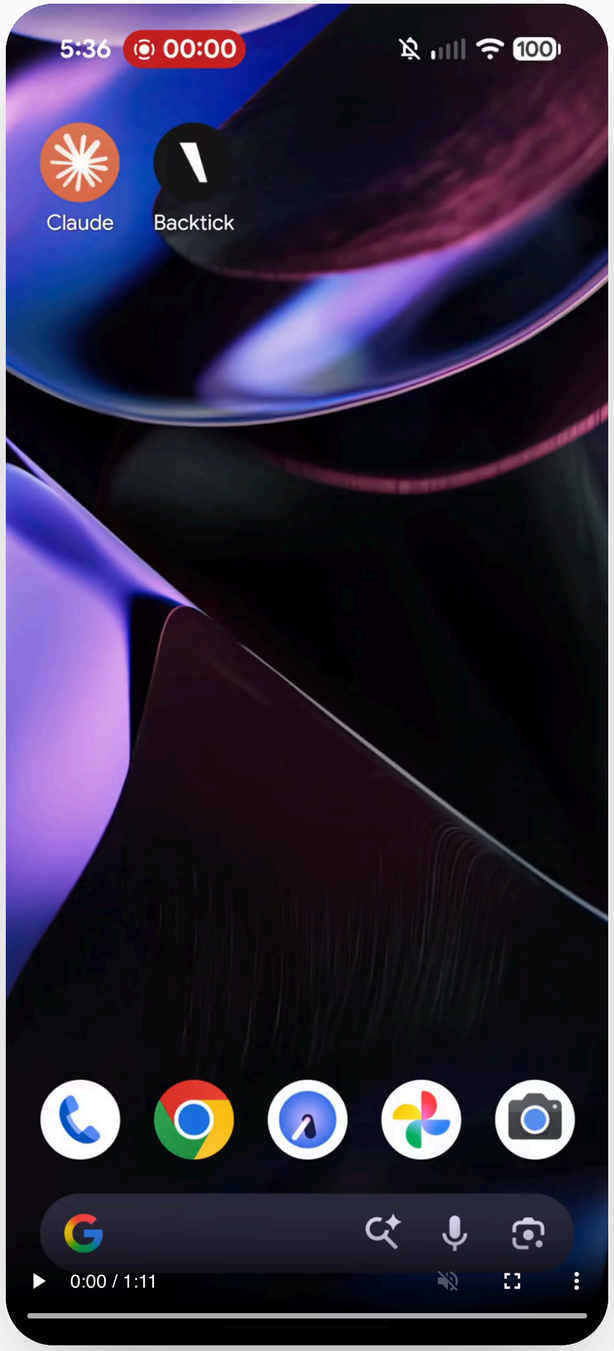
Showcase app

Every screen is native. Every screen comes from your server.

thornvale.backtickjs.app



habits.backtickjs.app


5:36
00:00
100
Claude
Backtick
0:00 / 1:11

A Backtick app

server/App.tsx — SIMPLIFIED

```
import { cs } from "@backtickjs/core";
import { Text } from "react-native" with { backtick: "client" };
import { useState } from "react" with { backtick: "client" };

const Counter = cs`({ from }: { from: number }) => {
  const [count, setCount] = $useState(from);
  const onPress = () => setCount(count + 1);
  return <$Text onPress={onPress}>{count}</$Text>;
}`;

app.get("/counter/:from", (c) => {
  const from = Number(c.req.param("from"));
  return c.render(cs`<$Counter from={$from} />`);
});
```

A Backtick app

server/App.tsx — SIMPLIFIED

```
import { cs } from "@backtickjs/core";
import { Text } from "react-native" with { backtick: "client" };
import { useState } from "react" with { backtick: "client" };

const Counter = cs`({ from }: { from: number }) => {
  const [count, setCount] = $useState(from);
  const onPress = () => setCount(count + 1);
  return <$Text onPress={onPress}>{count}</$Text>;
}`;

app.get("/counter/:from", (c) => {
  const from = Number(c.req.param("from"));
  return c.render(cs`<$Counter from={$from} />`);
});
```

A Backtick app

server/App.tsx — SIMPLIFIED

```
import { cs } from "@backtickjs/core";
import { Text } from "react-native" with { backtick: "client" };
import { useState } from "react" with { backtick: "client" };

const Counter = cs`({ from }: { from: number }) => {
  const [count, setCount] = $useState(from);
  const onPress = () => setCount(count + 1);
  return <$Text onPress={onPress}>{count}</$Text>;
}`;

app.get("/counter/:from", (c) => {
  const from = Number(c.req.param("from"));
  return c.render(cs`<$Counter from={$from} />`);
});
```

~~"use client"~~

A Backtick app

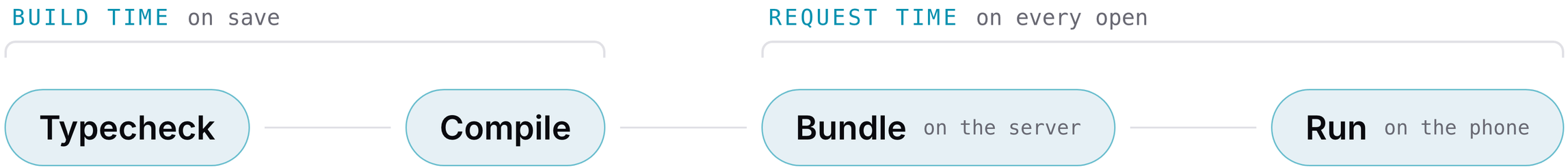
server/App.tsx — SIMPLIFIED

```
import { cs } from "@backtickjs/core";
import { Text } from "react-native" with { backtick: "client" };
import { useState } from "react" with { backtick: "client" };

const Counter = cs`({ from }: { from: number }) => {
  const [count, setCount] = $useState(from);
  const onPress = () => setCount(count + 1);
  return <$Text onPress={onPress}>{count}</$Text>;
}`;

app.get("/counter/:from", (c) => {
  const from = Number(c.req.param("from"));
  return c.render(cs`<$Counter from={$from} />`);
});
```

How a screen is made



Typechecked at build time

TypeScript checks every $ against the real types.

server/App.tsx

```
const Counter = cs`({ from }) => {
  const [count, setCount] = $useState(from);
  return <$Text>{count}</$Text>;
}`;
```

What TypeScript sees — SIMPLIFIED

```
const Counter = cs.lift(({ from }) => {
  const [count, setCount] = cs.splice(useState)(from);
  return <Text>{count}</Text>;
});
```

Compiled at build time

The script becomes a string. Each `$` becomes a parameter, filled in from `args`.

server/App.tsx

```tsx
const Counter = cs`({ from }) => {
  const [count, setCount] = $useState(from);
  return <$Text>{count}</$Text>;
}`;
```

server/App.js — SIMPLIFIED

```js
const Counter = {
  code: `(useState, Text) => ({ from }) => {
    const [count, setCount] = useState(from);
    return jsx(Text, { children: count });
  }`,
  args: [useState, Text],
};
```

Bundled at request time

Each request gets one bundle with only that screen's code.

server/App.tsx

```
function Counter({ user }) {
  if (isInExperiment(user)) {
    return cs`<$CounterV2 />`;
  } else {
    return cs`<$CounterV1 />`;
  }
}
```

Fetched on every open

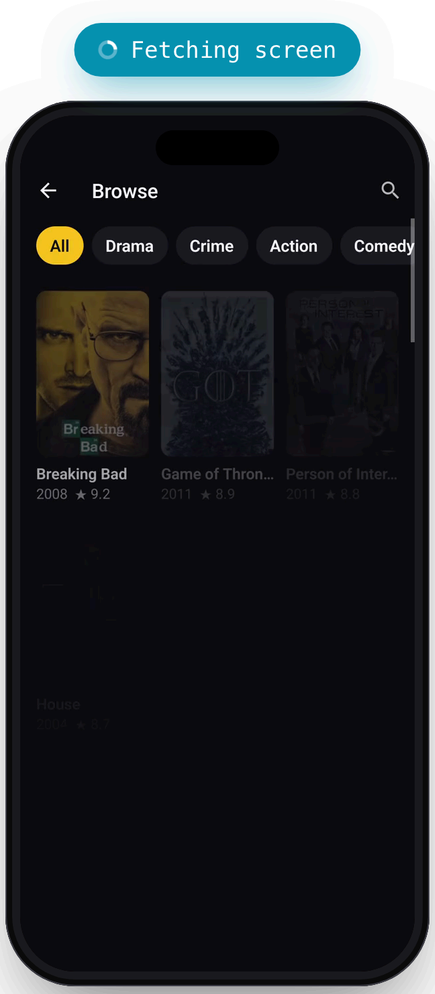
Layouts and loading screens are prefetched, so navigation feels instant.

client/App.tsx — SIMPLIFIED

```tsx
const response = await fetch("tvshows.backtickjs.app" + path);
```

Network

```
GET   /                200
GET   /show/82         pending
```



Data: TVmaze (CC BY-SA)

Backtick might be for you if...

You want to ship at the speed of the web.	RELEASE SPEED
Your WebView screens don't feel native.	UX QUALITY
Your server-driven UI keeps hitting limits.	UI EXPRESSIVENESS

Thank you

Read the code



Say hi

linkedin.com/in/jfbisson

Try the iOS app