Rendering

Client and server rendering with React Native for Web.

React Native for Web can be used for multi-platform and web-only applications. It can incrementally adopted by existing React Web apps and integrated with existing React Native apps. Preact is also supported.

React Native for Web components interoperate with React DOM components. They can be incrementally introduced at any point in an application's component tree. One thing to be aware of is that external CSS applied to *all* tags in a document may interfere with the default rendering of some React Native for Web components.

Client API

Render apps using AppRegistry:

```
// index.web.js

import { AppRegistry } from 'react-native';
import App from './src/App';

// register the app
AppRegistry.registerComponent('App', () => App);

AppRegistry.runApplication('App', {
  initialProps: {},
  rootTag: document.getElementById('root')
});
```

Or render individual components:

```
import { render } from 'react-native';
import Header from './src/Header';

render(<Header />, document.getElementById('header'))
```

You might need to adjust the styles of the HTML document’s root elements for your app to fill the viewport.

```
<html style="height:100%">
  <body style="height:100%">
    <div id="root" style="display:flex;height:100%"></div>
```

> **Warning!** Although components in can be rendered by calling `ReactDOM.render` directly, this will not set the global context React Native provides to components when using the `AppRegistry` API.

Server API

Server-side rendering to HTML is supported using `AppRegistry`:

```
import App from './src/App';
import ReactDOMServer from 'react-dom/server';
import { AppRegistry } from 'react-native-web';

// register the app
AppRegistry.registerComponent('App', () => App);

// prerender the app
const { element, getStyleElement } = AppRegistry.getApplication('App', { initialI
// first the element
const html = ReactDOMServer.renderToString(element);
```

```
// then the styles (optionally include a nonce if your CSP policy requires it)
const css = ReactDOMServer.renderToStaticMarkup(getStyleElement({ nonce }));

// example HTML document string
const document = `
<!DOCTYPE html>
<html style="height:100%">
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
${css}
<body style="height:100%; overflow-y:hidden">
<div id="root" style="display:flex; height: 100%">
${html}
</div>
<script nonce="${nonce}" src="${bundlePath}"></script>
`
```

Updated July 20, 2023   Edit