Styling

Because a document without styles would be very boring, react-pdf ships a powerful styling solution using CSS and Flexbox.

StyleSheet API

React-pdf also sticks with the primitives specs when it comes to styling.

StyleSheet.create()

Create a stylesheet. This method expects a valid JS object as only argument (containing as much css definitions as you want) and returns an object that you can pass down to components via the `style` prop

```
import React from 'react';
import { Page, Text, View, Document, StyleSheet } from '@react-pdf/renderer';

const styles = StyleSheet.create({
  page: { backgroundColor: 'tomato' },
  section: { color: 'white', textAlign: 'center', margin: 30 }
});

const doc = (
  <Document>
    <Page size="A4" style={styles.page}>
      <View style={styles.section}>
        <Text>Section #1</Text>
      </View>
    </Page>
  </Document>
);

ReactPDF.render(doc);
```

See it in action →

Inline styling

There's no need to call `StyleSheet.create` in order to style components. You can also just pass a plain JS object to the `style` prop and react-pdf will get the job done.

```
import React from 'react';
import { Page, Text, View, Document } from '@react-pdf/renderer';

const MyDocument = () => (
  <Document>
    <Page size="A4" style={{ backgroundColor: 'tomato' }}>
      <View style={{ color: 'white', textAlign: 'center', margin: 30 }}>
        <Text>Section #1</Text>
      </View>
    </Page>
  </Document>
);
```

See it in action →

Mixing both solutions

The `style` prop also accepts an Array as value, containing any possible combination of the last two alternatives

```jsx
import React from 'react';
import { Page, Text, View, Document, StyleSheet } from '@react-pdf/renderer';

const styles = StyleSheet.create({
  page: { backgroundColor: 'tomato' },
  section: { textAlign: 'center', margin: 30 }
});

const MyDocument = () => (
  <Document>
    <Page size="A4" style={styles.page}>
      <View style={[styles.section, { color: 'white' }]}>
        <Text>Section #1</Text>
      </View>
    </Page>
  </Document>
);
```

> **Protip:** This can be useful when you want to apply both predefined styles, and styles based on props

See it in action →

Media queries

There may be times in which you'll need to apply different styles based on the document context. For that, we provide media-queries support (just as you would do it for the web!). You can query based on both `width` and `height` (min and max), and also `orientation`:

```
import React from 'react';
import { Page, Text, View, Document, StyleSheet } from '@react-pdf/renderer';

const styles = StyleSheet.create({
  section: {
    width: 200,
    '@media max-width: 400': {
      width: 300,
    },
    '@media orientation: landscape': {
      width: 400,
    },
  }
});

const MyDocument = () => (
  <Document>
    <Page size="A4" style={styles.page}>
      <View style={styles.section}>
        <Text>Section #1</Text>
      </View>
    </Page>
  </Document>
);
```

See it in action →

Styled-components

Looking for a more neat way of styling your document? Now you can take advantage of the entire styled-components API inside your PDF documents!

Install

First, you should install styled-components binding:

```
yarn add @react-pdf/styled-components
```

How to use

This binding follows exactly the same styled-components API, so after installing it you can start creating styled primitives by importing `styled` object from it:

```
import styled from '@react-pdf/styled-components';

const Heading = styled.Text`
  margin: 10px;
  font-size: 22px;
  font-family: 'Helvetica';
`;
```

```
const MyDocument = () => (
  <Document>
    <Page>
      <Heading>
        D'oh!
      </Heading>
    </Page>
  </Document>
);
```

See it in action →

Note: `@react-pdf/styled-components` it's a separate new `styled-components` build, so you shouldn't install the latter package in your project

For more information about the API, please refer to the styled-components documentation.

Valid units

`pt` *(default. Based on the standard 72 dpi PDF document)*

`in` inches

`mm` millimeters

`cm` centimeters

`%` percentage

`vw` viewport/page width

`vh` viewport/page height

Valid CSS properties

Flexbox

- alignContent
- alignItems
- alignSelf
- flex
- flexDirection
- flexWrap
- flexFlow
- flexGrow
- flexShrink
- flexBasis
- justifyContent
- order

Layout

- bottom
- display
- left
- position
- right
- top

Dimension

- height
- maxHeight
- maxWidth
- minHeight
- minWidth
- width

Color

- backgroundColor
- color
- opacity

Text

- fontSize
- fontFamily
- fontStyle
- fontWeight
- letterSpacing
- lineHeight
- maxLines
- textAlign
- textDecoration
- textDecorationColor
- textDecorationStyle
- textIndent
- textOverflow
- textTransform

Sizing/positioning

- object-fit
- object-position

Margin/padding

- margin
- marginHorizontal
- marginVertical
- marginTop
- marginRight
- marginBottom
- marginLeft
- padding
- paddingHorizontal
- paddingVertical
- paddingTop
- paddingRight
- paddingBottom
- paddingLeft

Transformations

- transform:rotate
- transform:scale
- transform:scaleX
- transform:scaleY
- transform:translate
- transform:translateX

- transform:translateY
- transform:matrix
- transformOrigin

Borders

- border
- borderColor
- borderStyle
- borderWidth
- borderTop
- borderTopColor
- borderTopStyle
- borderTopWidth
- borderRight
- borderRightColor
- borderRightStyle
- borderRightWidth
- borderBottom
- borderBottomColor
- borderBottomStyle
- borderBottomWidth
- borderLeft
- borderLeftColor
- borderLeftStyle
- borderLeftWidth
- borderTopLeftRadius
- borderTopRightRadius

- borderBottomRightRadius
- borderBottomLeftRadius

← Components

Advanced →