React Native Styling Cheat Sheet

Most of the React Native styling material in one page. Imported from the official docs.

Contents

General

Components

Flexbox

Name	Type	Default	Description
alignContent	oneOf `flex-start`, `flex-end`, `center`, `stretch`, `space-between`, `space-around`		`alignContent` controls how rows align in the cross direction, overriding the `alignContent` of the parent. See https://developer.mozilla.org/en-US/docs/Web/CSS/align-content for more details.

Name	Type	Default	Description
alignItems	oneOf `flex-start`, `flex-end`, `center`, `stretch`, `baseline`	stretch	`alignItems` aligns children in the cross direction. For example, if children are flowing vertically, `alignItems` controls how they align horizontally. It works like `align-items` in CSS, except the default value is `stretch` instead of `flex-start`. See https://css-tricks.com/almanac/properties/a/align-items/ for more detail.
alignSelf	oneOf `auto`, `flex-start`, `flex-end`, `center`, `stretch`, `baseline`	auto	controls how a child aligns in the cross direction, overriding the `alignItems` of the parent. It works like `align-self` in CSS. See https://css-tricks.com/almanac/properties/a/align-self/ for more detail.
aspectRatio	number		`aspectRatio` controls the size of the undefined dimension of a node. `aspectRatio` is a non-standard property only available in React Native and not CSS. On a node with a set `width`/`height` `aspectRatio` controls the size of the unset dimension. On a node with a set `flexBasis` `aspectRatio` controls the size of the node in the cross axis if unset. On a node with a `measure` function `aspectRatio` works as though the `measure` function measures the `flexBasis`. On a node with `flexGrow`/`flexShrink` `aspectRatio` controls the size of the node in the cross axis if unset. `aspectRatio` takes min/max dimensions into account.
borderBottomWidth	number	0	`borderBottomWidth` works like `border-bottom-width` in CSS. See http://www.w3schools.com/cssref/pr_border-bottom_width.asp for more details.
borderLeftWidth	number	0	`borderLeftWidth` works like `border-left-width` in CSS. See http://www.w3schools.com/cssref/pr_border-bottom_width.asp for more details.
borderRightWidth	number	0	`borderRightWidth` works like `border-right-width` in CSS. See http://www.w3schools.com/cssref/pr_border-right_width.asp for more details.

Name	Type	Default	Description
borderTopWidth	number	0	`borderTopWidth` works like `border-top-width` in CSS. See http://www.w3schools.com/cssref/pr_border-top_width.asp for more details.
borderEndWidth	number	0	When direction is ltr, `borderEndWidth` is equivalent to `borderRightWidth`. When direction is rtl, `borderEndWidth` is equivalent to `borderLeftWidth`.
borderStartWidth	number	0	When direction is ltr, `borderStartWidth` is equivalent to `borderLeftWidth`. When direction is rtl, `borderStartWidth` is equivalent to `borderRightWidth`.
borderWidth	number	0	`borderWidth` works like `border-width` in CSS. See http://www.w3schools.com/cssref/pr_border-width.asp for more details.
bottom	number	auto*	`bottom` is the number of logical pixels to offset the bottom edge of this component. It works similarly to `bottom` in CSS, but in React Native you must use logical pixel units, rather than percents, ems, or any of that. See https://developer.mozilla.org/en-US/docs/Web/CSS/bottom for more details of how `top` affects layout.
direction	oneOf `inherit`, `ltr`, `rtl`	`inherit`	`direction` specifies the directional flow of the user interface. The default is `inherit`, except for root node which will have value based on the current locale. See https://facebook.github.io/yoga/docs/rtl/ for more details.
display	oneOf `none`, `flex`	`flex`	`display` sets the display type of this component. It works similarly to `display` in CSS, but only support 'flex' and 'none'.
end	number	auto*	When the direction is `ltr`, `end` is equivalent to `right`. When the direction is `rtl`, `end` is equivalent to `left`. This style takes precedence over the `left` and `right` styles.
start	number	auto*	When the direction is `ltr`, `start` is equivalent to `left`. When the direction is `rtl`, `start` is equivalent to `right`. This style takes precedence over the `left`, `right`, and `end` styles.

Name	Type	Default	Description
flex	number	0	In React Native `flex` does not work the same way that it does in CSS. `flex` is a number rather than a string, and it works according to the `css-layout` library at https://github.com/facebook/css-layout . When `flex` is a positive number, it makes the component flexible and it will be sized proportional to its flex value. So a component with `flex` set to 2 will take twice the space as a component with `flex` set to 1. When `flex` is 0, the component is sized according to `width` and `height` and it is inflexible. When `flex` is -1, the component is normally sized according `width` and `height`. However, if there's not enough space, the component will shrink to its `minWidth` and `minHeight`. `flexGrow`, `flexShrink` and `flexBasis` work the same as in CSS.
flexDirection	oneOf `row`, `row-reverse`, `column`, `column-reverse`	column	`flexDirection` controls which directions children of a container go. `row` goes left to right, `column` goes top to bottom, and you may be able to guess what the other two do. It works like `flex-direction` in CSS, except the default is `column`. See https://css-tricks.com/almanac/properties/f/flex-direction/ for more detail.
flexBasis	number	0	
flexGrow	number	0	
flexShrink	number	0	
flexWrap	oneOf `wrap`, `nowrap`	nowrap	`flexWrap` controls whether children can wrap around after they hit the end of a flex container. It works like `flex-wrap` in CSS. See https://css-tricks.com/almanac/properties/f/flex-wrap/ for more detail.
height	number	auto*	`height` sets the height of this component. It works similarly to `height` in CSS, but in React Native you must use logical pixel units, rather than percents, ems, or any of that. See http://www.w3schools.com/cssref/pr_dim_width.asp for more details.

Name	Type	Default	Description
justifyContent	oneOf `flex-start`, `flex-end`, `center`, `space-between`, `space-around`	flex-start	`justifyContent` aligns children in the main direction. For example, if children are flowing vertically, `justifyContent` controls how they align vertically. It works like `justify-content` in CSS. See https://css-tricks.com/almanac/properties/j/justify-content/ for more detail.
left	number	auto*	`left` is the number of logical pixels to offset the left edge of this component. It works similarly to `left` in CSS, but in React Native you must use logical pixel units, rather than percents, ems, or any of that. See https://developer.mozilla.org/en-US/docs/Web/CSS/left for more details of how `left` affects layout.
margin	number	0	Setting `margin` has the same effect as setting each of `marginTop`, `marginLeft`, `marginBottom`, and `marginRight`.
marginBottom	number	0	`marginBottom` works like `margin-bottom` in CSS. See http://www.w3schools.com/cssref/pr_margin-bottom.asp for more details.
marginHorizontal	number	0	Setting `marginHorizontal` has the same effect as setting both `marginLeft` and `marginRight`.
marginLeft	number	0	`marginLeft` works like `margin-left` in CSS. See http://www.w3schools.com/cssref/pr_margin-left.asp for more details.
marginRight	number	0	`marginRight` works like `margin-right` in CSS. See http://www.w3schools.com/cssref/pr_margin-right.asp for more details.
marginTop	number	0	`marginTop` works like `margin-top` in CSS. See http://www.w3schools.com/cssref/pr_margin-top.asp for more details.
marginVertical	number	0	Setting `marginVertical` has the same effect as setting both `marginTop` and `marginBottom`.
marginEnd	number	0	When direction is `ltr`, `marginEnd` is equivalent to `marginRight`. When direction is `rtl`, `marginEnd` is equivalent to `marginLeft`.

Name	Type	Default	Description
marginStart	number	0	When direction is `ltr`, `marginStart` is equivalent to `marginLeft`. When direction is `rtl`, `marginStart` is equivalent to `marginRight`.
maxHeight	number	auto*	`maxHeight` is the maximum height for this component, in logical pixels. It works similarly to `max-height` in CSS, but in React Native you must use logical pixel units, rather than percents, ems, or any of that. See http://www.w3schools.com/cssref/pr_dim_max-height.asp for more details.
maxWidth	number	auto*	`maxWidth` is the maximum width for this component, in logical pixels. It works similarly to `max-width` in CSS, but in React Native you must use logical pixel units, rather than percents, ems, or any of that. See http://www.w3schools.com/cssref/pr_dim_max-width.asp for more details.
minHeight	number	auto*	`minHeight` is the minimum height for this component, in logical pixels. It works similarly to `min-height` in CSS, but in React Native you must use logical pixel units, rather than percents, ems, or any of that. See http://www.w3schools.com/cssref/pr_dim_min-height.asp for more details.
minWidth	number	auto*	`minWidth` is the minimum width for this component, in logical pixels. It works similarly to `min-width` in CSS, but in React Native you must use logical pixel units, rather than percents, ems, or any of that. See http://www.w3schools.com/cssref/pr_dim_min-width.asp for more details.
padding	number, string	0	`padding` works like `padding` in CSS. It's like setting each of `paddingTop`, `paddingBottom`, `paddingLeft`, and `paddingRight` to the same thing. See http://www.w3schools.com/css/css_padding.asp for more details.
paddingBottom	number, string	0	`paddingBottom` works like `padding-bottom` in CSS. See http://www.w3schools.com/cssref/pr_padding-bottom.asp for more details.
paddingHorizontal	number, string	0	Setting `paddingHorizontal` is like setting both of `paddingLeft` and `paddingRight`.
paddingLeft	number, string	0	`paddingLeft` works like `padding-left` in CSS. See http://www.w3schools.com/cssref/pr_padding-left.asp for more details.

Name	Type	Default	Description
paddingRight	number, string	0	`paddingRight` works like `padding-right` in CSS. See http://www.w3schools.com/cssref/pr_padding-right.asp for more details.
paddingTop	number, string	0	`paddingTop` works like `padding-top` in CSS. See http://www.w3schools.com/cssref/pr_padding-top.asp for more details.
paddingVertical	number, string	0	Setting `paddingVertical` is like setting both of `paddingTop` and `paddingBottom`.
paddingEnd	number, string	0	When direction is `ltr`, `paddingEnd` is equivalent to `paddingRight`. When direction is `rtl`, `paddingEnd` is equivalent to `paddingLeft`.
paddingStart	number, string	0	When direction is `ltr`, `paddingStart` is equivalent to `paddingLeft`. When direction is `rtl`, `paddingStart` is equivalent to `paddingRight`.
position	oneOf `absolute`, `relative`	relative	`position` in React Native is similar to regular CSS, but everything is set to `relative` by default, so `absolute` positioning is always just relative to the parent. If you want to position a child using specific numbers of logical pixels relative to its parent, set the child to have `absolute` position. If you want to position a child relative to something that is not its parent, just don't use styles for that. Use the component tree. See https://github.com/facebook/css-layout for more details on how `position` differs between React Native and CSS.
right	number	auto*	`right` is the number of logical pixels to offset the right edge of this component. It works similarly to `right` in CSS, but in React Native you must use logical pixel units, rather than percents, ems, or any of that. See https://developer.mozilla.org/en-US/docs/Web/CSS/right for more details of how `right` affects layout.
top	number	auto*	`top` is the number of logical pixels to offset the top edge of this component. It works similarly to `top` in CSS, but in React Native you must use logical pixel units, rather than percents, ems, or any of that. See https://developer.mozilla.org/en-US/docs/Web/CSS/top for more details of how `top` affects layout.

Name	Type	Default	Description
width	number	auto*	`width` sets the width of this component. It works similarly to `width` in CSS, but in React Native you must use logical pixel units, rather than percents, ems, or any of that. See http://www.w3schools.com/cssref/pr_dim_width.asp for more details.
zIndex	number	auto*	`zIndex` controls which components display on top of others. Normally, you don't use `zIndex`. Components render according to their order in the document tree, so later components draw over earlier ones. `zIndex` may be useful if you have animations or custom modal interfaces where you don't want this behavior. It works like the CSS `z-index` property - components with a larger `zIndex` will render on top. Think of the z-direction like it's pointing from the phone into your eyeball. See https://developer.mozilla.org/en-US/docs/Web/CSS/z-index for more detail.

- properties with default value `auto` marked with asterisk are do not actually have `auto` as their default value, they just behave like if they would in *css* if they had `auto` as their value. `auto` is not valid value for those properties in react-native

Shadow Prop Types IOS

Name	Type	Description
shadowColor	`customColorPropType`	Sets the drop shadow color
shadowOffset	`customReactPropTypes.shape( {width: ReactPropTypes.number, height: ReactPropTypes.number} )`	Sets the drop shadow offset
shadowOpacity	number	Sets the drop shadow opacity (multiplied by the color's alpha component)
shadowRadius	number	Sets the drop shadow blur radius

Transforms

Name	Type
decomposedMatrix	`customDecomposedMatrixPropType`

Name	Type
transform	`customReactPropTypes.arrayOf(ReactPropTypes.oneOfType([ReactPropTypes.shape({perspective: ReactPropTypes.number}), ReactPropTypes.shape({rotate: ReactPropTypes.string}), ReactPropTypes.shape({rotateX: ReactPropTypes.string}), ReactPropTypes.shape({rotateY: ReactPropTypes.string}), ReactPropTypes.shape({rotateZ: ReactPropTypes.string}), ReactPropTypes.shape({scale: ReactPropTypes.number}), ReactPropTypes.shape({scaleX: ReactPropTypes.number}), ReactPropTypes.shape({scaleY: ReactPropTypes.number}), ReactPropTypes.shape({translateX: ReactPropTypes.number}), ReactPropTypes.shape({translateY: ReactPropTypes.number}), ReactPropTypes.shape({skewX: ReactPropTypes.string}), ReactPropTypes.shape({skewY: ReactPropTypes.string})]))`
transformMatrix	`customTransformMatrixPropType`

Image

Name	Required	Type	Platforms	Description
...Flexbox				
...ShadowPropTypesIOS				
...Transforms				
backfaceVisibility	false	oneOf `visible`, `hidden`		
backgroundColor	false	`ColorPropType`		
borderBottomLeftRadius	false	number		
borderBottomRightRadius	false	number		
borderColor	false	`ColorPropType`		
borderRadius	false	number		
borderTopLeftRadius	false	number		
borderTopRightRadius	false	number		
borderWidth	false	number		
opacity	false	number		
overflow	false	oneOf `visible`, `hidden`		

Name	Required	Type	Platforms	Description
resizeMode	false	oneOf `cover`, `contain`, `stretch`, `repeat`, `center`		Determines how to resize the image when the frame doesn't match the raw image dimensions. Visit the official docs for a guide on each
tintColor	false	`ColorPropType`		Changes the color of all the non-transparent pixels to the tintColor.
overlayColor	false	string	android	When the image has rounded corners, specifying an overlayColor will cause the remaining space in the corners to be filled with a solid color. This is useful in cases which are not supported by the Android implementation of rounded corners: - Certain resize modes, such as 'contain' - Animated GIFs A typical way to use this prop is with images displayed on a solid background and setting the `overlayColor` to the same color as the background. For details of how this works under the hood, see http://frescolib.org/docs/rounded-corners-and-circles.html

ScrollView

Name	Required	Type	Platforms	Description
...Flexbox				
...ShadowPropTypesIOS				
...Transforms				
backfaceVisibility	false	oneOf `visible`, `hidden`		
backgroundColor	false	`ColorPropType`		
borderBottomColor	false	`ColorPropType`		
borderBottomLeftRadius	false	number		
borderBottomRightRadius	false	number		

Name	Required	Type	Platforms	Description
borderBottomWidth	false	number		
borderColor	false	`ColorPropType`		
borderLeftColor	false	`ColorPropType`		
borderLeftWidth	false	number		
borderRadius	false	number		
borderRightColor	false	`ColorPropType`		
borderRightWidth	false	number		
borderStyle	false	oneOf `solid`, `dotted`, `dashed`		
borderTopColor	false	`ColorPropType`		
borderTopLeftRadius	false	number		
borderTopRightRadius	false	number		
borderTopWidth	false	number		
borderWidth	false	number		
opacity	false	number		
overflow	false	oneOf `visible`, `hidden`		
elevation	false	number	android	(Android-only) Sets the elevation of a view, using Android's underlying elevation API. This adds a drop shadow to the item and affects z-order for overlapping views. Only supported on Android 5.0+, has no effect on earlier versions.

Text

Name	Required	Type	Platforms	Description
...View				
color	false	`ColorPropType`		
fontFamily	false	string		
fontSize	false	number		

Name	Required	Type	Platforms	Description
fontStyle	false	oneOf `normal`, `italic`		
fontVariant	false	arrayOf(oneOf`small-caps`, `oldstyle-nums`, `lining-nums`, `tabular-nums`, `proportional-nums`)	ios	
textTransform	false	oneOf `none`, `uppercase`, `lowercase`, `capitalize`		
fontWeight	false	oneOf `normal`, `bold`, `100`, `200`, `300`, `400`, `500`, `600`, `700`, `800`, `900`		Specifies font weight. The values 'normal' and 'bold' are supported for most fonts. Not all fonts have a variant for each of the numeric values, in that case the closest one is chosen.
includeFontPadding	false	bool	android	Set to false to remove extra font padding intended to make space for certain ascenders / descenders. With some fonts, this padding can make text look slightly misaligned when centered vertically. For best results also set `textAlignVertical` to center. Default is true.
lineHeight	false	number		
textAlign	false	oneOf `auto`, `left`, `right`, `center`, `justify`		Specifies text alignment. The value 'justify' is only supported on iOS and fallbacks to `left` on Android.
textDecorationLine	false	oneOf `none`, `underline`, `line-through`		
textShadowColor	false	`ColorPropType`		

Name	Required	Type	Platforms	Description
textShadowOffset	false	`ReactPropTypes.shape( {width: ReactPropTypes.number, height: ReactPropTypes.number} )`		
textShadowRadius	false	number		
textAlignVertical	false	oneOf `auto`, `top`, `bottom`, `center`	android	
letterSpacing	false	number	ios	
textDecorationColor	false	`ColorPropType`	ios	
textDecorationStyle	false	oneOf `solid`, `double`, `dotted`, `dashed`	ios	
writingDirection	false	oneOf `auto`, `ltr`, `rtl`	ios	

View

Name	Required	Type	Platforms	Description
...Flexbox				
...ShadowPropTypesIOS				
...Transforms				
backfaceVisibility	false	oneOf `visible`, `hidden`		
backgroundColor	false	`ColorPropType`		
borderBottomColor	false	`ColorPropType`		
borderBottomEndRadius	false	number		
borderBottomStartRadius	false	number		
borderBottomLeftRadius	false	number		
borderBottomRightRadius	false	number		
borderBottomWidth	false	number		
borderColor	false	`ColorPropType`		
borderEndColor	false	`ColorPropType`		
borderStartColor	false	`ColorPropType`		
borderLeftColor	false	`ColorPropType`		

Name	Required	Type	Platforms	Description
borderLeftWidth	false	number		
borderRadius	false	number		
borderRightColor	false	`ColorPropType`		
borderRightWidth	false	number		
borderStyle	false	oneOf `solid`, `dotted`, `dashed`		
borderTopColor	false	`ColorPropType`		
borderTopEndRadius	false	number		
borderTopStartRadius	false	number		
borderTopLeftRadius	false	number		
borderTopRightRadius	false	number		
borderTopWidth	false	number		
borderWidth	false	number		
opacity	false	number		
overflow	false	oneOf `visible`, `hidden`		
elevation	false	number	android	(Android-only) Sets the elevation of a view, using Android's underlying elevation API. This adds a drop shadow to the item and affects z-order for overlapping views. Only supported on Android 5.0+, has no effect on earlier versions.

TextInput

Name	Required	Type	Platforms	Description
autoFocus	false	bool		If true, focuses the input on componentDidMount. The default value is false.

Name	Required	Type	Platforms	Description
keyboardType	false	oneOf `default`, `email-address`, `numeric`, `phone-pad`, // iOS-only `ascii-capable`, `numbers-and-punctuation`, `url`, `number-pad`, `name-phone-pad`, `decimal-pad`, `twitter`, `web-search`		Determines which keyboard to open
maxLength	false	number		Limits the maximum number of characters that can be entered
onChangeText	false	callback func		Callback that is called when the text input's text changes. Changed text is passed as an argument to the callback handler.

Appendix

Types

number

ReactPropTypes.number

string

ReactPropTypes.string

bool

ReactPropTypes.bool

oneOf

ReactPropTypes.oneOf([values])

arrayOf

ReactPropTypes.arrayOf(value)