

Dimensions

`useWindowDimensions` is the preferred API for React components. Unlike `Dimensions`, it updates as the window's dimensions update. This works nicely with the React paradigm.

```
import {Dimensions} from 'react-native';
```

You can get the application window's width and height using the following code:

```
const windowWidth = Dimensions.get('window').width;  
const windowHeight = Dimensions.get('window').height;
```

Although dimensions are available immediately, they may change (e.g due to device rotation, foldable devices etc) so any rendering logic or styles that depend on these constants should try to call this function on every render, rather than caching the value (for example, using inline styles rather than setting a value in a `StyleSheet`).

If you are targeting foldable devices or devices which can change the screen size or app window size, you can use the event listener available in the `Dimensions` module as shown in the below example.

Example

Dimensions



```
import React, {useState, useEffect} from 'react';
import {View, StyleSheet, Text, Dimensions} from 'react-native';

const windowDimensions = Dimensions.get('window');
const screenDimensions = Dimensions.get('screen');

const App = () => {
  const [dimensions, setDimensions] = useState({
    window: windowDimensions,
    screen: screenDimensions,
  });

  useEffect(() => {
    const subscription = Dimensions.addEventListener(
      'change',
      ({window, screen}) => {
        setDimensions({window, screen});
      },
    );
  });
  return () => subscription?.remove();
};

return (
  <View style={styles.container}>
    <Text style={styles.header}>Window Dimensions</Text>
    {Object.entries(dimensions.window).map(([key, value]) => (
```

Preview



My Device

iOS

Android

Web

Reference

Methods

addEventListener()

```
static addEventListener(
  type: 'change',
  handler: ({
    window,
    screen,
  }: DimensionsValue) => void,
): EmitterSubscription;
```

Add an event handler. Supported events:

- `change`: Fires when a property within the `Dimensions` object changes. The argument to the event handler is a `DimensionsValue` type object.

`get()`

```
static get(dim: 'window' | 'screen'): ScaledSize;
```

Initial dimensions are set before `runApplication` is called so they should be available before any other require's are run, but may be updated later.

Example: `const {height, width} = Dimensions.get('window');`

Parameters:

NAME	TYPE	DESCRIPTION
<code>dim</code> <div>Required</div>	string	Name of dimension as defined when calling <code>set</code> . Returns value for the dimension.

For Android the window dimension will exclude the size used by the `status bar` (if not translucent) and `bottom navigation bar`

Type Definitions

DimensionsValue

Properties:

NAME	TYPE	DESCRIPTION
<code>window</code>	<code>ScaledSize</code>	Size of the visible Application window.

NAME	TYPE	DESCRIPTION
screen	<u>ScaledSize</u>	Size of the device's screen.


ScaledSize

TYPE
object

Properties:

NAME	TYPE
width	number
height	number
scale	number
fontScale	number

Is this page useful?  

 Edit this page

Last updated on **Jun 21, 2023**