PixelRatio

`PixelRatio` gives you access to the device's pixel density and font scale.

Fetching a correctly sized image

You should get a higher resolution image if you are on a high pixel density device. A good rule of thumb is to multiply the size of the image you display by the pixel ratio.

```
const image = getImage({
  width: PixelRatio.getPixelSizeForLayoutSize(200),
  height: PixelRatio.getPixelSizeForLayoutSize(100),
});
<Image source={image} style={{width: 200, height: 100}} />;
```

Pixel grid snapping

In iOS, you can specify positions and dimensions for elements with arbitrary precision, for example 29.674825. But, ultimately the physical display only have a fixed number of pixels, for example 640×1136 for iPhone SE (1st generation) or 828×1792 for iPhone 11. iOS tries to be as faithful as possible to the user value by spreading one original pixel into multiple ones to trick the eye. The downside of this technique is that it makes the resulting element look blurry.

In practice, we found out that developers do not want this feature and they have to work around it by doing manual rounding in order to avoid having blurry elements. In React Native, we are rounding all the pixels automatically.

We have to be careful when to do this rounding. You never want to work with rounded and unrounded values at the same time as you're going to accumulate rounding errors. Having even one rounding error is deadly because a one pixel border may vanish or be twice as big.

In React Native, everything in JavaScript and within the layout engine works with arbitrary precision numbers. It's only when we set the position and dimensions of the native element on the main thread that we round. Also, rounding is done relative to the root rather than the parent, again to avoid accumulating rounding errors.

Example

PixelRatio Example

Expo

```
import React from 'react';
import {
  Image,
  PixelRatio,
  ScrollView,
  StyleSheet,
  Text,
  View,
} from 'react-native';

const size = 50;
const cat = {
  uri: 'https://reactnative.dev/docs/assets/p_cat1.png',
  width: size,
  height: size,
};

const App = () => (
  <ScrollView style={styles.scrollContainer}>
    <View style={styles.container}>
      <Text>Current Pixel Ratio is:</Text>
      <Text style={styles.value}>{PixelRatio.get()}</Text>
    </View>
    <View style={styles.container}>
      <Text>Current Font Scale is:</Text>
      <Text style={styles.value}>{PixelRatio.getFontScale()}</Text>
```

Preview My Device iOS Android Web

Reference

Methods

`get()`

```
static get(): number;
```

Returns the device pixel density. Some examples:

- `PixelRatio.get() === 1`
  - mdpi Android devices
- `PixelRatio.get() === 1.5`
  - hdpi Android devices
- `PixelRatio.get() === 2`
  - iPhone SE, 6S, 7, 8
  - iPhone XR
  - iPhone 11
  - xhdpi Android devices
- `PixelRatio.get() === 3`
  - iPhone 6S Plus, 7 Plus, 8 Plus
  - iPhone X, XS, XS Max
  - iPhone 11 Pro, 11 Pro Max
  - Pixel, Pixel 2
  - xxhdpi Android devices
- `PixelRatio.get() === 3.5`
  - Nexus 6
  - Pixel XL, Pixel 2 XL
  - xxxhdpi Android devices

`getFontScale()`

```
static getFontScale(): number;
```

Returns the scaling factor for font sizes. This is the ratio that is used to calculate the absolute font size, so any elements that heavily depend on that should use this to do calculations.

- on Android value reflects the user preference set in **Settings > Display > Font size**
- on iOS value reflects the user preference set in **Settings > Display & Brightness > Text Size**, value can also be updated in **Settings > Accessibility > Display & Text Size > Larger Text**

If a font scale is not set, this returns the device pixel ratio.

`getPixelSizeForLayoutSize()`

```
static getPixelSizeForLayoutSize(layoutSize: number): number;
```

Converts a layout size (dp) to pixel size (px).

Guaranteed to return an integer number.

`roundToNearestPixel()`

```
static roundToNearestPixel(layoutSize: number): number;
```

Rounds a layout size (dp) to the nearest layout size that corresponds to an integer number of pixels. For example, on a device with a PixelRatio of 3, `PixelRatio.roundToNearestPixel(8.4) = 8.33`, which corresponds to exactly (8.33 * 3) = 25 pixels.