1

I. The relationship between UI and UX

UI (User Interface)

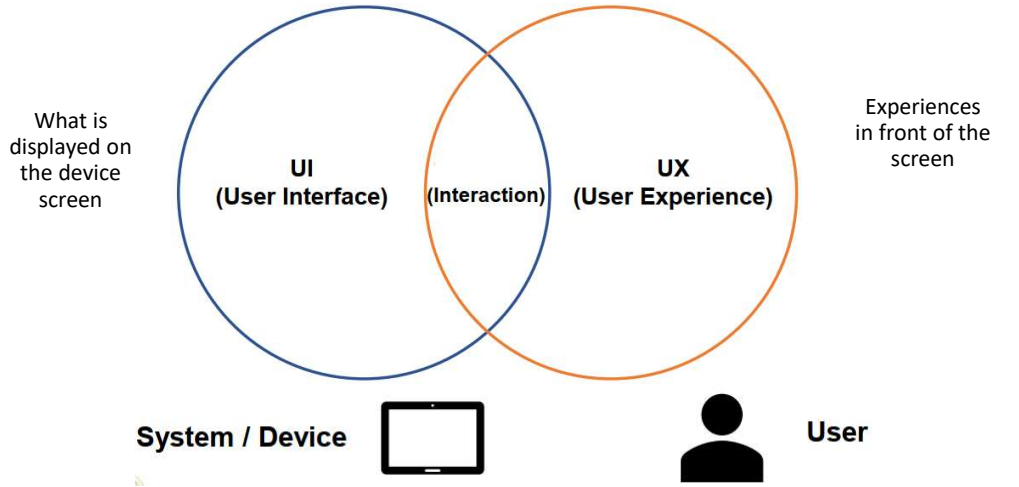
Include everything that the user can see on the device screen, such as layout, colors, fonts, images, etc.

UX (User Experience)

Includes many other factors besides the interface such as experience, emotions, value received when interacting with products and services.

2

I. The relationship between UI and UX

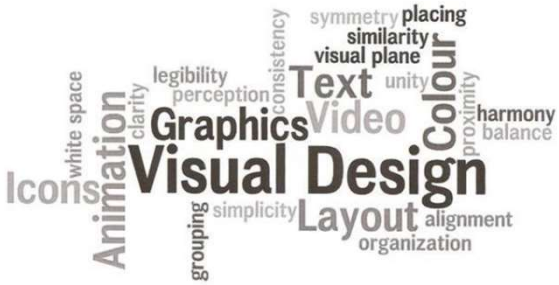


3

I. The relationship between UI and UX

UI design refers to how content is presented visually:

- Layout
- Color and contrast
- Images and icons
- Font
- Vocabulary and terms

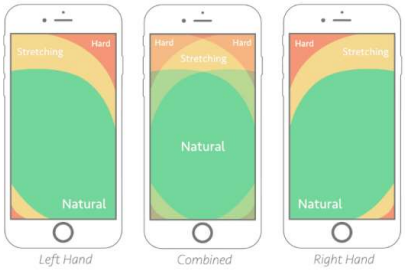


4

I. The relationship between UI and UX

Interaction design elements in mobile apps:

- Gestures
- Data Entry
- Navigation

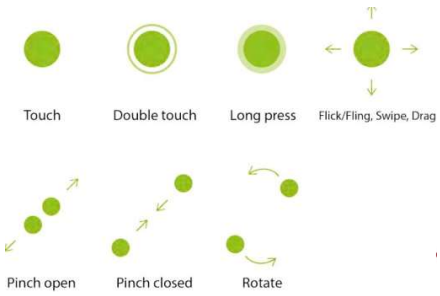


5

I. The relationship between UI and UX

Gestures

- The touch screen covers almost the entire front of the mobile device, leaving no space for physical buttons.
- Touchscreen devices rely heavily on gesture control, which uses human hands to interact with content on the screen.
- Standard touchscreen gestures:



6

I. The relationship between UI and UX

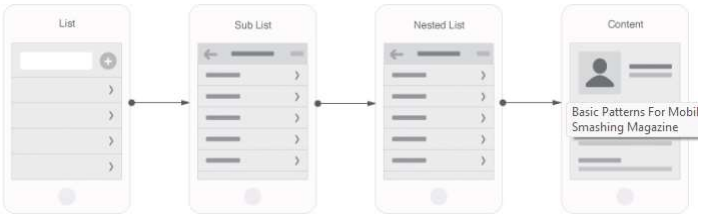
Data Entry

- Data entry is an essential form of interaction.
- The validity of the entered data can be ensured by reducing incorrect inputs.
- On devices with a touch screen, enter data via the on-screen keyboard, the size of which depends on the screen.
- Input methods that require a virtual keyboard should be kept to a minimum, alternative input methods can be used (e.g. date picker, drop-down fields, radio button groups, defaults...

I. The relationship between UI and UX

Navigation

- Displays a lot of information in screens that are usually organized in a hierarchical structure throughout the application
- Users have to navigate through different screens
- Important information and key functionality should be placed higher in the navigation hierarchy, allowing users to reach it faster.
- Ability to return to the previous screen (swipe gesture / back button in the app)



I. The relationship between UI and UX

Challenges of mobile interfaces:

- Small screen size
- Interact via touch screen
- always on & always connected
- Personalize the user experience.
- Diverse hardware with a variety of sensors

9

II. Mobile interface design

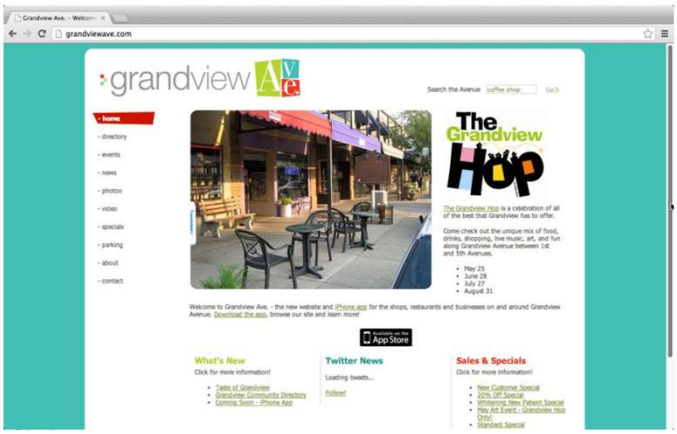
Mobile-first design

- Users use mobile phones everywhere.
- Designers have a shift in priorities when building layouts first for mobile screens.
- Optimizing all content, features, and especially enhancing the experience are key concerns.
  - Responsiveness.
  - Keep it simple·
  - Finger-Friendly design.
  - Feedback

10

II. Mobile interface design

Responsive layouts



11

II. Mobile interface design

Keep it simple

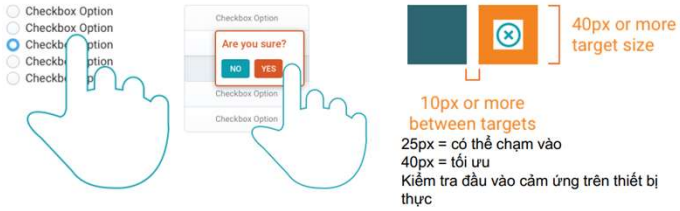
- Simplicity is the key to a better user experience
- Remove UI clutter

12

II. Mobile interface design

Finger-Friendly design

- Most gestures are designed to be used with one hand
- Interacting with the touch screen through fingers: smaller touch targets => unwieldy and more likely to occur errors => designed to be optimal for touch

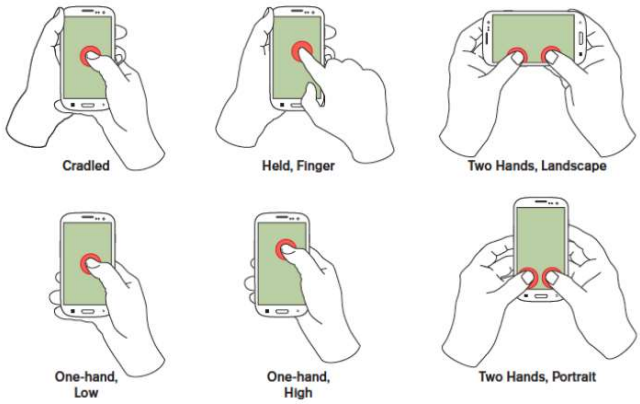


13

II. Thiết kế giao diện cho di động

Finger-Friendly design

- Hold the device in different ways



14

II. Mobile interface design

Feedback

- App feedback is important during user interaction.
- Give feedback to users not only when loading content, but after each completion of a specific task.
- Incorporate different types of feedback such as flashing lights, vibrations, or sounds.

15

II. Mobile interface design

- Analyze user behavior
- Context analysis

16

III. Layout in ReactNative

- All basic components of RN have a **prop** called **style**
- **Prop style** can be a pure JavaScript object
- Use **StyleSheet.create** to define some focus types that are usually easier to manage
- Styles can be "cascaded" (cascade) just like in CSS

17

III. Layout in ReactNative

- Apply styles in react native applications

```
<View style={{marginLeft: 20,
marginTop: 20}}>
<Text style={{fontSize:
18,color: 'red'}}>Some
Text</Text>
</View>
```

Use inline styles

```
<View style={styles.container}>
<Text
style={[styles.message,styles.warning]}>S
ome Text</Text>
</View>

const styles = StyleSheet.create({
container: {
marginLeft: 20,
marginTop: 20
},
message: {
fontSize: 18
},
warning: {
color: 'red'
}
});
```

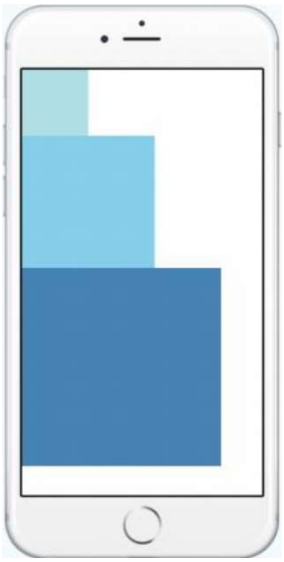
using StyleSheet

18

III. Layout in ReactNative

- The component size is set via the style values: **width** and **height**

```
import React, { Component } from 'react';
import { AppRegistry, View } from 'react-native';
export default class FixedDimensionsBasics extends Component
{
render() {
   return (
      <View>
         <View style={{width: 50, height: 50,
          backgroundColor: 'powderblue'}} />
         <View style={{width: 100, height: 100,
          backgroundColor: 'skyblue'}} />
         <View style={{width: 150, height: 150,
          backgroundColor: 'steelblue'}} />
```



III. Layout in ReactNative

- Flexbox is designed to provide consistent layout across different screen sizes
- Flexboxes are properties used to build component layouts
- Combine **flexDirection**, **alignItems**, and **justifyContent** to achieve the right layout

Property	Values	Description
flexDirection	'column', 'row'	Used to specify whether elements will be aligned vertically or horizontally.
justifyContent	'center', 'flex-start', 'flex-end', 'space-around', 'space-between'	Used to determine how content displays inside
alignItems	'center', 'flex-start', 'flex-end', 'stretched'	Used to determine how elements display inside

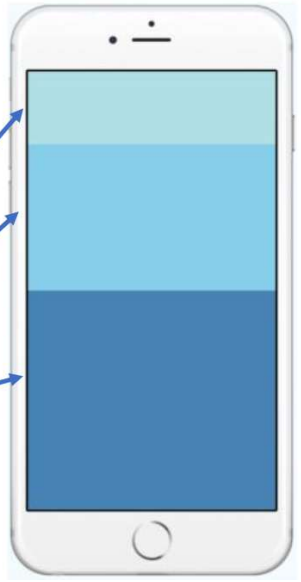
III. Layout in ReactNative

- **Examples of Flex**

```
import React, { Component } from 'react';
import { AppRegistry, View } from 'react-native';
export default class FlexDimensionsBasics extends
Component {
render() {
return (
   <View style={{flex: 1}}>
   <View style={{flex: 1, backgroundColor:
                 'powderblue'}} />
   <View style={{flex: 2, backgroundColor:
                 'skyblue'}} />
   <View style={{flex: 3, backgroundColor:
                 'steelblue'}} />
   </View>
);  }. }
```

Root View - flex: 1

Total flex :6



21

III. Layout in ReactNative

- **Examples of flexDirection**

```
import React, { Component } from 'react';
import { AppRegistry, View } from 'react-native';
export default class FlexDirectionBasics extends Component {
render() {
return (
// Try setting `flexDirection` to `column`.
<View style={{flex: 1, flexDirection: 'row'}}>
<View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />
<View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />
<View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />
</View>
); } };
```

22

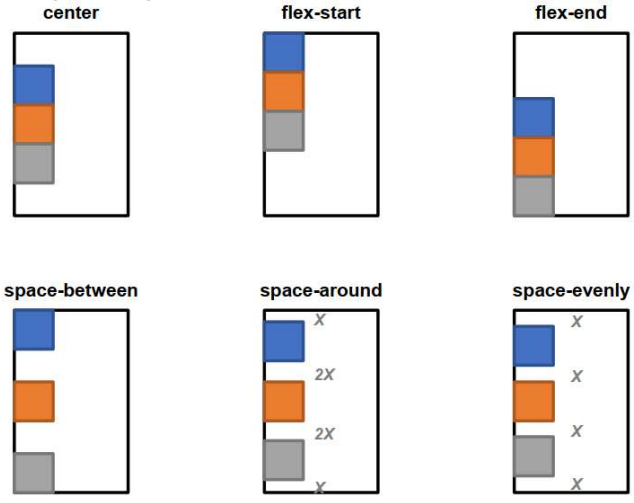
III. Layout in ReactNative

- **Examples of justifyContent**


```
import React, { Component }
from 'react';
import { AppRegistry, View }
from 'react-native';
export default class JustifyContentBasics
extends Component {
render() {
return (
// Try setting `justifyContent` to `center`.
// Try setting `flexDirection` to `row`.
<View style={{
flex: 1,
flexDirection: 'column',
justifyContent: 'space-between',
}}>
<View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />
<View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />
<View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />
</View>
);  }. };
```

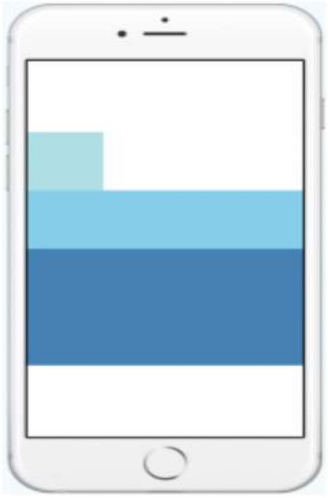

III. Layout in ReactNative

- **Examples of justifyContent**



III. Layout in ReactNative

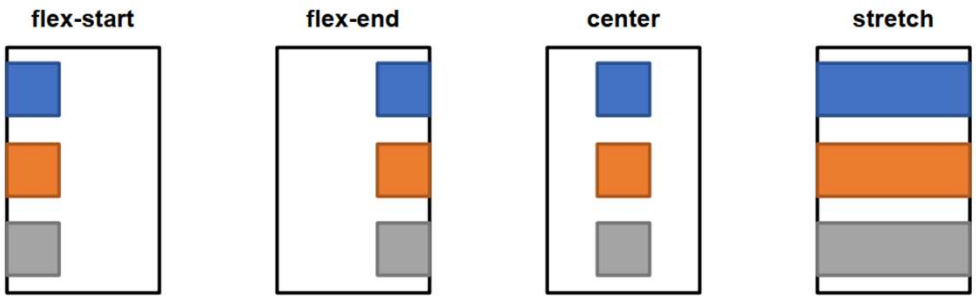
- **Examples of alignitems**



```
import React, { Component } from 'react';
import { AppRegistry, View } from 'react-native';
export default class AlignItemsBasics
extends Component {
render() {
return (
// Try setting `alignItems` to 'flex-start'
// Try setting `justifyContent` to `flex-end`.
// Try setting `flexDirection` to `row`.
<View style={{
flex: 1,
flexDirection: 'column',
justifyContent: 'center',
alignItems: 'stretch',
}}>
<View style={{width: 50, height: 50, backgroundColor: 'powderblue'}}
/>
<View style={{height: 50, backgroundColor: 'skyblue'}} />
<View style={{height: 100, backgroundColor: 'steelblue'}} />
</View>
);  }. };
```


III. Layout trong ReactNative

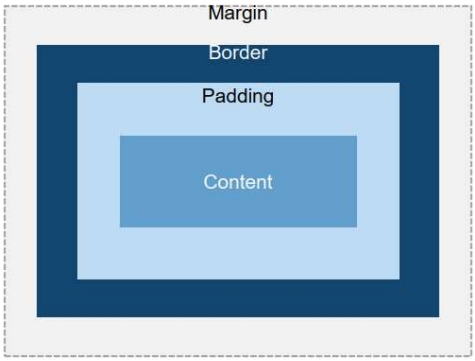
- **Examples of attributes alignitems**



III. Layout trong ReactNative

- **Margins, Borders & Padding**

The components follow the Box model, similar to the CSS box model:



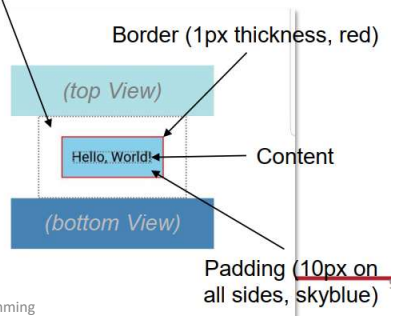

```
const styles = StyleSheet.create({
content: {
padding: 20,
margin: 0,
backgroundColor: '#ef4c',
width: 125,
height: 125,
borderWidth: 1,
borderColor: 'red',
textAlign: 'center'
}
});
```


III. Layout trong ReactNative

- **Margins, Borders & Padding**

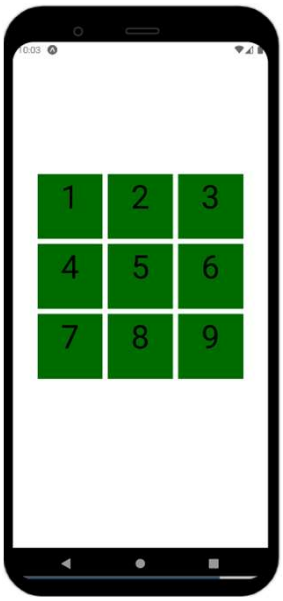
Margin (20px on all sides, transparent)

```
<View style={{…}} />
<Text style={{
   backgroundColor: 'skyblue',
   padding: 10,
   borderWidth: 1,
   borderColor: 'red',
   margin: 20
}}>Hello, World!</Text>
<View style={{…}} />
```

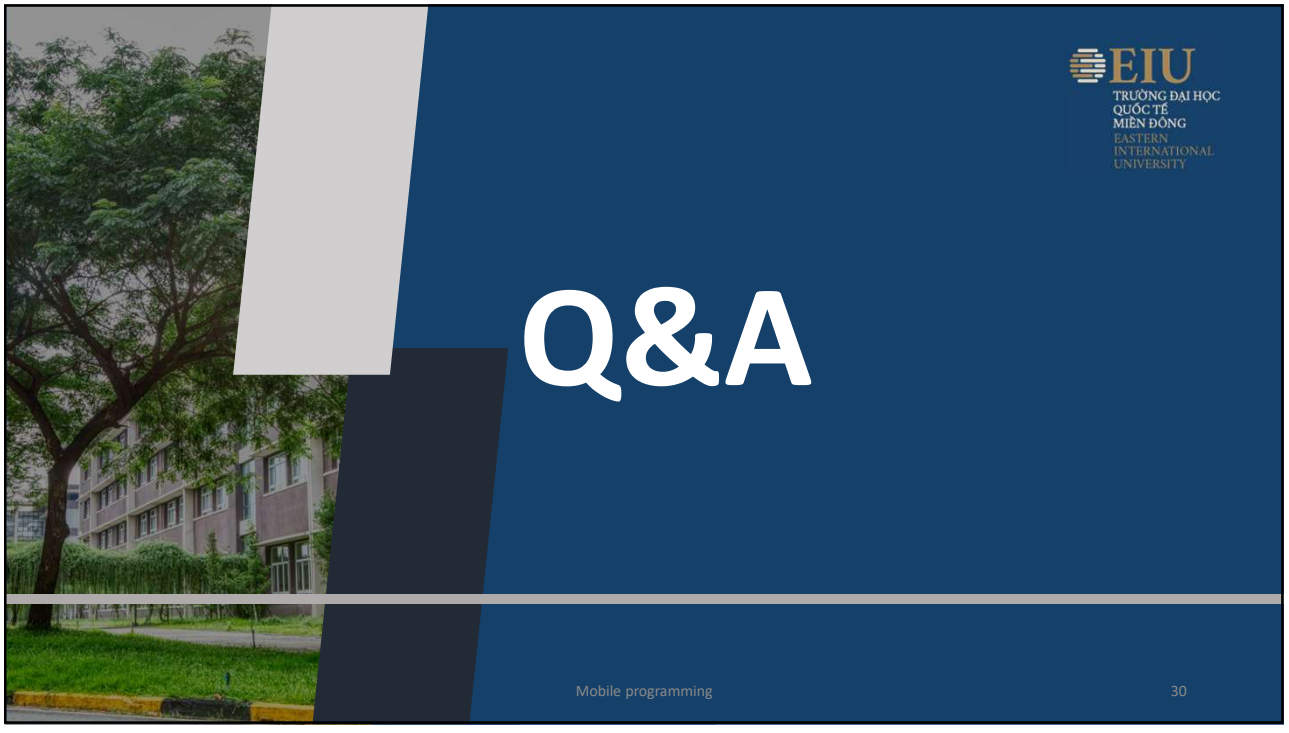


Exercises

Use https://snack.expo.dev/ **to design the interface as shown**



29



30