链滴

Flutter 的一些知识点

作者：devcui

原文链接：https://ld246.com/article/1630246837383

来源网站：链滴

<h2 id="flutter">flutter</h2>
<h2 id="1-intro">1.intro</h2>
flutter 本质是 widget 树

<code>Text</code>: 格式文本
<code>Row</code>: 水平布局
<code>Column</code>: 垂直布局
<code>Stack</code>: 线性布局，<code>Positioned</code> 定位
<code>Container</code>: 矩阵，<code>BoxDecoration</code> 装饰 <code>Container
/code>
<code>Expanded</code>: 填充剩余空间
<code>Navigator</code>: 管理 <code>Widget</code> 栈
<code>Scaffold</code>: Material 的一种布局结构
<code>GestureDetector</code>: 识别用户手势,<code>onTap</code> 回调
<code>StatefulWidget</code>: 有状态控件,实现 <code>createState</code> 返回一个继
自 <code>State<WidgetType</code> 的状态型控件
<code>State</code>: 可以通过给 <code>action</code> 来分层，Widget 都存在 State
，通过 AcationWidget 修改 State 里的值，达到渲染 DisplayWdiget 的目的，一个负责渲染/发送 A
tion，一个负责渲染数据

<h2 id="2-layout">2.layout</h2>

数据结构为树
所有 layout 本质都是 <code>widget</code>
所有布局 <code>widget</code> 都含有 <code>child</code> 和 <code>children</code
属性

<code>child</code>: 包含一个子 <code>widget</code>
<code>children</code>: 含有多个子项,布局子项，直至最后布局子项包含一个可见 <code>W
dget</code>

纵向用 Column 横向用 Row 可嵌套

<code>AxisAligment</code>: 表示对齐
<code>Expanded</code>:可以自适应布局高宽

<code>flex</code>: 弹性系数

<code>Container</code>: 可扩展 <code>padding,margins,borders,background color</
ode> 等其他装饰元素 <code>decoration</code>
<code>GridView</code>: 可以做滚动网格
<code>ListView</code>: 滚动列表
<code>Stack</code>: 栈，先进后出，也就是后面的 <code>Widget</code> 盖到前一个 <
ode>Widget</code> 上，形成一个 <code>Widget</code> 栈
<code>Card</code> 是 <code> Material</code> 中的一个组件
<code>EdgeInsets</code>: 间距

自适应: 根据设备发生变化。
响应式: 根据屏幕大小发生变化。
<h2 id="自适应应用">自适应应用</h2>

Single Child

<code>Align</code>: 子级内部对齐
<code>AspectRatio</code>: 为子级指定比例
<code>ConstrainedBox</code>: 对子级施加最大最小尺寸限制
<code>CustomSingleChildLayout</code>: 代理方法对子级进行定位
<code>Expanded,Flexble</code>: <code>Row,Column</code> 填充剩余空间
<code>FractionallySizedBox</code>: 基于剩余控件比例限定子级大小
<code>LayoutBuilder</code>: 让子级可以基于父级的尺寸重新调整其布局
<code>SingleChildScrollView</code>: 为单一子级添加滚动

Multi Child

<code>Column,Row,Flex</code>
<code>CustomMultiChildLayout</code>: 代理多个子级定位
<code>Flow</code>: 类似 :arrow_up:
<code>ListView,GridView,CustomScrollView</code>: 为所有子级添加滚动
<code>Stack</code>: 基于 <code>Stack</code> 边界对多个子级定位
<code>Table</code>: 表格布局
<code>Wrap</code>: 将子级顺序显示在多行/多列内

<h2 id="视觉密度">视觉密度</h2>
widget 的疏密程度，紧凑程度等意思，其实就是一个动态的值来自动调整宽高/间距什么的

<code>VisualDensity</code>
double densityAmt = enableTouchMode ? 0.0 : -1.0;
// 水平竖直
VisualDensity density = VisualDensity(horizontal: density, vertical: density);
return MaterialApp(
// 应用到 MaterialApp 主题
theme: ThemeData(visualDensity: density),
...
);
// 使用主题的疏密度
VisualDensity density = Theme.of(context).visualDensity;
return Padding(
padding: EdgeInsets.all(Insets.large + density.vertical * 4),
child: ...
);

<h2 id="基于Context的布局">基于 Context 的布局</h2>

<code>MediaQuery</code>: 媒体查询，给了一些分界点
ScreenType getFormFactor(BuildContext context) {
// Use .shortestSide to detect device type regardless of orientation
double deviceWidth = MediaQuery.of(context).size.shortestSide;
if (deviceWidth > FormFactor.desktop) return ScreenType.Desktop;

if (deviceWidth > FormFactor.tablet) return ScreenType.Tablet;
if (deviceWidth > FormFactor.handset) return ScreenType.Handset;
return ScreenType.Watch;
}

以下是反转屏幕的一个判断，可以基于媒体查询判断是否反转，从顶传入子，使得整个 UI 做出幕翻转的响应
<pre><code class="highlight-chroma">bool isHandset = MediaQuery.of(context).size.width < 600;
return Flex(
  children: [...],
  direction: isHand et ?
    Axis.vertical :
    Axis.horizontal
);
</code></pre>
<h2 id="layout-builder">layout builder</h2>
layout builder 和之前不同的是，提供了一些对象，让你更好的做出判断和使用
<pre><code class="highlight-chroma">Widget foo = LayoutBuilder(builder: (_, constraints, __){
 bool useVerticalL yout = constraints.maxWidth < 400.0;
 return Flex(
    children: [...],
    direction: useV rticalLayout ?
    Axis.vertical : A is.horizontal
 );
});
</code></pre>
<h2 id="设备细分">设备细分</h2>
<pre><code class="highlight-chroma">// Platform对象判断设备类型
bool get isMobile evice => !kIsWeb && (Platform.isIOS || Platform.isAndroid);
// macos windows
bool get isDeskto Device =>
  !kIsWeb & & (Platform.isMacOS || Platform.isWindows || Platform.isLinux);
// browser
bool get isMobile eviceOrWeb => kIsWeb || isMobileDevice;
bool get isDeskto DeviceOrWeb => kIsWeb || isDesktopDevice;
</code></pre>
<h2 id="input">input</h2>

<code>scrollView,ListView</code>: 自动支持 <code>onPointerSingal.PointerScrollEvent /code> 滑轮输入

实现自定义滑动: <code>Listener Widget</code> 的 <code>onPointerSingal.PointerScroll
vent</code>

<code>FocusableActionDetector</code>: 在元素外包裹，处理焦点，悬浮，鼠标跟随

<code>RawKeyboardListener</code>: 监听键盘事件

<code>Shortcuts</code>: 直接绑定快捷键

<code>RawKeyboard.instance.addListener(_handleKey);</code> 全局监听键盘事件
// 单一快捷键是否按下了
static bool isKeyDown(Set keys) {
return
keys.intersection(RawKeyboard.instance.keysPressed).isNotEmpty;
}
void _handleKey(event){
if (event is RawKeyDownEvent) {
// 两个 shift 组合键
bool isShiftDown = isKeyDown({
LogicalKeyboardKey.shiftLeft,
LogicalKeyboardKey.shiftRight,
});
if (isShiftDown && event.logicalKey == LogicalKeyboardKey.keyN) {
_createNewItem();
}
}
}

鼠标
<pre><code class="highlight-chroma"><span class="highlight
cl">MouseRegion
return MouseRegi
n(
// 进入
  onEnter: (_) =&gt
 setState(() => _isMouseOver = true),
// 离开
onExit: (_) => s
tState(() => _isMouseOver = false),
// 悬停
onHover: (Pointer
overEvent e) => print(e.localPosition),
  child: ...,
);
</code></pre>
<h2 id="用户期望">用户期望</h2>

<code>ScrollBar</code>: 根据平台切换状态
// 不同设备的输入是不同的
static bool get isMultiSelectModifierDown {
bool isDown = false;
// 使用 DeviceOS 来判断设备
if (DeviceOS.isMacOS) {
isDown = isKeyDown([LogicalKeyboardKey.metaLeft, LogicalKeyboardKey.metaRight]);
} else {
isDown = isKeyDown([LogicalKeyboardKey.controlLeft, LogicalKeyboardKey.controlRight]);<b
>
}
return isDown;
}

<code>SelectableText</code>: 文字是否可以被选中

<code>SelectableText.rich(TexSpan)</code>: 富文本选中

<code>Tooltip</code>: 悬停

<h2 id="布局约束">布局约束</h2>
首先，上层 widget 向下层 widget 传递约束条件。
然后，下层 widget 向上层 widget 传递大小信息。
最后，上层 widget 决定下层 widget 的位置。
长宽计算更像是 子向父报备，父汇集各个子然后计算，在向父父汇报

<code>ConstrainedBox</code>: 从父级接收约束施加给子级
<code>Center(child: ConstrainedBox())</code>: 允许控制
<code>UnconstrainedBox</code>: 允许子级改变为任意大小
<code>OverflowBox</code>: 可溢出，需要在本级做出限制

限制 外到内，外告知内可小于外宽松约束，外告知内必须变成某个大小严格约束
<h2 id="添加互动">添加互动</h2>

<code>StatefulWidget</code>: 和 React 一样
其实就是状态管理，类似 React 的 State，这个 State 可以是自己的 State，也可以是自己的 Pro
s，从父级 State 传入过来，只不过这里依靠构造函数显示声明
当然可以从父级传回调函数在子级改变父级状态
<code>GestureDetector</code>: 添加其他互动，什么是互动啊，就是用户的行为能改变数据
切换图标也算是数据变动)的地方就是有互动，理解了吧

<h2 id="指定资源">指定资源</h2>
<pre><code class="highlight-chroma"><span class="highlight
cl">flutter:
  assets:
    - assets/my_ico
.png
    - assets/backgr
und.png

```
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">flutter:
</span></span><span class="highlight-line"><span class="highlight-cl">  assets:
</span></span><span class="highlight-line"><span class="highlight-cl">    - directory/
</span></span><span class="highlight-line"><span class="highlight-cl">    - directory/subd
rectory/
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">// 直接读文件了
</span></span><span class="highlight-line"><span class="highlight-cl">Future&lt;String&
t; loadAsset() async {
</span></span><span class="highlight-line"><span class="highlight-cl">  return await roo
Bundle.loadString('assets/config.json');
</span></span><span class="highlight-line"><span class="highlight-cl">}
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">// 不同分辨率的图
，可FLutter可以对应不同的设备自动切换
</span></span><span class="highlight-line"><span class="highlight-cl">.../my_icon.png
</span></span><span class="highlight-line"><span class="highlight-cl">.../2.0x/my_icon.p
g
</span></span><span class="highlight-line"><span class="highlight-cl">.../3.0x/my_icon.p
g
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">// 使用别的包的图
，也要声明出来
</span></span><span class="highlight-line"><span class="highlight-cl">flutter:
</span></span><span class="highlight-line"><span class="highlight-cl">  assets:
</span></span><span class="highlight-line"><span class="highlight-cl">    - packages/fan
y_backgrounds/backgrounds/background1.png
</span></span></code></pre>
<p>注意: 读取 android/ios 图片插件和方式不同，文件夹内有 <code>android</code>,<code>io
</code> 两个文件夹，需要添加各自的插件，<code>图标</code>,<code>预加载图</code> 也
如此</p>
<h2 id="路由">路由</h2>
<p>安卓启动深度路由 <code>AndroidManifest.xml</code></p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">&lt;!-- Deep linking --&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">&lt;meta-data and
oid:name="flutter_deeplinking_enabled" android:value="true" /&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">&lt;intent-filter a
droid:autoVerify="true"&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    &lt;action andro
d:name="android.intent.action.VIEW" /&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    &lt;category an
roid:name="android.intent.category.DEFAULT" /&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    &lt;category an
roid:name="android.intent.category.BROWSABLE" /&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    &lt;data androi
:scheme="http" android:host="flutterbooksample.com" /&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    &lt;data androi
:scheme="https" /&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">&lt;/intent-filter&
t;
</span></span></code></pre>
```

配合 adb 测试
<pre><code class="highlight-chroma"><span class="highlight
cl">adb shell am start -a android.intent.action.VIEW \
    -c android.inten
.category.BROWSABLE \
    -d "http://flutte
booksample.com/book/1"
</code></pre>
IOS <code>Info.plist</code>
<pre><code class="highlight-chroma"><span class="highlight
cl"><key>FlutterDeepLinkingEnabled</key>
<true/>
<key>CFBun
leURLTypes</key>
<array>
    <dict>
    <key>CFB
ndleTypeRole</key>
    <string>Ed
tor</string>
    <key>CFB
ndleURLName</key>
    <string>fl
tterbooksample.com</string>
    <key>CFB
ndleURLSchemes</key>
    <array>
    <string>c
stomscheme</string>
    </array>
    </dict>
</array>
</code></pre>
配合 <code>xsrun</code> xshell
<pre><code class="highlight-chroma"><span class="highlight
cl">xcrun simctl openurl booted customscheme://flutterbooksample.com/book/1
</code></pre>
<h2 id="动画">动画</h2>
pass
<h2 id="http">http</h2>
<pre><code class="highlight-chroma"><span class="highlight
cl">import 'package:http/http.dart' as http;

var url = Uri.parse(
https://example.com/whatsit/create');
var response = a
ait http.post(url, body: {'name': 'doodle', 'color': 'blue'});
print('Response st
tus: ${response.statusCode}');
print('Response b
dy: ${response.body}');

print(await http.re
d('https://example.com/foobar.txt'));

</code></pre>
安卓需要在 <code>AndroidManifest.xml</code> 中添加需要访问的网络权限
<pre><code class="language-zed highlight-chroma"><span cl
ss="highlight-cl"><<span class="highlight-n">manifes
 xmlns<spa
class="highlight-o">:android<span class="highl
ght-p">...>
<span cla
s="highlight-w"> ...
<span cla
s="highlight-w"> <<span class="highlight-n"
uses-permission</
pan> android<span
class="highlight-o">:name<span class="highlig
t-o">="android</s
an>.permission<sp
n class="highlight-p">.INTERNET<span class="h
ghlight-err">" /&gt

<span cla
s="highlight-w"> <<span class="highlight-n"
application ...</spa
>
<span cla
s="highlight-w">
</<span class="highlight-n"
manifest>
</code></pre>
<h2 id="json">json</h2>
<pre><code class="highlight-chroma"><span class="highlight
cl">class User {
  final String name

  final String email;

  User(this.name, t
is.email);
// 解析json
  User.fromJson(M
p<String, dynamic> json)
      : name = json[
name'],
        email = json[
email'];
// 转成json
  Map<String, d
namic> toJson() => {
        'name': name

        'email': email,
      };
}
</code></pre>
或者
<pre><code class="highlight-chroma"><span class="highlight

cl">dependencies:
  # Your other reg
lar dependencies here
  json_annotation:
<latest_version>

dev_dependencies

  # Your other dev
dependencies here
  build_runner: <
atest_version>
  json_serializable:
<latest_version>
</code></pre>
引入依赖，使用注解
<pre><code class="language-mysql highlight-chroma"><span
class="highlight-cl">@Json
erializable()
<span cla
s="highlight-w">
class<span class="highlight-w
> User <sp
n class="highlight-err">{
<span cla
s="highlight-w"> User<span class="highlight-p
>(this.<spa
 class="highlight-n">name,<span class="highli
ht-w"> this.</span
email);<span class="
ighlight-w">
<span cla
s="highlight-w">
<span cla
s="highlight-w"> String<span class="highlight
w"> name;
span class="highlight-w">
<span cla
s="highlight-w"> String<span class="highlight
w"> email;<
pan class="highlight-w">
<span cla
s="highlight-w">
<span cla
s="highlight-w"> ///<span class="highlight-w"
 A <span cl
ss="highlight-n">necessary <span class="highli
ht-n">factory const
uctor for<s
an class="highlight-w"> creating<span class="h
ghlight-w"> a </sp
n>new <span class=
highlight-k">User in
tance
<span cla

s="highlight-w"> ///<span class="highlight-w"
 from a <span class="highlight-n
>map. <sp
n class="highlight-n">Pass <span class="highli
ht-n">the map</sp
n> to<span class="h
ghlight-w"> the </s
an>generated <spa
 class="highlight-o">`_<span class="highlight-er
">$UserFromJson()
/span>` <span class
"highlight-n">constructor.<span class="highligh
-w">
<span cla
s="highlight-w"> ///<span class="highlight-w"
 The constructor <span class="hi
hlight-k">is named
/span> after<span c
ass="highlight-w"> the<span class="highlight-
"> source <
pan class="highlight-n">class,<span class="high
ight-w"> in </span
this <span class="hi
hlight-k">case, </s
an>User.<span clas
="highlight-w">
<span cla
s="highlight-w"> factory<span class="highlight
w"> User.<
pan class="highlight-nf">fromJson(<span class
"highlight-n">Map<<span class="highlight-n
>String, <s
an class="highlight-n">dynamic><span clas
="highlight-w"> json
 =><sp
n class="highlight-w"> _<span class="highlight
err">$UserFromJson<span class="highlight-p"
(json);
<span cla
s="highlight-w">
<span cla
s="highlight-w"> ///<span class="highlight-w"
 `toJson<sp
n class="highlight-o">` <span class="highlight-
">is the<sp
n class="highlight-w"> convention<span class=
highlight-w"> for </
pan>a <span class=
highlight-n">class t
 declare<sp
n class="highlight-w"> support<span class="hi

hlight-w"> for </sp
n>serialization
<span cla
s="highlight-w"> ///<span class="highlight-w"
 to <span cl
ss="highlight-n">JSON.<span class="highlight-
"> The <sp
n class="highlight-n">implementation <span cl
ss="highlight-n">simply <span class="highlight
n">calls the</span
 private<span class=
highlight-p">, gene
ated
<span cla
s="highlight-w"> ///<span class="highlight-w"
 helper <sp
n class="highlight-n">method <span class="hi
hlight-o">`_$</spa
>UserToJson`.
<span cla
s="highlight-w"> Map<span class="highlight-o
><String,
span class="highlight-w"> dynamic<span class=
highlight-o">> t
Json() <spa
 class="highlight-o">=> <span class="highli
ht-n">_$UserToJs
n(this<span
class="highlight-p">);
<span cla
s="highlight-w">
}
</code></pre>

<code>@JsonKey</code>: 指定 json 名称
<code>@JsonSerializable(fieldRename: FieldRename.snake) </code>: json 风格
<code>@JsonKey(defaultValue: false)</code>,<code>@JsonKey(required: true)</code>
<code>@JsonKey(ignore: true)</code>: json 校验
<code>flutter pub run build_runner build</code> 使用这个命令配合 <code>注解</code>
生成,xxx.g.dart
<code>flutter pub run build_runner watch </code> 持续生成
<code>@JsonSerializable(explicitToJson: true)</code>: 子类一起 JSON

<h2 id="i18n">i18n</h2>
<pre><code class="highlight-chroma"><span class="highlight
cl">dependencies:
  flutter:
    sdk: flutter
  flutter_localizati
ns: # Add this line
    sdk: flutter
Add this line

```
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">return const MaterialApp(
</span></span><span class="highlight-line"><span class="highlight-cl">  title: 'Localizations Sample App',
</span></span><span class="highlight-line"><span class="highlight-cl">  localizationsDelegates: [
</span></span><span class="highlight-line"><span class="highlight-cl">    GlobalMaterialLocalizations.delegate,
</span></span><span class="highlight-line"><span class="highlight-cl">    GlobalWidgetsLocalizations.delegate,
</span></span><span class="highlight-line"><span class="highlight-cl">    GlobalCupertinoLocalizations.delegate,
</span></span><span class="highlight-line"><span class="highlight-cl">  ],
</span></span><span class="highlight-line"><span class="highlight-cl">  supportedLocales: [
</span></span><span class="highlight-line"><span class="highlight-cl">    Locale('en', ''), // English, no country code
</span></span><span class="highlight-line"><span class="highlight-cl">    Locale('es', ''), // Spanish, no country code
</span></span><span class="highlight-line"><span class="highlight-cl">  ],
</span></span><span class="highlight-line"><span class="highlight-cl">  home: MyHomePage(),
</span></span><span class="highlight-line"><span class="highlight-cl">);
</span></span></code></pre>
<p>添加自定义国际化文件</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight-cl">dependencies:
</span></span><span class="highlight-line"><span class="highlight-cl">  flutter:
</span></span><span class="highlight-line"><span class="highlight-cl">    sdk: flutter
</span></span><span class="highlight-line"><span class="highlight-cl">  flutter_localizations:
</span></span><span class="highlight-line"><span class="highlight-cl">    sdk: flutter
</span></span><span class="highlight-line"><span class="highlight-cl">  intl: ^0.17.0 # Add this line
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl"># The following section is specific to Flutter.
</span></span><span class="highlight-line"><span class="highlight-cl">flutter:
</span></span><span class="highlight-line"><span class="highlight-cl">  generate: true # add this line
</span></span></code></pre>
<p>然后添加 arb 配置</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight-cl">arb-dir: lib/l10n
</span></span><span class="highlight-line"><span class="highlight-cl">template-arb-file: app_en.arb
</span></span><span class="highlight-line"><span class="highlight-cl">output-localization-file: app_localizations.dart
</span></span></code></pre>
<p>arb 文件</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
```

```
cl">{
</span></span><span class="highlight-line"><span class="highlight-cl">    "helloWorld": "
ello World!",
</span></span><span class="highlight-line"><span class="highlight-cl">    "@helloWorld":

</span></span><span class="highlight-line"><span class="highlight-cl">      "description":
The conventional newborn programmer greeting"
</span></span><span class="highlight-line"><span class="highlight-cl">    }
</span></span><span class="highlight-line"><span class="highlight-cl">}
</span></span></code></pre>
<p>导入本地配置</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">localizationsDelegates: [
</span></span><span class="highlight-line"><span class="highlight-cl">    AppLocalizatio
s.delegate, // Add this line
</span></span><span class="highlight-line"><span class="highlight-cl">``   ],
</span></span></code></pre>
<p>代码中使用</p>
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">Text(AppLocalizations.of(context)!.helloWorld);
</span></span></code></pre>
```