

JAVASCRIPT从前端到全终端

ELECTRON初探&开发者
应该了解的设计



Build cross platform desktop apps
with JavaScript, HTML, and CSS

USE ELECTRON TO BUILD
THE FUTURE JS-DESKTOP APP

SUPPORTED-PLATFORMS



OS X 10.8+



Windows 7+



Ubuntu 12.04

Electron's platforms

基于CHROME V8的MINI浏览器

- ▶ Electron使用javascript + html + css作为App GUI
- ▶ 通过接口控制原生操作系统API
- ▶ 通过js+API进行内存管理以及窗口生命周期的控制

```
const electron = require('electron');
// Module to control application life.
const {app} = electron;
// Module to create native browser window.
const {BrowserWindow} = electron;

// Keep a global reference of the window object, if you don't, the window will
// be closed automatically when the JavaScript object is garbage collected.
let win;

function createWindow() {
  // Create the browser window.
  win = new BrowserWindow({width: 800, height: 600});

  // and load the index.html of the app.
  win.loadURL(`file://${__dirname}/index.html`);

  // Open the DevTools.
  win.webContents.openDevTools();

  // Emitted when the window is closed.
  win.on('closed', () => {
    // Dereference the window object, usually you would store windows
    // in an array if your app supports multi windows, this is the time
    // when you should delete the corresponding element.
    win = null;
  });
}

// This method will be called when Electron has finished
// initialization and is ready to create browser windows.
// Some APIs can only be used after this event occurs.
app.on('ready', createWindow);
```

PROCESS

- ▶ 主进程：应用初始化的脚本，在其中调用控制App生命周期的模块，并首次进行GUI窗口的创建。looks like that:

```
const electron = require('electron');
const app = electron.app;
const BrowserWindow = electron.BrowserWindow;

var window = null;

app.on('ready', function() {
  window = new BrowserWindow({width: 800, height: 600});
  window.loadURL('https://github.com');
});
```

- ▶ 渲染进程：Electron应用中每一个页面都有自己的进程，即渲染进程。渲染进程如同传统的HTML，但它可以直接使用Node模块：

```
<!DOCTYPE html>
<html>
  <body>
    <script>
      const remote = require('electron').remote;
      console.log(remote.app.getVersion());
    </script>
  </body>
</html>
```

PROCESS

- ▶ 主进程通过实例化BrowserWindow，每个BrowserWindow实例都在它自己的渲染进程内返回一个web页面。当BrowserWindow实例销毁时，相应的渲染进程也会终止
- ▶ 在页面（渲染进程）中不允许调用原生GUI相关的API（易造成内存泄露）。如果想在网页中进行GUI的操作，渲染进程必须向主进程传达请求，然后在主进程中完成操作
- ▶ 但在渲染进程中也可以通过Electron提供的模块与主进程进行通讯

```
// remote is used to connect with main-process and render-process
import {remote} from 'electron';

// for example, you can use remote to get the currentWindow
let currentWindow = remote.getCurrentWindow();
console.log(currentWindow); // It's an Object
// some example of window's function
currentWindow.on('focus', () => {
  console.log('I\'m on focus');
});
currentWindow.on('blur', () => {
  console.log('I\'m on blur');
});
currentWindow.on('enter-full-screen', () => {
  console.log('I\'m on Max Size~');
});
```

- ▶ global-shortcut: 快捷键
- ▶ menu: 原生菜单
- ▶ screen: 监测屏幕size/显示/鼠标位置
- ▶ clipboard: 控制复制粘贴
- ▶ ipcMain: 主进程/渲染进程通讯。主进程使用，控制由渲染进程发送的同步或异步消息
- ▶ remote: 主进程/渲染进程通讯。渲染进程使用，用于调用主进程相关的GUI模块
- ▶ dialog: 原生对话框（打开文件/保存/消息/错误信息等多种类型）
- ▶ BrowserWindow: 创建浏览器窗口。可通过参数控制大小/样式/位置/缩放等

Hi,



~~Material~~ Design

THE DEVELOPMENT OF DESIGN

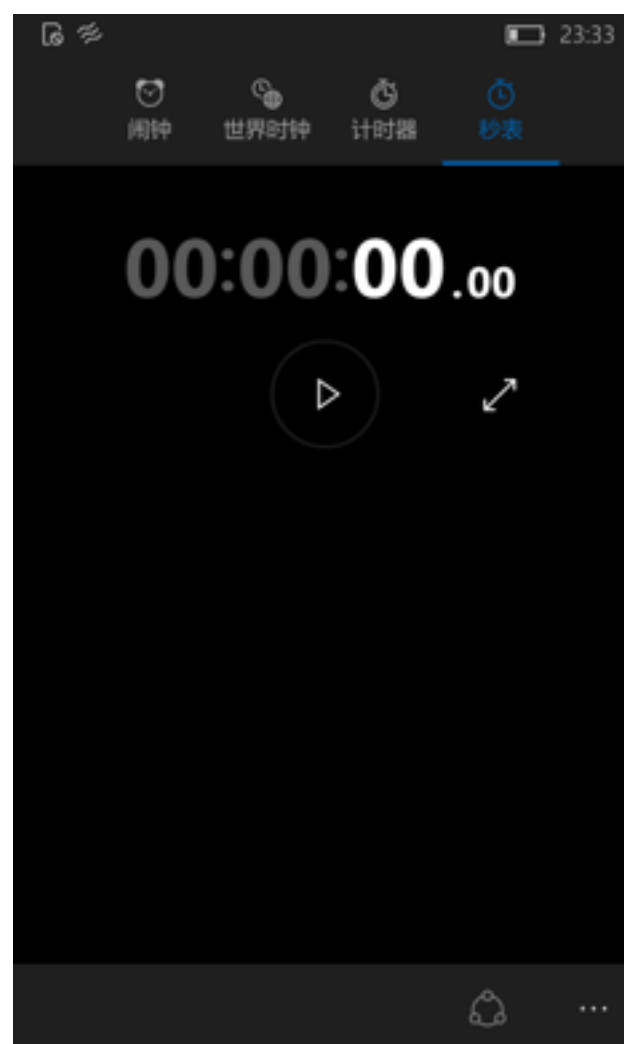
从一开始的拟物化设计到之后的扁平化设计，再到现在的材料设计，设计趋势总是在以一种脱胎换骨的方式进行着改革



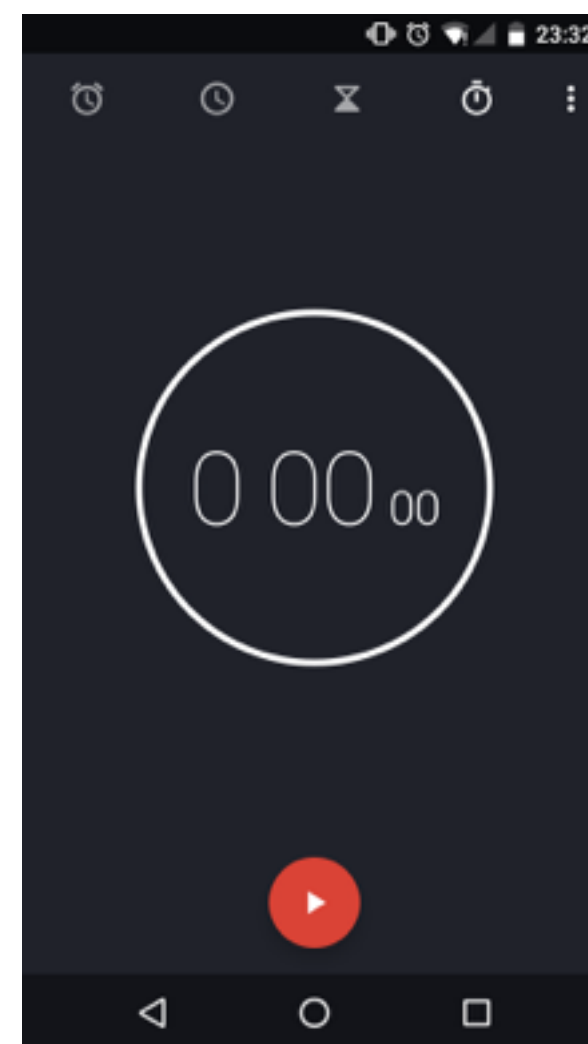
拟物化设计



扁平化设计



材料设计





Music



Email



Calender



Camera



Photoes



Files



Chat



Video



Text



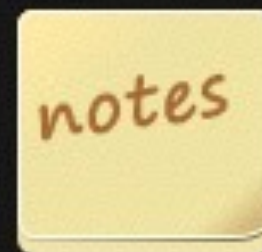
Clock



Setting



Calculator



Note



Contacts



GPS

SKETCHMORPHIC DESIGN

SKETCHOMORPHIC DESIGN

骚

学习成本低

符合当时的大众审美观念

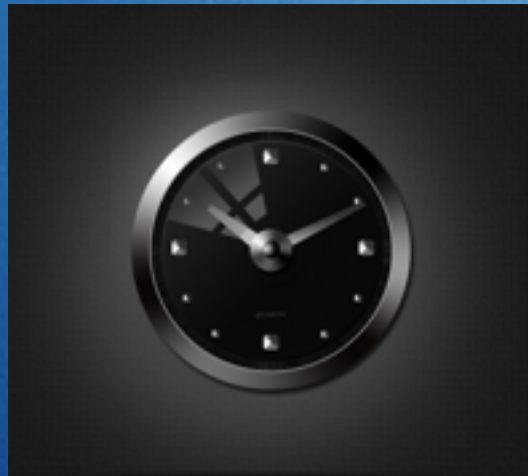
在电子设备以及软件技术飞速发展的时代，还有什么能比在手机、电脑上看见高仿真的设计更令人印象亲切呢？

SKETCHUOMORPHIC DESIGN

拟物化设计，通过阴影、纹理、高光等手法，模仿真实世界物体UI，力求打造真实物体所具有的视觉感官



camera



clock



calendar



game

几乎为0的学习成本。所以知道为什么那么多老人机至今仍是拟物设计了吧？

FLAT DESIGN

拟物化设计极端的对立面

拍平一切，弱化大部分修饰元素，使用单色调和大字体，突出重点信息。

单色调和大字体

使用单色调，杜绝渐变色。使用大字体以及无衬线字体，突出重点信息。

效率优先

非常适合为办公场景以及大信息量的场景服务

FLAT DESIGN

扁平化的发展史

它最早诞生于传统的纸质排版印刷。并逐渐从纸质印刷上，发展到地铁站



FLAT DESIGN

扁平化的发展史

FINALLY，它从现实生活中发展到了我们的电子设备上。以WINDOWS8+以及WINDOWS PHONE 7+为代表，微软试图将这种带有未来感的略“反人类设计”散播到世界





Material Design

为真实的世界而进行的扁平化设计

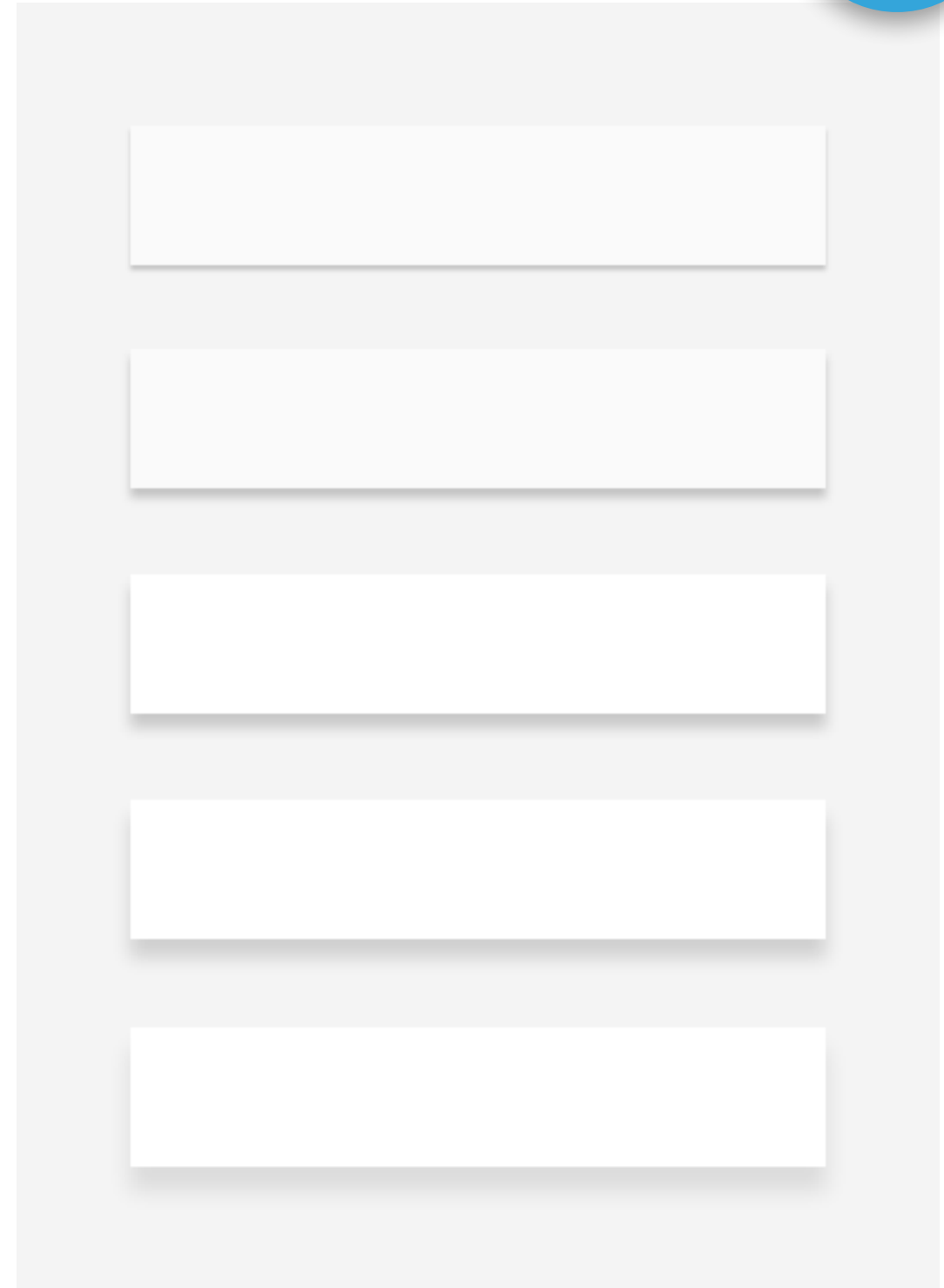
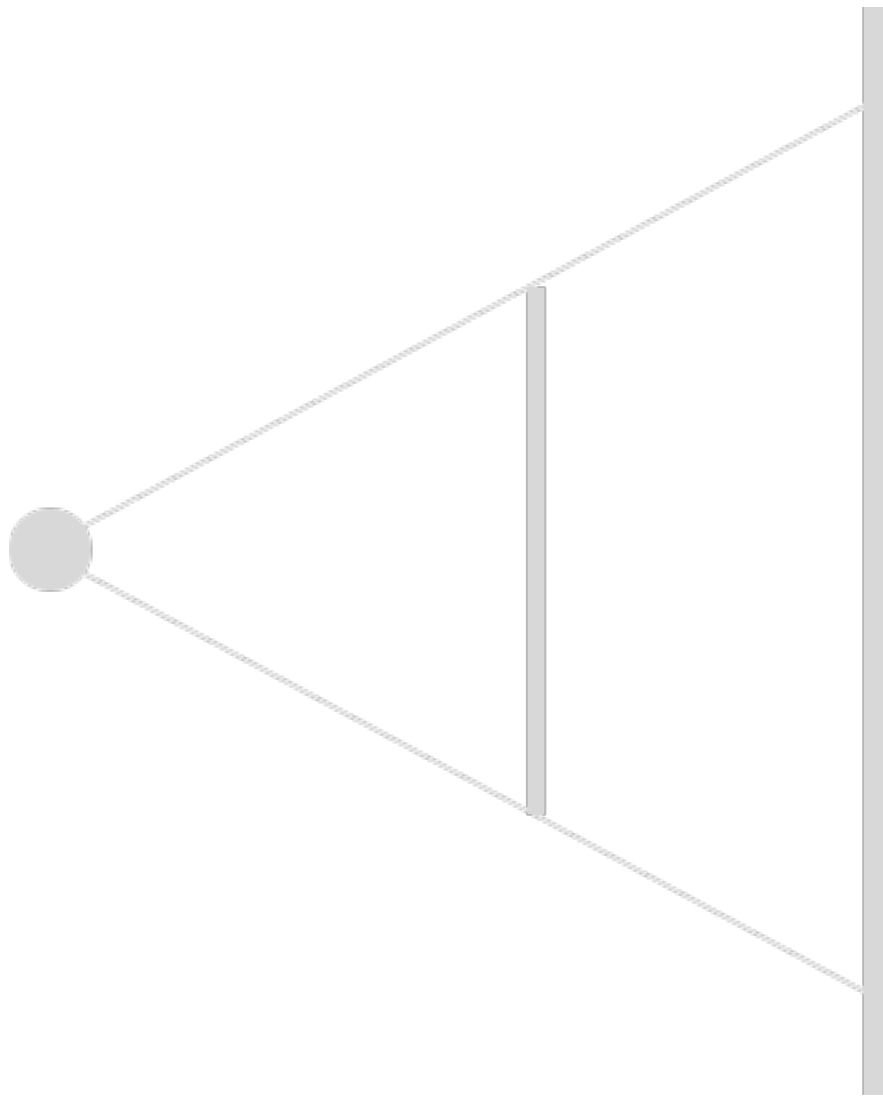
Material Design

材料设计/质感设计

更像是拟物设计与扁平化设计的结合体。它没有拟物设计那样多的光影的修饰，但却适度保留了其阴影以及空间感。除此以外，还从扁平化中吸取了简洁、平面的优势

Material Design

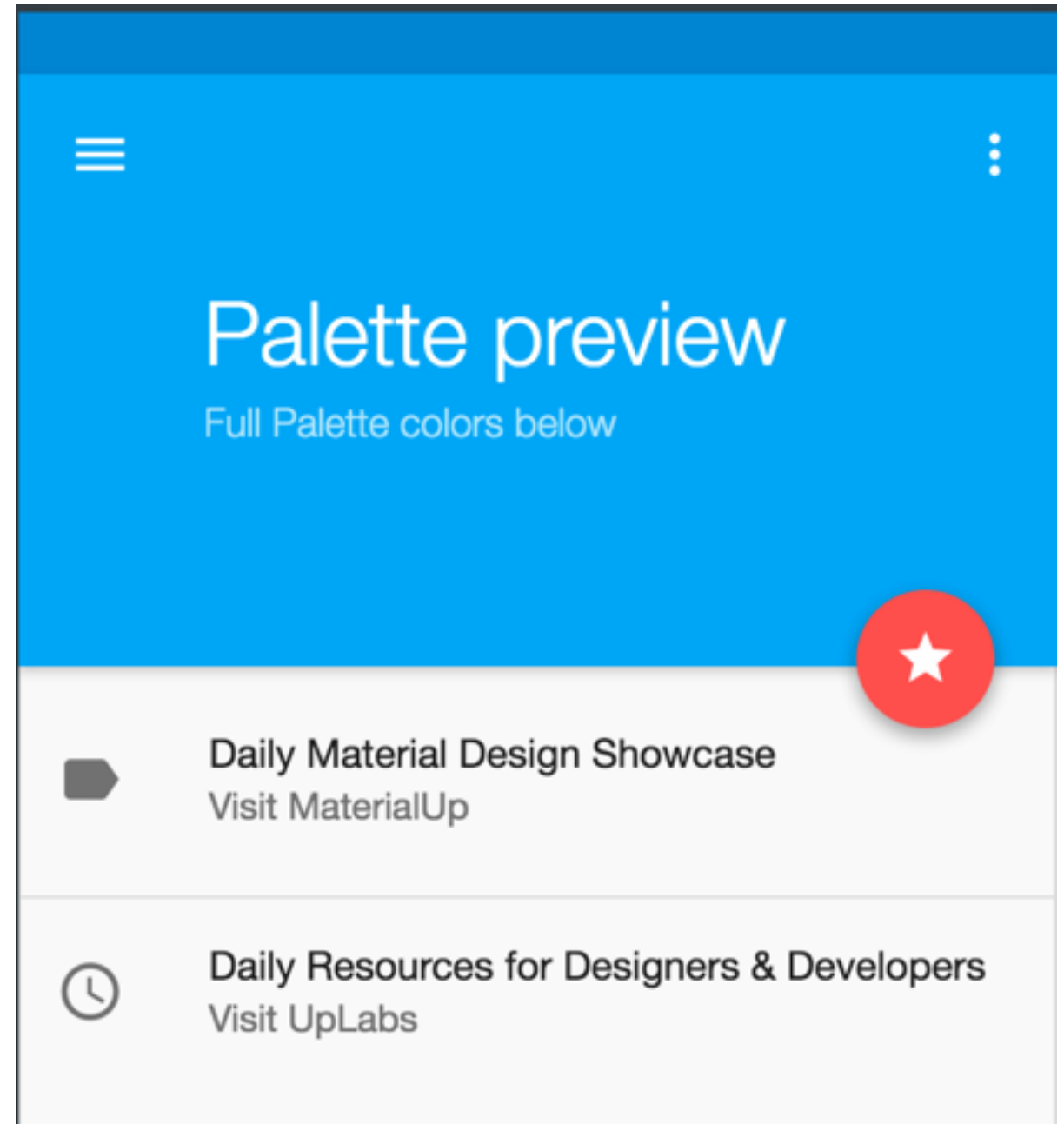
阴影与扁平化



Material Design

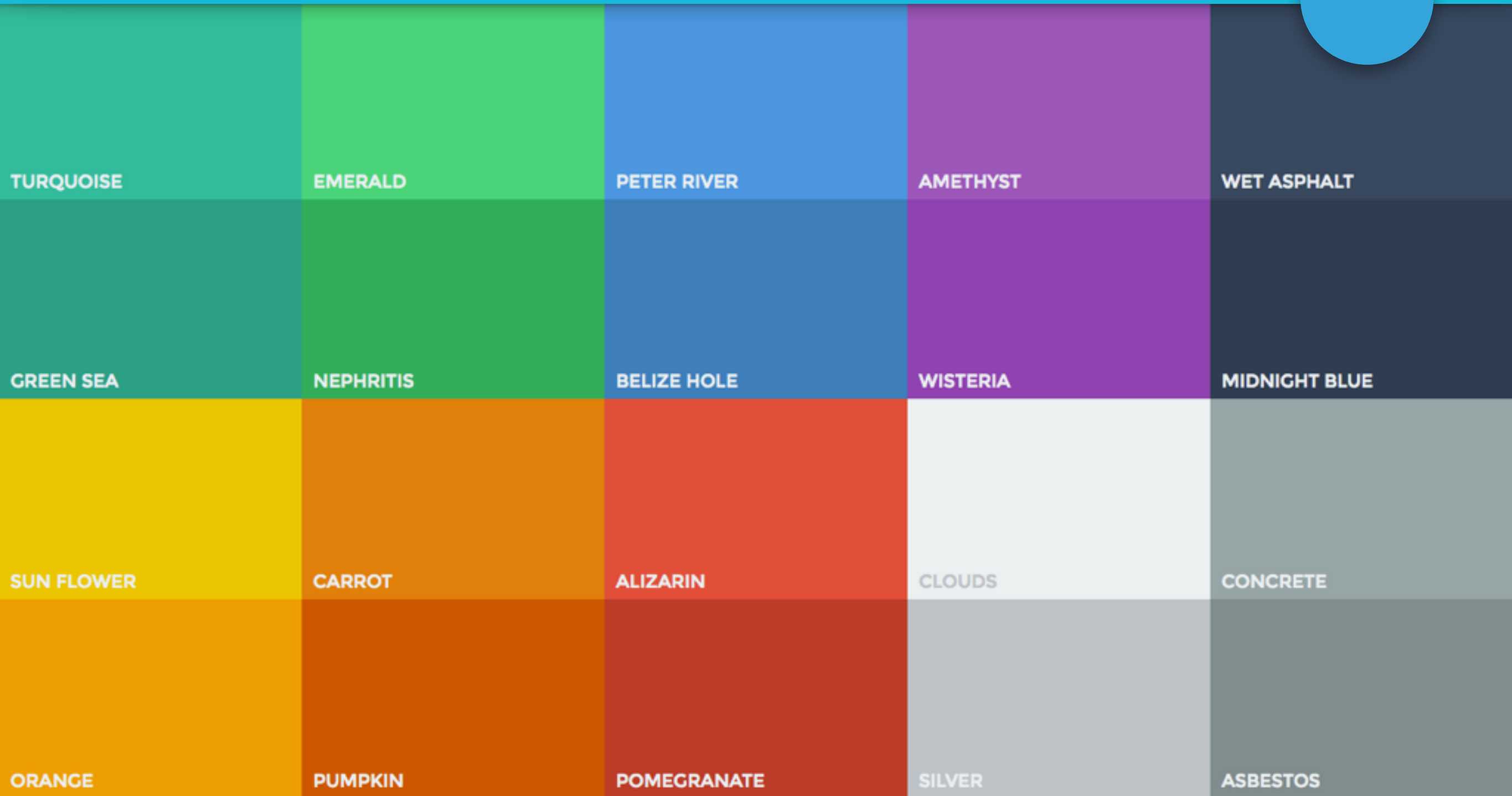
阴影与扁平化

通过阴影深度、
范围传达Z轴的深度，
以此展示真实世界里的物体之间
层级的关系



Material Design

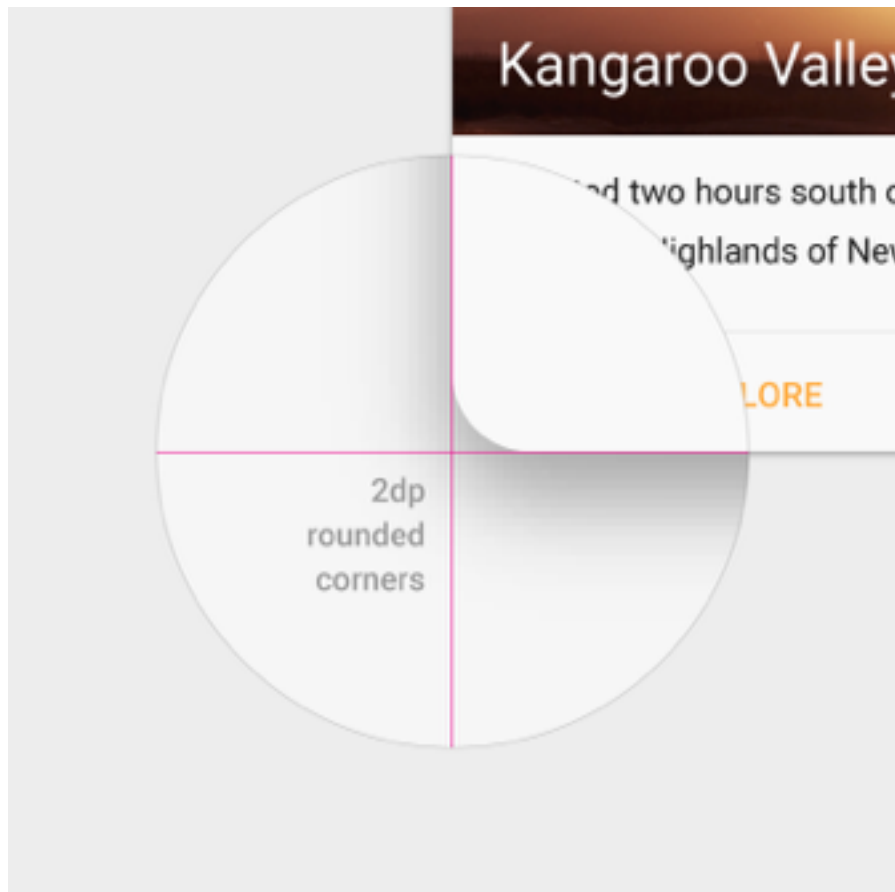
色彩



主色调+辅助重点色 纯色 同色系

Material Design

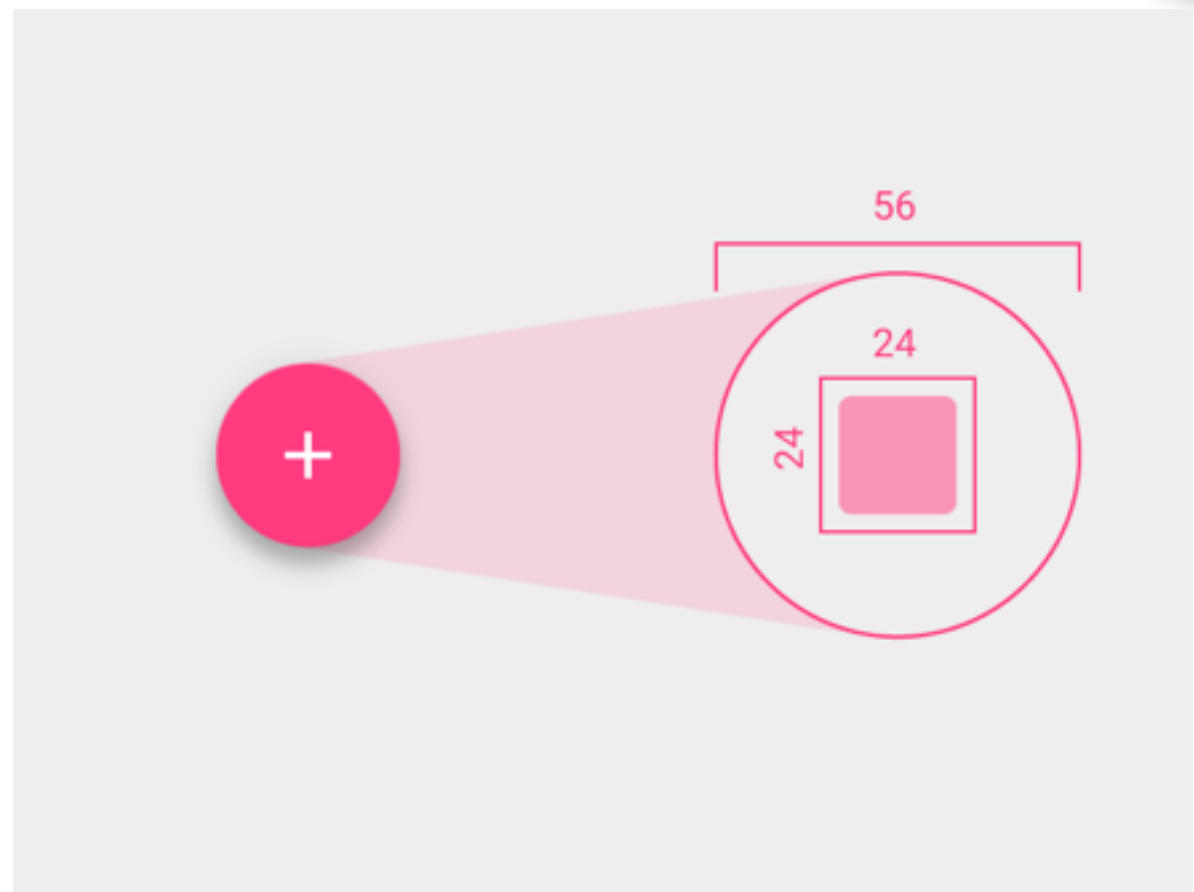
卡片与FAB



圆角

阴影

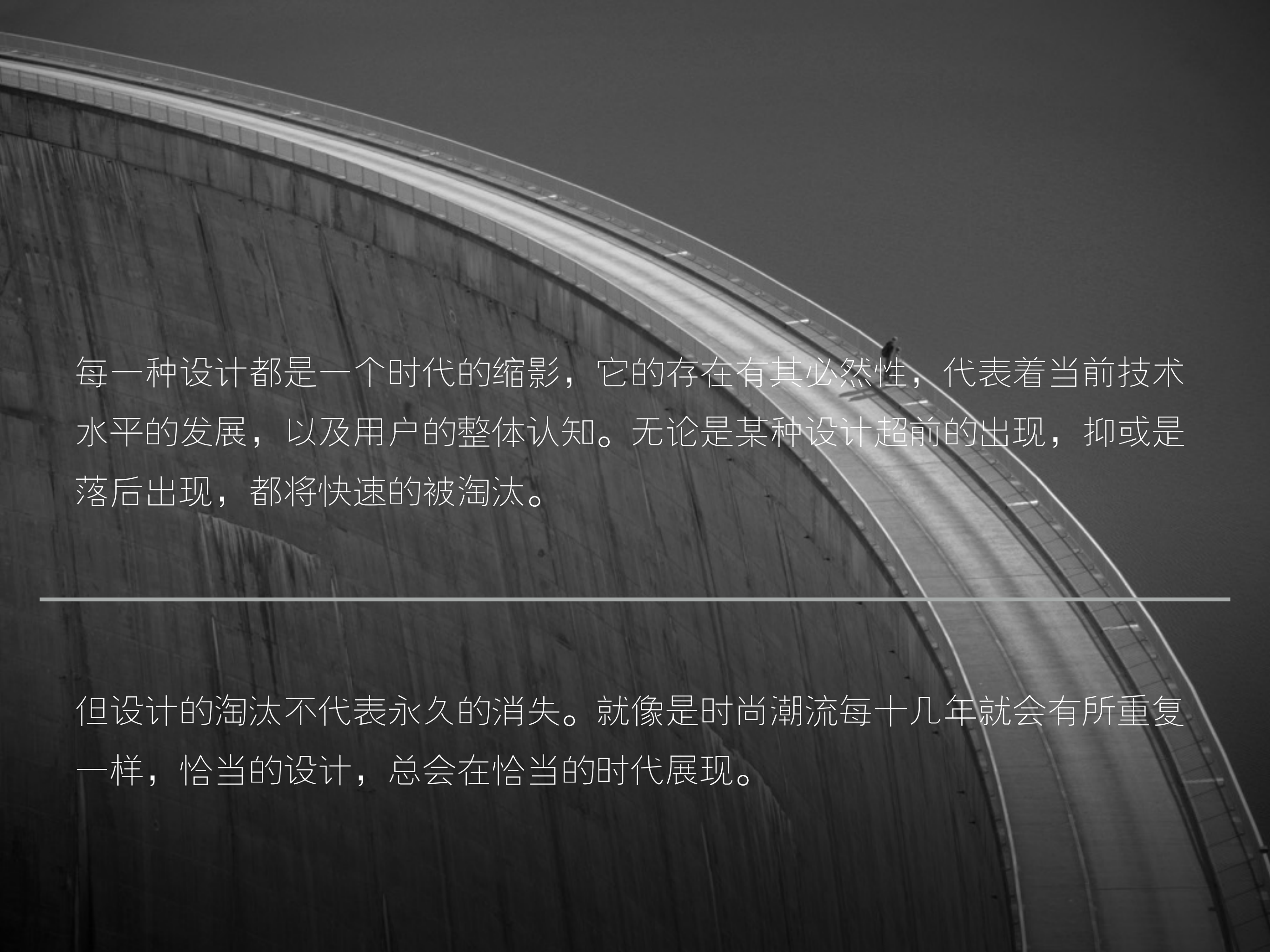
操作



阴影

操作/属性

地位



每一种设计都是一个时代的缩影，它的存在有其必然性，代表着当前技术水平的发展，以及用户的整体认知。无论是某种设计超前的出现，抑或是落后出现，都将快速的被淘汰。

但设计的淘汰不代表永久的消失。就像是时尚潮流每十几年就会有所重复一样，恰当的设计，总会在恰当的时代展现。