前端小程序八股文 （10题）

1. 小程序的生命周期有哪些?

参考答案:

小程序生命周期分为应用生命周期和页面生命周期:

应用生命周期（App）:

- `onLaunch`：小程序初始化完成时触发（全局只触发一次）
- `onShow`：小程序启动或从后台进入前台时触发
- `onHide`：小程序从前台进入后台时触发
- `onError`：小程序发生脚本错误或API调用失败时触发

页面生命周期（Page）:

- `onLoad`：页面加载时触发，只会调用一次
- `onShow`：页面显示/切入前台时触发
- `onReady`：页面初次渲染完成时触发，只会调用一次
- `onHide`：页面隐藏/切入后台时触发
- `onUnload`：页面卸载时触发

---

2. 小程序的双线程架构是什么?

参考答案:

小程序采用双线程架构，分为渲染层和逻辑层:

渲染层（View Thread）：

- 由WebView线程负责
- 处理WXML模板和WXSS样式的渲染
- 处理用户交互事件
- 多个页面对应多个WebView线程

逻辑层（App Service Thread）：

- 由JavaScript引擎线程负责
- 运行JS脚本，处理数据逻辑
- 调用小程序API
- 整个小程序只有一个逻辑层线程

通信机制：

- 两个线程通过系统层的JSBridge进行通信
- 数据传输需要序列化，不能直接传递函数等复杂对象
- 这种架构保证了渲染层的安全性和稳定性

---

3. 小程序的数据绑定机制是怎样的？

参考答案：

小程序使用单向数据绑定机制：

数据绑定语法：

代码块

```
// 页面数据定义
data: {
  message: 'Hello World',
  userInfo: {
    name: '张三',
    age: 25
```

```
  }
}
```

代码块

```
<!-- WXML中使用 -->
<view>{{message}}</view>
<view>{{userInfo.name}}</view>
```

数据更新：

- 必须使用 `setData()` 方法更新数据
- `setData()` 会触发视图层重新渲染
- 支持部分更新：`this.setData({'userInfo.name': '李四'})`

注意事项：

- 不能直接修改 `this.data`
- `setData()` 是异步的
- 频繁调用 `setData()` 会影响性能

---

4. 小程序的路由跳转方式有哪些？

参考答案：

小程序提供多种路由跳转方式：

1. wx.navigateTo()：

- 保留当前页面，跳转到应用内的某个页面
- 可以使用 `wx.navigateBack()` 返回
- 页面栈最多10层

2. wx.redirectTo()：

- 关闭当前页面，跳转到应用内的某个页面
- 不能返回到被关闭的页面

3. wx.switchTab()：

- 跳转到tabBar页面，并关闭其他所有非tabBar页面

4. wx.reLaunch()：

- 关闭所有页面，打开到应用内的某个页面

5. wx.navigateBack()：

- 关闭当前页面，返回上一页面或多级页面

参数传递：

代码块

```
wx.navigateTo({
  url: '/pages/detail/detail?id=123&name=test'
})
```

---

5. 小程序的组件化开发如何实现？

参考答案：

小程序支持自定义组件开发：

组件创建：

代码块

```
// component.js
Component({
  properties: {
    title: {
      type: String,
      value: '默认标题'
    }
  },
  data: {
    count: 0
  },
  methods: {
    increment() {
      this.setData({
        count: this.data.count + 1
      })
      // 触发自定义事件
      this.triggerEvent('countchange', {
        count: this.data.count
      })
    }
  }
})
```

组件使用：

代码块

```
// 页面json配置
{
  "usingComponents": {
    "my-component": "/components/my-component/my-component"
  }
}
```

代码块

```
<!-- 页面wxml -->
<my-component title="自定义标题" bind:countchange="onCountChange"></my-component>
```

组件通信：

- 父传子：通过properties
- 子传父：通过triggerEvent触发自定义事件
- 获取组件实例：使用selectComponent

---

6. 小程序的性能优化策略有哪些？

参考答案：

小程序性能优化主要从以下几个方面：

1. 代码包优化：

- 清理无用代码和资源
- 图片压缩，使用webp格式
- 分包加载，按需加载
- 使用CDN存储大文件

2. 渲染优化：

- 减少setData的调用频率和数据量
- 避免频繁切换显示隐藏
- 使用wx:if而不是hidden控制大块内容
- 长列表使用虚拟化或分页

3. 网络优化：

- 合并网络请求
- 使用缓存策略
- 预加载关键数据
- 图片懒加载

4. 内存优化：

- 及时清理定时器和监听器
- 避免内存泄漏
- 合理使用全局数据

5. 体验优化：

- 添加loading状态
- 骨架屏占位
- 错误边界处理

---

7. 小程序的存储方式有哪些？

参考答案：

小程序提供多种数据存储方式：

1. 本地存储：

代码块

```
// 同步存储
wx.setStorageSync('key', 'value')
const value = wx.getStorageSync('key')
wx.removeStorageSync('key')
wx.clearStorageSync()

// 异步存储
wx.setStorage({
  key: 'key',
  data: 'value'
})
```

2. 全局数据：

代码块

```
// app.js
App({
  globalData: {
    userInfo: null
  }
})

// 其他页面使用
const app = getApp()
app.globalData.userInfo = userInfo
```

3. 页面数据：

- 通过data定义
- 使用setData更新

4. 云存储：

- 云开发数据库
- 云文件存储

存储限制：

- 本地存储上限10MB
- 单个key存储上限1MB
- 数据类型支持Object、String、Number等

---

8. 小程序的网络请求如何处理？

参考答案：

小程序网络请求主要使用wx.request()：

基本用法：

代码块

```
wx.request({
  url: 'https://api.example.com/data',
  method: 'GET',
  data: {
    id: 123
  },
  header: {
    'content-type': 'application/json'
  },
  success: (res) => {
    console.log(res.data)
  },
  fail: (err) => {
    console.error(err)
  }
})
```

封装请求：

代码块

```
const request = (url, options = {}) => {
  return new Promise((resolve, reject) => {
    wx.request({
      url: baseURL + url,
      method: options.method || 'GET',
      data: options.data || {},
      header: {
        'Authorization': wx.getStorageSync('token'),
        ...options.header
      },
      success: resolve,
      fail: reject
    })
  })
}
```

注意事项：

- 域名必须在后台配置合法域名
- 默认超时时间60秒
- 并发请求限制10个
- 支持HTTP/HTTPS，生产环境必须HTTPS

---

9. 小程序的事件系统是怎样的？

参考答案：

小程序事件系统包括事件绑定、事件对象和事件传播：

事件绑定：

代码块

```
<!-- 冒泡事件 -->
<view bindtap="handleTap">点击我</view>
<!-- 非冒泡事件 -->
<view catchtap="handleTap">点击我</view>
<!-- 互斥事件 -->
<view mut-bind:tap="handleTap">点击我</view>
```

事件对象：

代码块

```
handleTap(event) {
  console.log(event.type) // 事件类型
  console.log(event.target) // 触发事件的组件
  console.log(event.currentTarget) // 绑定事件的组件
  console.log(event.detail) // 额外信息
  console.log(event.touches) // 触摸事件的触摸点信息
}
```

数据传递：

代码块

```
<view bindtap="handleTap" data-id="123" data-name="test">
  点击我
</view>
```

代码块

```
handleTap(event) {
  const { id, name } = event.currentTarget.dataset
  console.log(id, name) // 123, test
}
```

事件类型：

- 触摸事件：touchstart、touchmove、touchend
- 点击事件：tap、longpress
- 表单事件：input、submit、reset
- 自定义事件：通过triggerEvent触发

---

10. 小程序与H5的区别是什么？

参考答案：

小程序与H5在多个方面存在显著区别：

运行环境：

- 小程序：运行在微信客户端，双线程架构
- H5：运行在浏览器中，单线程

开发技术：

- 小程序：WXML、WXSS、JavaScript
- H5：HTML、CSS、JavaScript

功能权限：

- 小程序：可调用更多原生API（摄像头、位置、支付等）
- H5：受浏览器安全策略限制

性能表现：

- 小程序：接近原生应用体验，启动更快
- H5：依赖网络加载，可能存在白屏

分发方式：

- 小程序：通过微信分发，无需下载安装
- H5：通过链接访问，需要浏览器

开发限制：

- 小程序：代码包大小限制、API限制
- H5：跨域限制、浏览器兼容性

用户体验：

- 小程序：统一的交互规范，体验一致
- H5：体验取决于具体实现和浏览器