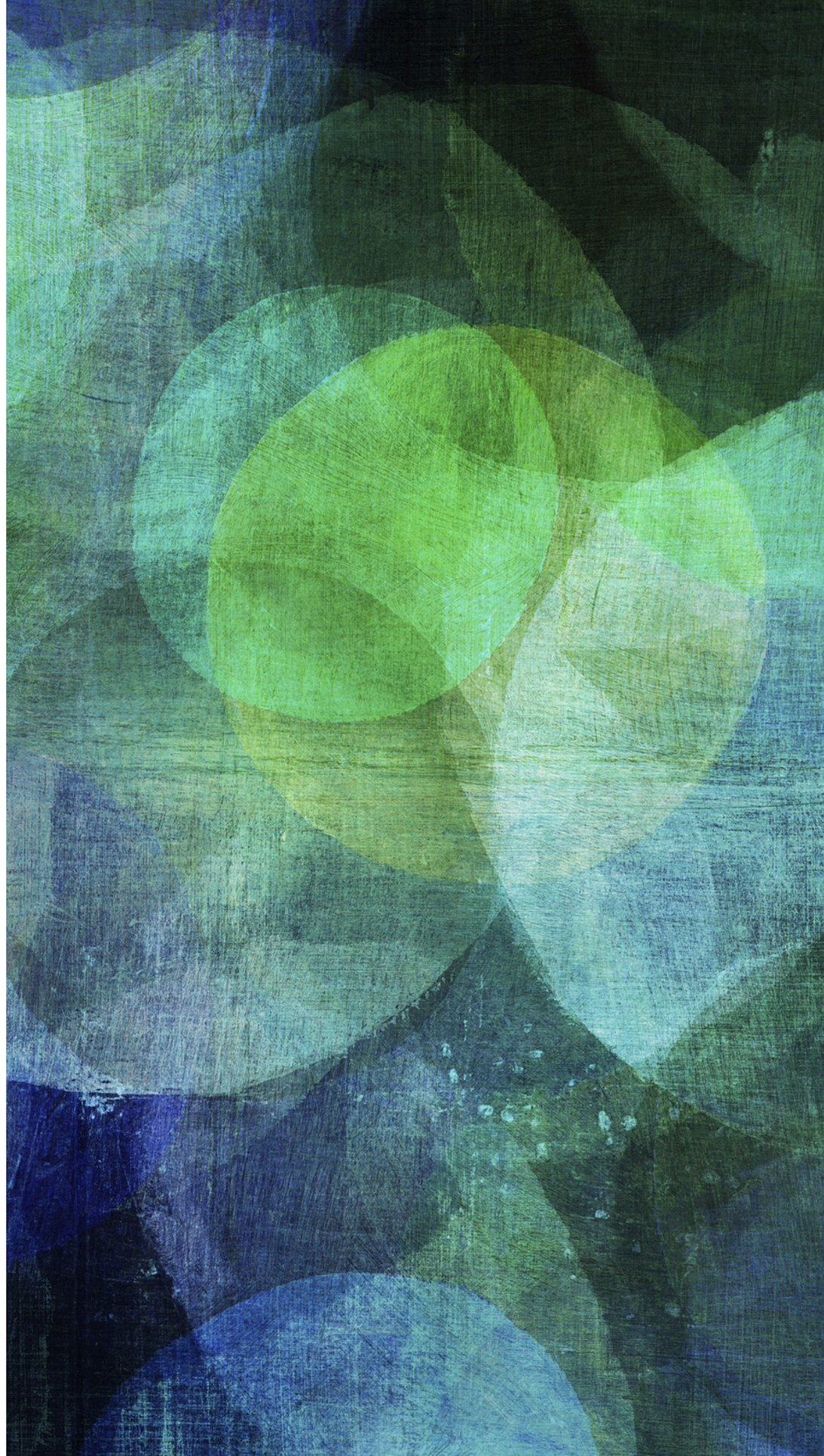
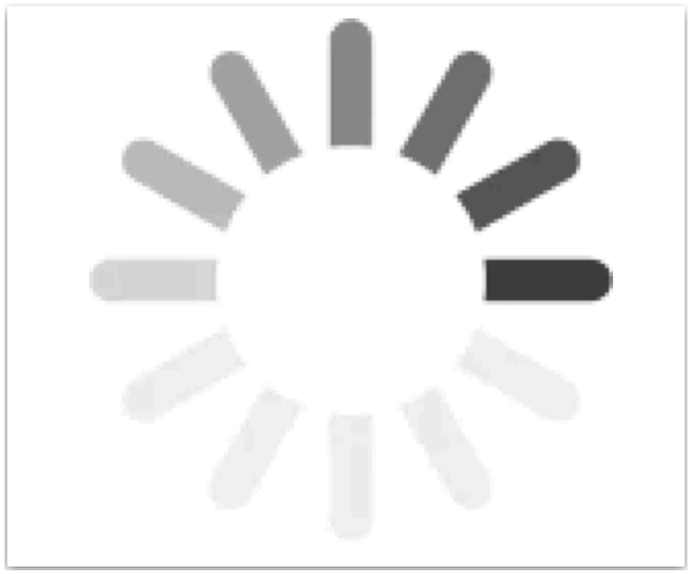
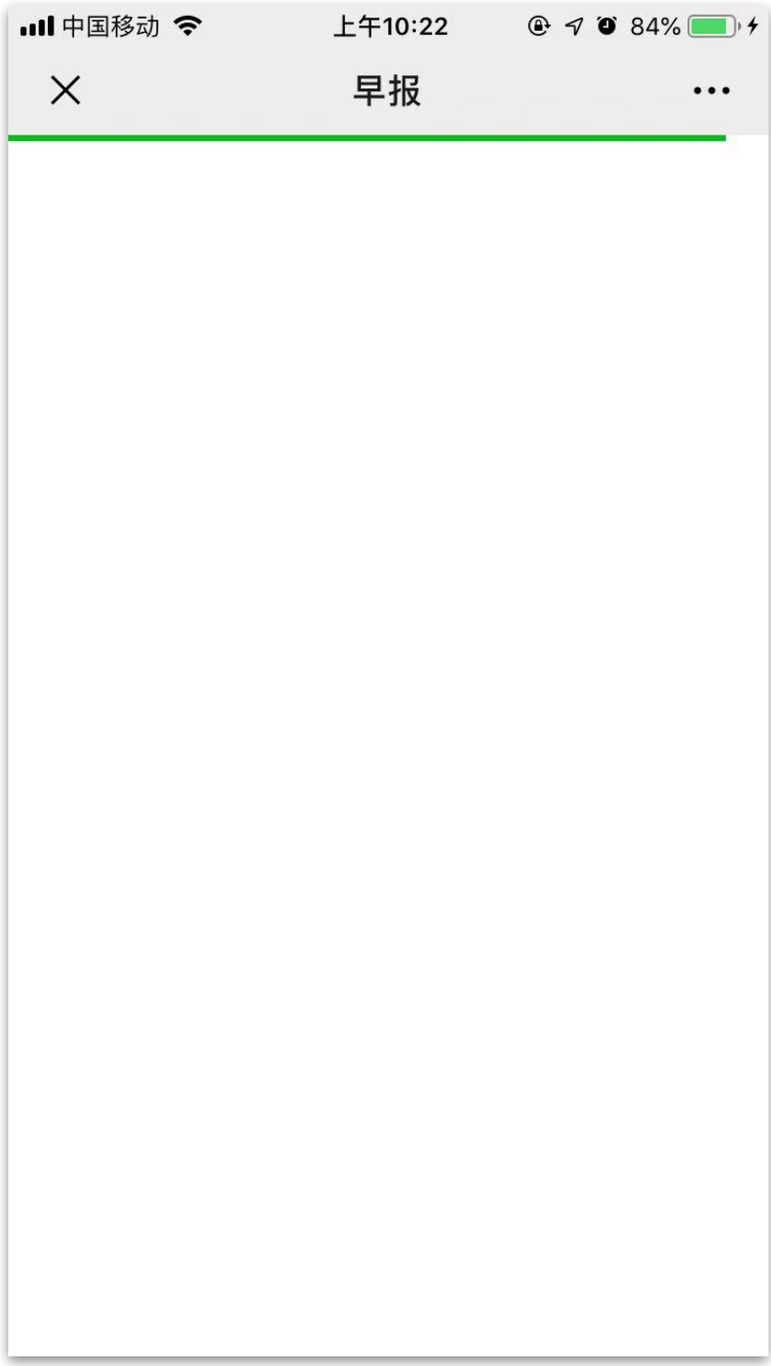
骨架屏

骨架屏（*skeleton screen*）实践探索

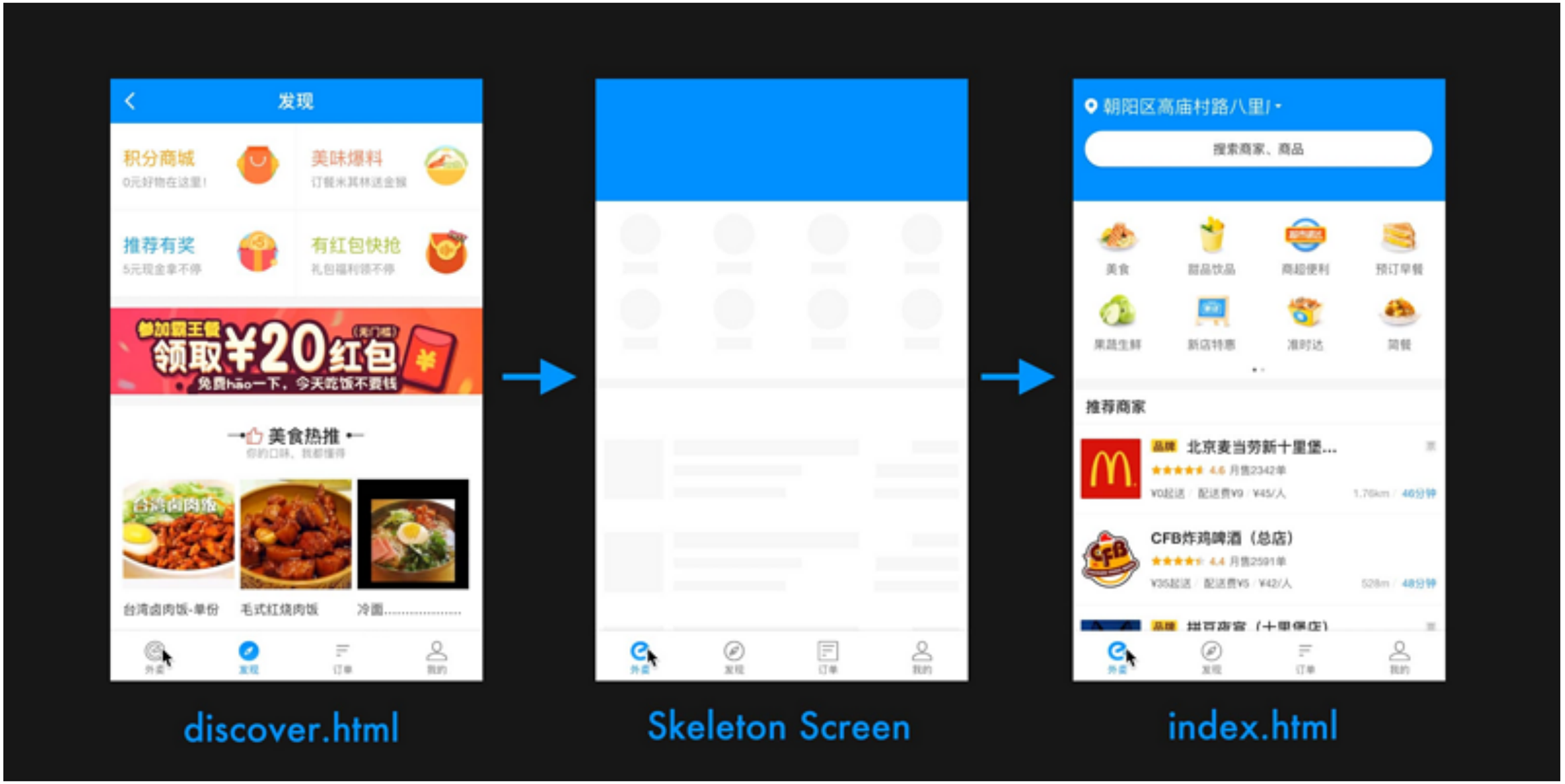


一般的LOADING状态





骨架屏


发现
积分商城
0元好物在这里!
美味爆料
订餐米其林送金猴
推荐有奖
5元现金拿不停
有红包快抢
礼包福利领不停
参加霸王餐
领取¥20红包
免费hāo一下，今天吃饭不要钱
美食热推
台湾卤肉饭-单份
毛式红烧肉饭
冷面.....
外卖
发现
订单
我的
discover.html
Skeleton Screen
朝阳区高庙村路八里
搜索商家、商品
美食
甜品饮品
商超便利
预订早餐
果蔬生鲜
新店特惠
准时达
简餐
推荐商家
北京麦当劳新十里堡...
CFB炸鸡啤酒（总店）
index.html

1.vue-content-loader/react-content-loader

Create vue Content Loader

文档

Svg!!!

可缩放的矢量图形:它是基于XML，由W3C进行开发的,严格来说应该是一种开放标准的矢量图形语言。


程序，你快下来吧
我们不改功能了
你们立字据
策划
产品经理
经理

饿了么骨架屏实现方式:

➤ 基本将页面分成以下不同的块

- 文本块：仅包含文本节点（NodeType 为 Node.TEXT_NODE）的元素（NodeType 为 Node.ELEMENT_NODE），一个文本块可能是一个 p 元素也可能是 div 等。文本块将会被转化为灰色条纹。

- 图片块：图片块是很好区分的，任何 img 元素都将被视为图片块，图片块的颜色将被处理成配置的颜色，形状也被修改为配置的矩形或者圆型。

- 按钮块：任何 button 元素、 type 为 button 的 input 元素，role 为 button 的 a 元素，都将被视为按钮块。按钮块中的文本块不在处理。

- svg 块：任何最外层是 svg 的元素都被视为 svg 块。

- 伪类元素块：任何伪类元素都将视为伪类元素块，如 ::before 或者 ::after。

- ...

文本块的骨架结构生成


饿了么是中国大陆的一个订餐平台
由拉扎斯网络科技（上海）有限公司运营。该网站于2009年4月创立，以O2O为其经营模式。

文本块的骨架结构生成

- 单行文本内容的高度，可以通过 fontSize 获取到。

- 单行文本内容加空白间隙的高度，可以通过 lineHeight 获取到。

- p 元素总共有多少行文本，也就是所谓行数，这个可以通过 p 元素的（height – paddingTop – paddingBottom）/ lineHeight 大概算出。

- 文本的 textAlign 属性。

Window.getComputedStyle()

图片块的骨架生成

- 1. DIV 元素来替换 IMG 元素,设置背景色为黑色
- 缺点: 原来通过元素选择器设置到 IMG 元素上的样式无法运用到 DIV 元素上面

- 2.Canvas 来绘制和原来图片大小相同的灰色块
- 缺点: 生成的HTML文件过大

- 3.将一张1 * 1 像素的 gif 透明图片，转化成 dataUrl ,放到src上

注入骨架屏

```
SkeletonPlugin.prototype.apply = function (compiler)
{
  // 其他代码
  compiler.plugin('after-emit', async (compilation,
done) => {
    try {
      await outputSkeletonScreen(this.originalHtml,
this.options, this.server.log.info)
    } catch (err) {
      this.server.log.warn(err.toString())
    }
    done()
  })
  // 其他代码
}
```

➤

注入骨架屏

```
const outputSkeletonScreen = async (originHtml, options, log) => {
  const { pathname, staticDir, routes } = options
   // 遍历路由
  return Promise.all(routes.map(async (route) => {
     // 格式化路由
    const trimedRoute = route.replace(/\//g, '')
     // 获取页面路径
    const filePath = path.join(pathname, trimedRoute ? `${trimedRoute}.html` :
'index.html')
     // 获取页面
    const html = await promisify(fs.readFile)(filePath, 'utf-8')
     // 拿页面替换锚点
    const finalHtml = originHtml.replace('<!-- shell -->', html)
     // 输出路径
    const outputDir = path.join(staticDir, route)
     //输出文件
    const outputFile = path.join(outputDir, 'index.html')
     //确定输出路径存在
    await fse.ensureDir(outputDir)
     // 写入文件
    await promisify(fs.writeFile)(outputFile, finalHtml, 'utf-8')
    log(`write ${outputFile} successfully in ${route}`)
    return Promise.resolve()
  }))
}
```

➤