**Web无障碍开发漫游指南下卷**

20190523.1

目录

[一、无障碍背景介绍 3](#_Toc9510960)

[1.1. 社会背景 3](#_Toc9510961)

[1.2. 无障碍标准 3](#_Toc9510962)

[1.2.1. 国家无障碍标准 3](#_Toc9510963)

[1.2.2. WCAG及其他地区标准 4](#_Toc9510964)

[1.3. 无障碍术语 5](#_Toc9510965)

[二、无障碍改造 9](#_Toc9510966)

[2.1. 技术背景 9](#_Toc9510967)

[2.1.1. 屏幕阅读器及其原理 9](#_Toc9510968)

[2.1.2.可访问富互联网应用（WAI-ARIA）基础 10](#_Toc9510969)

[2.2 改造实施 24](#_Toc9510970)

[2.2.1. 改造综述 24](#_Toc9510971)

[2.2.2. 步骤一：使用无障碍组件 25](#_Toc9510972)

[2.2.3. 步骤二：开发Checklist 30](#_Toc9510973)

[2.2.4. 步骤三：页面其他内容 34](#_Toc9510974)

[三、无障碍检验及测试 50](#_Toc9510975)

[3.1. 开发辅助工具 50](#_Toc9510976)

[3.1.1. 静态分析工具 50](#_Toc9510977)

[3.1.2. 浏览器检测工具 50](#_Toc9510978)

[3.1.3. 视觉测试工具 51](#_Toc9510979)

[3.2. 整体测试方法 51](#_Toc9510980)

[3.2.1. 测试样本选择 51](#_Toc9510981)

[3.2.2 多层次测试检验 52](#_Toc9510982)

# 一、无障碍背景介绍

## 1.1. 社会背景

网络是对所有人开放的，可访问性（Accessibility）指的是让网站对尽可能多的人而言是可用的。无论有怎样身体条件、处于什么样的现实环境，比如有视觉障碍的人，坐轮椅的人，抱着婴儿的成人等，每一个人都应该拥有相同的机会能够访问网站，本文主要讨论 PC Web 的无障碍。

有障碍的访问用户可分为以下几类：

|  |  |
| --- | --- |
| **类型** | **举例** |
| 视觉障碍 | 色弱、色盲、完全失明 |
| 身体障碍 | 无法使用鼠标甚至键盘 |
| 认知障碍 | 学习困难或记忆力差 |
| 读写障碍 | 无法阅读，难以认字 |
| 听力障碍 | 耳聋，听不到或听不清 |

## 1.2. 无障碍标准

通过前面的介绍，我们知道了如何使用 WAI-ARIA 来使得网页更具无障碍性，但是在进行无障碍开发与检测时，我们还需要了解相应的标准规范，才能具体知道哪些无障碍改造是必须要做的，哪些改造是视情况而定的，这有利于各位开发者根据需求，高效的完成改造的过程。

### 1.2.1. 国家无障碍标准

目前我国的无障碍国标也正处于报批阶段，其中分为一级、二级、三级这三个等级。

其中，一级须满足对应 20 项指标；二级须满足一级全部指标与二级 17 项指标；三级需满足全部 59 项指标。

### 1.2.2. WCAG及其他地区标准

在 Web 无障碍领域，有一份由 WAI (Web Accessibility Initiative) Web 可访问性和 W3C (World Wide Web Consortium) 万维网联盟发布的创建可访问性网站的权威指南WCAG (Web Content Accessibility Guidelines) Web 内容可访问性指南，来约束不同用户代理（浏览器，屏幕阅读器）的行为。WCAG 目前的最新版本是版本 2.1，可以在[官网](https://www.w3.org/TR/WCAG21/)找到。

WCAG 实际上可以当做是一个方法论，比如说这个产品要满足哪些点，对比度要达到什么样的要求，错误提示应该给出一种什么样的形式。它里面可以分为四块，像一个方法论，可以用来检查自己的产品在信息无障碍这块做得怎么样：

* 可感知性
* 可操作性
* 可理解性
* 鲁棒性

在 WCAG 中，规定了系统可访问性必须遵循的规则。主要分为 A、AA、AAA 三个等级：

* A 是最低一致性要求，往往对设计影响很小，对于那些不使用辅助技术的人来说可能看不到
* AA 和 AAA 标准的限制越来越严格，满足每个标准可能是不现实的，WCAG 官方也不建议将 AAA 级合规性作为整个网站的一般策略

因此，我们推荐网站至少满足 A 级标准，对于部分页面、功能再渐进增强满足更高标准的要求，比如 AA 甚至 AAA。

不同国家地区针对无障碍都颁布了自己的法律法规，比如美国有 508 无障碍法案，更多的信息可查阅中国残联网站上的[相关国家和地区的无障碍法规](http://www.cdpf.org.cn/special/wzajstl/xggjhdqwzafg/)。

## 1.3. 无障碍术语

在无障碍开发中，会经常碰到一些术语、首字母缩略词、缩写词。了解它们的含义更利于您之后的无障碍开发。

* 网页 Page

指使用HTTP从单一URI获取的非嵌入式资源，以及由用户代理用于呈现或为了与非嵌入式资源一起呈现的其他资源，是构成网站的基本要素。

* 移动设备 Mobile Device

小到能够随身携带的小型计算设备，通常能通过显示屏、触摸屏、或数字字母键盘进行用户交互操作并提供多种功能。

* 移动应用 Mobile Application

在移动设备上能够运行的应用软件。

* 辅助技术 Assistive Technology

作为用户代理或与主流用户代理协同工作的硬件、软件，以提供满足残障用户需求的功能，这些需求常超出主流用户代理所提供的功能要求。

* 用户代理 User Agent

为用户检索和呈现网页内容的任何软件。

* 可访问性 Accessibility

互联网内容对于用户（包括残障用户、老年人和其他用户）的可阅读性和可理解性，互联网内容被通过用户代理进行访问时，向用户提供信息的能力。

* 网页无障碍 Web Accessibility

所有人，不管身体能力有没有障碍，都可以访问基于主页的信息和服务。主页无障碍通过适应性技术在主页的应用而达到。

* 非文本内容 Non-text Content

任何非字符序列的内容，该字符序列可编程式确定，或者用于序列不是表达人类语言含义的地方。

* 替代文本 Text Alternative

可以编程式确定、用于替代非文本内容的文本，或用于补充非文本内容并可以从通过编程式确定的文本中进行引用的文本。评测过程中往往以<alt>、<title>、<caption>标签中的文本内容为例，以及HTML5中<figcaption>标签中的文本为基础（HTML5限定）。

注：操作可以是过程或服务操作。

* HTML5导航结构标签HTML 5 Navigation Structure Tag

用于帮助浏览器快速访问不同页面内容的HTML5标签。

* 辅助工具 Assistive Tool

辅助进行某项任务、某项操作时所需要使用的工具。为了满足存在身体机能差异性的特殊人群（例如残疾人）的需要，在主流用户代理提供的功能之外，它们提供了一些特殊的功能，也可以与主流用户代理一起提供特殊的功能。本标准涉及的比较重要的辅助工具包括：

——屏幕放大器，由存在视力障碍的人使用，可以放大屏幕上的字体并改变颜色，以改善改变后的文本和图像的视觉可读性；

——读屏软件，由存在视力障碍或阅读障碍的人使用，以便通过同步语音或盲文显示来读取文字信息；

——语音识别软件，可以由存在某些生理障碍的人使用；

——替代性键盘，由存在某些生理障碍的人使用，以便模拟键盘的功能；

——替代性指点设备，由存在某些生理障碍的人使用，以模拟鼠标定位和按钮动作。

* 点显器 Braille Display

典型的替代性指点设备，能够将计算机上的信息用盲文同步显示，以便盲人摸读。可与读屏软件配合使用，将读屏软件获得文字以盲文形式展示给用户。

* 非文本链接 Non-TextLinks

任何非字符序列的链接，该链接采用非文本内容表示，如图片链接，控件链接，动画链接等。

* 装饰性内容 Decorated Content

用于装饰的元素，只满足美观的目的，不提供信息，不执行功能，如广告悬浮窗，特效动画等。

* 预录多媒体 Pre-record Media

事先录制好的音频或视频，用于呈现信息的交互式组件。

* 实时多媒体 Synchronous Media

与其他格式实时同步的音频或视频，用于呈现信息或基于时间的交互式组件，除非该媒体是文本的替代媒体，该替代已被清楚地标示。

* 焦点 Focus

电脑操作系统或应用程序需要获得鼠标、键盘或者其他类型的输入设备（比如手写笔）在屏幕（或窗口）上输入信息的位置，这个位置被称作输入焦点，操作系统或应用程序获得输入焦点的行为被称作聚焦。通常鼠标的单击会使系统自动获得焦点，光标的移动也会使系统自动获得焦点。

* 组件 Component

软件系统中具有相对独立功能、可独立部署、可组装的软件实体。包括：

——用户界面组件：帮助完成软件开发中视窗、文本框、按钮、下拉式菜单等UI界面元素的开发；

——图表组件：帮助软件实现数据可视化，实现开发时较难独立完成的复杂图表；

——报表组件：在软件中实现报表的浏览查看、设计、编辑、打印等功能；

——表格组件：主要实现数据处理和操作的功能。

* 控件 Control

可视化的组件，拥有自己的属性和方法。

* 网页模板 Webpage Template

当网站中有许多页面版式色彩相同的情况下，将其定义为网页模板，并定义其中部分可编辑，部分不可利用网页模板制作其他页面时就会很方便，不易出错。

* 布局 Layout

APP内容中所有对组件进行规划和安排的有关事项。

* 触摸 Touch

用户在滑动屏幕时，触摸点落到APP内容中的某组件上的行为。

* 聚焦框 Focus Frame

在用户代理接入下，APP内容中某组件的可触摸区域，点选该区域中任一点可对该组件进行聚焦。

* CSS像素点 CSS Pixel

CSS像素是CSS中所有长度和测量的规范度量单位。该单元与密度无关，与显示器中存在的实际硬件像素不同。考虑显示器的物理大小和预设的观看距离，用户代理和操作系统应确保将CSS像素设置为尽可能接近CSS值和单位模块3级参考像素 [ css3-values ]。

* 单指针激活 single pointer activation

与屏幕的一个接触点（与多点触摸）。指针可以是由鼠标光标，笔，触摸（包括多点触摸）或其他指示输入设备制成的屏幕上的任何接触点。这种模式使得无论用户拥有什么硬件，都能轻松编写能够正常工作的站点和应用程序。

* A11y

代表 Accessibility，数字 11 代表的是这个单词的字母 a 和字母 y 之间的 11 个字符

* W3C

World Wide Web Consortium，万维网联盟，互联网的主要国际标准组织。

* WAI

Web Accessibility Initiative, 无障碍网页倡议，由 W3C 发起的旨在改善残疾人对万维网的可访问性的倡议。

* WCAG

Web Content Accessibility Guidelines，网页内容可访问性指引，由 WAI 发布的 Web 可访问性指南。

* WAI-ARIA

Accessible Rich Internet Applications，W3C 发布的技术规范，指定如何增加网页的可访问性，特别是动态内容和用以下开发的用户界面组件

* AT

代表 [Assistive Technology](https://en.wikipedia.org/wiki/Assistive_technology)，辅助技术，它是一个总括性术语包括助听器、轮椅、握笔器等。本文内主要指任何用于帮助残疾用户与网站、应用程序或其他技术进行交互的技术（比如屏幕放大镜，屏幕阅读器，文本转语音软件，语音识别软件等）

* User Agent

为用户检索和显示 Web 内容的任何软件——根据 WCAG 的定义，这包括但不限于 web 浏览器、媒体播放器和辅助技术

* Accessibility API

可访问性 API

* Accessibility tree

可访问性树

# 二、无障碍改造

## 2.1. 技术背景

### 2.1.1. 屏幕阅读器及其原理

无障碍辅助技术有很多，比如屏幕放大镜，屏幕阅读器，文本转语音软件，语音识别软件等，其中最常使用的便是屏幕阅读器（screen reader）。屏幕阅读器通过大声播放屏幕上展示的内容，帮助盲人或弱视群体使用应用。

现在有很多屏幕阅读器，例如:

* MAC

[VoiceOver](https://help.apple.com/voiceover/mac/10.14/)

* Windows:

[NVDA](https://www.nvaccess.org/)：NVDA是一个免费开源的读屏软件

JAWS

* Chrome插件

[ChromeVox](https://chrome.google.com/webstore/detail/chromevox/kgejglhpjiefppelpmljglcjbhoiplfn)：一款浏览器下使用的屏幕阅读器

[根据相关机构统计](https://webaim.org/projects/screenreadersurvey7/)，截止到 2017 年 10 月，约有 46%的用户使用 JAWS，32%用户使用 NVDA，12%的用户使用 VoiceOver。 **屏幕阅读器也是无障碍测试的主要工具。**

辅助技术通过 **可访问性树** 与页面进行交互，构建完整、健康的 **可访问性树** 是提高网站可访问的关键。 **可访问性树** 与 **DOM 树** 有什么关联呢？

其实，**可访问性树** 与 **DOM 树** 有着”如胶似漆“的关系，它们的响应变化顺序如下：

1. 浏览器使用服务器上的 HTML 标记构建 DOM 树
2. 浏览器使用 DOM 树 构建 可访问性树
3. 辅助技术与 可访问性树 进行交互，来完成用户的指令
4. 当用户产生动态内容交互时，DOM 树会发生变化
5. 浏览器响应由这些 DOM 树 更改引起的事件，并相应地更新可访问性树
6. 辅助技术发现可访问性树已更改的相关事件触发时，将这些更改信息传达给用户

因此如果页面语义不清晰时，浏览器构建出来的可访问性树就存在问题，使得屏幕阅读器这类的辅助技术无法有效进行访问，也就无法准确有效的读出用户界面上的内容。

理想情况下，仅用原生的 HTML 就可以写出让屏幕阅读器可理解的语义化内容，但对于现代复杂的 web 应用来说，仅有原生的 HTML 无法满足需求，WAI-ARIA 应运而生，它是 W3C 编写的一套规范，定义了一组可用于其他元素的 HTML 特性，用于提供额外的语义化以及改善缺乏的可访问性。

WAI-ARIA 将语义划分为角色、状态和属性三部分，通过为没有语义的标签添加 role 、aria-\* 等属性，来向可访问性树解释当前元素是什么、在做什么、有怎样的状态变化，使得 Web 内容更容易被屏幕阅读器完整的访问。

### 2.1.2.可访问富互联网应用（WAI-ARIA）基础

#### 2.1.2.1. 什么是可访问富互联网应用（WAI-ARIA）

[ARIA](https://www.w3.org/TR/wai-aria-1.1/)是"Accessible Rich Internet Applications"的缩写。它是 W3C 的 Web 无障碍推进组织(Web Accessibility Initiative / WAI)在 2014 年 3 月 20 日发布的可访问富互联网应用实现指南。它定义了一组可用于其他元素的 HTML 特性，用于提供额外的语义化以及改善缺乏的可访问性。

[简单的总结一下](https://www.w3cplus.com/wai-aria/wai-aria.html)：

* ARIA 是 W3C 的一个独立规范，帮助 Web 应用程序和 Web 页面变得更具可访问性
* ARIA 是对 HTML 语义化的补充。它具备比现有的 HTML 语义化标签和属性更完善的表达能力，并让你页面中元素的关系和含义更明确
* ARIA 规范为浏览器和解析 HTML 文档的辅助性技术提供了一种可以让人们以多种方式访问和使用 Web 的标准方法

#### 2.1.2.2.可访问富互联网应用（WAI-ARIA）的组成与使用规范

1. **ARIA的组成**

|  |
| --- |
| <div role="button" aria-pressed="false">I'm a button</div> |

从上面的例子中我们可以看到，ARIA 主要由两部分组成：

* role

角色，用来表示元素的含义

* aria-\*

标记元素的属性、状态

**role**

关于role，我们需要知道它的取值不是随意的，而且大部分是为了补充HTML的语义，这些role的值可参阅下表，也可以[点击此处](http://whatsock.com/training/matrices/)获得完整的role分类与介绍：

|  |  |  |  |
| --- | --- | --- | --- |
| alert | alertdialog | application | directory |
| document | feed | grid | gridcell |
| group | log | marquee | menu |
| menubar | menuitemcheckbox | menuitemradio | none |
| note | presentation | scrollbar | search |
| status | switch | tab | tablist |
| tabpanel | timer | toolbar | tooltip |

**aria-\***

aria-\*往往是搭配role进行使用的。为了更方便读者理解如何进行搭配，我们总结了两种对aria-\*进行分类的方式：

* 按照全局属性和私有属性分类

我们可以发现，有的aria-\*可以被运用在所有的主语言元素上，即使此时role没有被使用。我们将这些aira-\*被称作global states and properties，即全局状态或属性。

实际上，看似role没有被使用，其实它们是被应用在一个名为roletype的role上，roletype属于base role，能被所有的role继承。这些全局状态或属性的aria-\*的取值可参见下表，也可以[点击此处](https://w3c.github.io/aria/#global_states)获取详细信息：

|  |  |  |
| --- | --- | --- |
| aria-atomic | aria-busy (state) | aria-controls |
| aria-current (state) | aria-describedby | aria-details |
| aria-disabled (state) | aria-dropeffect | aria-errormessage |
| aria-flowto | aria-grabbed (state) | aria-haspopup |
| aria-hidden (state) | aria-invalid (state) | aria-keyshortcuts |
| aria-label | aria-labelledby | aria-live |
| aria-owns | aria-relevant | aria-roledescription |

除全局属性之外的其他属性，就不能随心所欲的混合搭配，比如aria-valuemax就不能用在role=button上，这需要遵守一定的规范，这些规范可以[在官网上](https://www.w3.org/TR/html-aria/#allowed-aria-roles-states-and-properties)找到官方说明，也可以[点击此处](http://whatsock.com/training/matrices/)获取比官网更直观易懂的aira role关系矩阵说明。

* 按照required和supported分类

我们可以将aria-\*分为required properties与supported properties。

比如对于role=button，它有两个supported properties：aria-expanded与aria-pressed，我们可以选择添加或者不添加这两个属性。

比如对于role=checkbox，它有一个required properties：aria-checked，这个与supported properties不同，这个属性是必须添加的。

关于这些搭配的具体方式，我们可以[在官网上](https://www.w3.org/TR/html-aria/#allowed-aria-roles-states-and-properties)找到官方说明，也可以[点击此处](http://whatsock.com/training/matrices/)获取比官网更直观易懂的aira role关系矩阵说明。

**ARIA组成小结**

在上面的内容中我们介绍了role和aria-\*，也提及了一些参考资源。在今后的使用中，有两张表格是每位开发者可以经常查阅的：

* [原生HTML标签与role的对照表](https://www.w3.org/TR/html-aria/#docconformance)
* [role与aria-\*必填属性的对照表](http://whatsock.com/training/matrices/)

**另外并不是说在HTML中使用了ARIA，Web页面就可以提高可访问性、实现无障碍，如果没有用好，反而会把你带到另一个坑中，使你的页面可访问性更差。因此在使用ARIA时，我们需要仔细理解ARIA的使用规范**。

1. **ARIA的使用规范**

**1. role必须使用有效的值**

开发者需要确保所有的role的取值都是有效的，有效的role取值可参见上文内容，比如：

|  |
| --- |
| <div role="button">I'm a button</div> // 正确使用案例  <div role="bigbutton">I'm a big button</div> // 错误使用案例，"bigbutton"不是一个 ARIA role |

**2. role的特定嵌套关系**

开发者在使用role的过程中需要注意，一些role必须被包含在特定的父元素中，一些role必须包含特定的子元素。比如：

|  |
| --- |
| // 正确使用案例，role=listitem 包含在 role=list 元素中  <div role="list">  <span role="listitem">Rainbow Trout</span>  <span role="listitem">Brook Trout</span>  <span role="listitem">Lake Trout</span>  </div>  // 错误使用案例，role=listitem 未包含在 role=list 元素中  <div>  <span role="listitem">Rainbow Trout</span>  <span role="listitem">Brook Trout</span>  <span role="listitem">Lake Trout</span>  </div> |

**3. role的使用不能改变原生语义**

比如开发者想构建一个选项卡，可以按如下方式构建：

|  |
| --- |
| <div role=tab><h2>heading tab</h2></div> |

但是不能按照如下方式构建:

|  |
| --- |
| <h2 role=tab>heading tab</h2> // 错误案例，改变了原生语义 |

另外需要注意的是，ie9/10/11并没有支持全部的html5标签，比如main，它在ie9/10/11中会被视作div处理。因此开发者在使用main时，为了支持在ie9/10/11中的无障碍，则还需要添加role=main。

**4. role的使用不能产生冗余**

比如下面添加了role是无用的，会带来信息的冗余：

|  |
| --- |
| <button role="button">press me</button> |

**5. role=application需谨慎使用**

简而言之，真正会使用role=application的场景是非常罕见的，除非需要模拟真实的桌面应用程序，因为当使用role=application时，屏幕阅读器会停止拦截击键，并将所有击键直接传递给浏览器，这会使得用户无法轻松浏览页面，更具体的介绍请[参考此处](https://www.w3.org/TR/using-aria/#using-application)。

**6. role="presentation"或aria-hidden="true"不能用在可聚焦元素上**

role=presentation或其同义词role=none，表示在它所在的元素中删除语义，在可聚焦元素上使用role="presentation"或aria-hidden="true"会导致该元素失去焦点，比如：

|  |
| --- |
| <button role=presentation>press me</button> // 错误使用案例  <button aria-hidden="true">press me</button> // 错误使用案例 |

**7. role=heading的使用**

我们使用role=heading来标识标题。应用role="heading"的元素会使AT（如屏幕阅读器）将其视为标题。

|  |
| --- |
| // 实现简单标题  <div role="heading">Global News items</div>  <p>a list of global news<p>  // 实现其他标题级别  <div role="heading" aria-level="7">Global News level 7/div>  <p>a list of global news</p> |

原生的HTML只能实现级别6的标题，role="heading"和aria-level属性配合可以实现级别其他级别的标题。

**8. role=region的使用**

当标准文档标记角色的语义无法对应待标记页面的某部分时，可使用通用region角色：role=region。这些区域可以通过使用aria-labelledby、aria-label或其他方式命名，这样可以帮助更好的建立起页面上内容信息之间的联系。

|  |
| --- |
| <div role="region" aria-label="weather portlet">  Today's weather is very good for outing!  </div> |

**9. 使用role的landmark标识页面区域**

landmark用来表示区域的类型，部分内容如下，详情请点击[这里](https://www.w3.org/TR/WCAG20-TECHS/aria.html#ARIA11)：

|  |  |
| --- | --- |
| **role** | **说明** |
| banner | 页面主标题或内部标题的区域 |
| complementary | 文档中支持主要内容的任何部分且独立 |
| contentinfo | 信息区域，例如版权和隐私声明的链接 |
| form | 表单区域 |
| main | 主要内容区域 |
| navigation | 导航区域 |
| search | web文档搜索工具 |
| application | 声明为Web应用程序的区域 |

|  |
| --- |
| <form role="search">  <label for="s6">search</label><input id="s6" type="text" size="20">  ...  </form> |

在上面例子中，使用role="search"来标识为搜索表单。

**10. aria-\*必须符合有效的名称**

开发者不能使用一个不存在或者错误拼写的aria-\*，比如：

|  |
| --- |
| aria-hidden="true" // 正确使用案例  aria-hide="true" // 错误使用案例，不存在该属性 |

为了确保使用的是存在且拼写正确的属性名称，开发者可以参考以下资料：

* [The ARIA Role Matrices](http://whatsock.com/training/matrices/)

推荐资料，建议保存链接以备日后频繁查询

* [Taxonomy of WAI-ARIA States and Properties](https://www.w3.org/TR/wai-aria-1.1/#state_prop_taxonomy)

分类介绍了widget attributes, live region attributes, drag-and-drop attributes, relationship attributes

* [Definitions of States and Properties all aria-\* attributes](https://www.w3.org/TR/wai-aria-1.1/#state_prop_def)

WAI-ARIA状态和属性的字母顺序列表

* [State and Property Attributes](https://www.w3.org/TR/wai-aria-1.1/#host_general_attrs)
* [State and Property Attribute Processing](https://www.w3.org/TR/wai-aria-1.1/#state_property_processing)
* [WAI-ARIA Attributes Module](https://www.w3.org/TR/wai-aria-1.1/#xhtml_mod)
* [WAI-ARIA Attributes XML Schema Module](https://www.w3.org/TR/wai-aria-1.1/#xhtml_schema_mod)

**11. aria-\*必须使用有效的值**

开发者在使用aria-\*时，需要对其赋予有效的值，比如：

|  |
| --- |
| aria-hidden="true" // 正确使用案例  aria-hidden="yes" // 错误使用案例，aria-hidden的可选值中无"yes"选择 |

为了确保使用的是允许的值，开发者可以参考以下资料：

* [W3C WAI-ARIA 1.1 Supported States and Properties](https://www.w3.org/TR/wai-aria-1.1/#state_prop_values)

提供了ARIA状态和属性的允许取值

**12. aria-\*的required properties必须提供**

开发者需要检查role是否含有必须提供的aria-\*。

我们可以在这张[表格](http://whatsock.com/training/matrices/)（建议保存这张表格以备日后频繁查询）中找到这些必须提供特定aria-\*的role的详细说明，比如对于role=heading，它有一个必须要求提供的属性aria-level：

|  |
| --- |
| <div role="heading" aria-level="1">I'm a heading</div> // 正确使用案例  <div role="heading" >I'm a heading</div> // 错误使用案例，未提供必需的aria-level属性 |

**13. aria-labelledby属性的使用**

* 使用aria-labelledby进行链接

使用aria-labelledby属性，开发者可以使用页面上的可见文本元素作为可聚焦元素（表单控件或链接）的标签。

|  |
| --- |
| <h2 id="headline">Storms hit east coast</h2>  <p>Torrential rain and gale force winds have struck the east coast, causing flooding in many coastal towns.  <a id="p1" href="news.html" aria-labelledby="headline">Read more...</a></p> |

屏幕阅读器在阅读上述例子时表现如下："Read more... Storm hit east coast"，这对于通过链接来浏览页面的屏幕阅读器使用者非常有用，它提供了关于链接更多的信息。

* 使用aria-labelledby连接多个文本节点的标签

|  |
| --- |
| <form>  <p>  <span id="timeout-label" tabindex="-1"><label for="timeout-duration">Extend time-out to</label></span>  <input type="text" size="3" id="timeout-duration" value="20"  aria-labelledby="timeout-label timeout-duration timeout-unit" />  <span id="timeout-unit" tabindex="-1"> minutes</span>  </p>  </form> |

在上面例子中，input标签通过使用aria-labelledby来引用文本标签。当聚焦到input时，屏幕阅读器会读取“Extend time-out to 20 minutes”。

* 使用aria-labelledby为非文本内容提供文本替代

aria-labelledby属性通过使用与标注元素的ID属性匹配的ID引用值，将元素与页面上其他位置可见的文本相关联。

|  |
| --- |
| <div role="img" aria-labelledby="star\_num">  <img src="fullstar.png" alt=""/>  <img src="fullstar.png" alt=""/>  <img src="fullstar.png" alt=""/>  <img src="fullstar.png" alt=""/>  <img src="emptystar.png" alt=""/>  </div>  <div id="star\_num">4 of 5 stars are full</div> |

* 使用aria-labelledby为用户界面控件提供名称

|  |
| --- |
| <input name="searchtxt" type="text" aria-labelledby="searchbtn">  <input name="searchbtn" id="searchbtn" type="submit" value="Search"> |

在上面例子中，在没有使用label标签的情况下,通过aria-labelledby提供对标签描述。

* aria-labelledby属性应引用存在于DOM中的元素

开发者在给aria-labelledby属性赋值时，应注意与DOM元素中的id值对应，比如：

|  |
| --- |
| <div id="test\_one">it's a test</div>  <input type="text" aria-labelledby="test\_one"></input> // 正确使用案例  <input type="text" aria-labelledby="test\_two"></input> // 错误使用案例，文档中没有对应的id="test\_two" |

**14. aria-label属性的使用**

* 使用aria-label为区域提供标签描述

|  |
| --- |
| <div role="region" aria-label="weather portlet">  Today's weather is very good for outing!  </div> |

* 使用aria-label表明链接目的

|  |
| --- |
| <h4>Neighborhood News</h4>  <p>  Seminole tax hike: Seminole city managers are proposing a 75% increase in  property taxes for the coming fiscal year.  <a href="taxhike.html" aria-label="Read more about Seminole tax hike">[Read more...]</a>  </p> |

aria-label文本将覆盖链接中提供的文本，即屏幕阅读器将使用aria-label中提供的文本"Read more about Seminole tax hike"而不是链接文本"[Read more...]"。

* 使用aria-label提供不可见标签

|  |
| --- |
| <div id="box">  This is a pop-up box.  <button aria-label="Close" onclick="document.getElementById('box').style.display='none';" class="close-button">X</button>  </div> |

在上面的例子中，我们为弹出框中的包含'X'的按钮提供了屏幕阅读器可访问的名称，同时这个名称在页面中不可见。

**15. aria-describedby属性的使用**

* 使用aria-describedby为用户界面控件提供描述性标签

|  |
| --- |
| <p>  <span id="fontDesc">Select the font faces and sizes to be used on this page</span>  <button id="fontB" onclick="doAction('Fonts');" aria-describedby="fontDesc">Fonts</button>  </p>  <p>  <span id="colorDesc">Select the colors to be used on this page</span>  <button id="colorB" onclick="doAction('Colors');" aria-describedby="colorDesc">Colors</button>  </p> |

在上面的例子中，我们使用了aria-describedby属性提供有关按钮的更多详细信息。

* 使用aria-describedby提供图像描述

|  |
| --- |
| <img src="ladymacbeth.jpg" alt="Lady MacBeth" aria-describedby="p1">  <p id="p1">This painting dates back to 1730 and is oil on canvas. It was created by  Jean-Guy Millome, and represents ...</p> |

在上面的例子中，我们使用了aria-describedby来给图像提供了长描述。

**16. aria-required属性的使用**

aria-required属性表示输入框聚焦时，提示用户是否为必填项，属性的值可以是true或false。

|  |
| --- |
| <form action="#" method="post" id="login" onsubmit="return errorCheck()">  <p>Note: [\*]denotes mandatory field</p>  <p>  <label for="usrname">Login name: </label>  <input type="text" name="usrname" id="usrname" aria-required="true"/>[\*]  </p>  <p>  <label for="pwd">Password</label>  <input type="password" name="pwd" id="pwd" size="12" aria-required="true" />[\*]  </p>  <p>  <input type="submit" value="Login" id="next\_btn" name="next\_btn"/>  </p>  </form> |

在上面的例子中，用户名和密码的输入框被设置为aria-required="true"，因此是必须输入的值。

**17. 使用aria-invalid属性表明输入框数据是否有效**

aria-invalid属性false代表验证通过，true代表验证不通过，例如

|  |
| --- |
| // 表明验证的数据有效  <input type="text" size="3" aria-invalid="false" />  // 表明验证的数据无效  <input type="text" size="3" aria-invalid="true" /> |

**18. aria-alertdialog属性的使用**

开发者可以使用role="alertdialog"创建通知。

此通知应为模态，具有以下特征：

* aria-label或aria-labelledby属性为alertdialog提供了一个可访问的名称。
* aria-describedby提供对警报文本的引用。

|  |
| --- |
| <div role="alertdialog" aria-labelledby="alertHeading" aria-describedby="alertText">  <h1 id="alertHeading">Error</h1>  <div id="alertText">Employee's Birth Date is after their hire date. Please verify the birth date and hire date.</div>  <button>Save and Continue</button>  <button>Return to page and correct error</button>  </div> |

**19. ARIA中使用的id必须唯一**

重复的id是非常常见的一个错误，分配相同的id值会导致辅助技术忽略第二个实例，破坏内容的可访问性。

|  |
| --- |
| // 错误案例，使用了重复的id，应该修改其中的一个id值  <div role="button" id="hibutton">click me</div>  <div role="button" id="hibutton">click me again</div> |

#### 2.1.2.3.可访问富互联网应用（WAI-ARIA）总结

需要注意的是，ARIA 规范仍然是一个工作草案，并不是所有的浏览器都实现这些属性（[目前只有Edge 浏览器做到了满分](https://www.html5accessibility.com/)），可能需要有多个属性来处理各种浏览器，并且实现上可能不一致。

## 2.2 改造实施

### 2.2.1. 改造综述

在我们进行无障碍改造时，我们推荐按照以下步骤进行：

改造使用支持无障碍组件的部分，凭借无障碍组件自身的无障碍支持，可实现无障碍组件绝大部分的无障碍；

对照开发 Checklist，进行进一步的无障碍改造；

页面改造，结合具体案例说明继续改造，完成大部分的无障碍改造；

最后通过自动化检测工具如，查漏补缺。

具体步骤内容可参考我们的文档：

步骤一：使用无障碍组件

步骤二：开发 Checklist

步骤三：页面其他内容

**无障碍的改造过程，就像拿石头、小石子、沙子、水来填满水瓶，层层递进**。

### 2.2.2. 步骤一：使用无障碍组件

#### 2.2.2.1. 无障碍组件简介

通过使用支持无障碍的组件构建Web，可以大幅减轻开发者在开发Web时为支持无障碍所需要的精力。支持无障碍的组件包括[BootstrapVue](https://bootstrap-vue.js.org/)、[Fusion Next](https://github.com/alibaba-fusion/next)等等，开发者可以根据自己的喜好进行选取。

#### 2.2.2.2 数据展示类组件

#### 2.2.2.3 即时反馈类组件

#### 2.2.2.4 菜单与导航类组件

#### 2.2.2.5 表单类组件

使用Fusion组件进行无障碍开发

Fusion 是一种旨在提升设计与开发之间 UI 构建效率的工作方式。我们致力于对无障碍的支持，其中 Fusion 基础组件已提供无障碍支持，开发者可以进行测试和使用，并提供Fusion无障碍使用和说明文档。所有非组件 API 属性都可以透传至 DOM 元素(我们可以传递参数，修改aria和role属性)。

经过上一章节的学习，对无障碍有初步的认识，如何进行改造呢，下面给出一些指引：

对于组件，我们为开发者内置role和特定aria-属性，开发者也可以对非组件 API 属性都可以透传至 DOM 元素，进行修改 role 和 aria-参数，但是要注意对应关系，请参考

对一些特殊的组件传递参数才能支持无障碍，设置id，autoFocus和传参数，如下：

* id - Balloon，Rating
* autoFocus - 弹层自动聚焦，例如Dialog，Overlay，Dropdown
* 传参数 - 有些组件需要根据具体的业务，实现不同的可访问性，这里为开发者内置一些参数，想使用无障碍的时候，用户只需要根据现有的需求，选择对应的内置参数，例如设置 aria-label，以下组件需要用户传参数才支持无障碍组件如下：NumberPicker、Transfer

我们为开发者提供无障碍的使用文档，请参考组件 API 中ARIA and KeyBoard。

接下来介绍具体如何对组件改造，让开发者更好的使用组件可访问性，分如下四类：

1. **数据展示类**

* Icon
  + 图标类元素
  + 若为装饰性 icon，请设置<Icon aria-hidden> 或 <Icon role="presentation">
  + 若有真实语义，请设置aria-label以描述 icon 行为<Icon aria-label="xxx">
* Badge
  + 如需朗读含义，可通过 css 设置仅读屏软件可读的内容 参考文档
* Slider
  + 内容如需朗读，请自定义 alt 属性（若为 img）或 aria-label 属性 参考文档
  + Table
  + 默认为纯数据展示的表格 - 若作为布局使用复杂功能，需自定义 role 等属性，参考文档
* Tag
  + 删除类标签，删除按钮默认朗读为“删除”，可自定义文案，[自定义方式参考国际化语言包的设置](https://fusion.design/component/doc/102)
* Paragraph、Progress、Timeline、Collapse

1. **即时反馈类**

* Message
  + 默认 role="alert"，当出现在 document 中时，屏幕阅读器会优先朗读它的内容。
  + 若为非典型使用，例如一直展示在页面上，请根据实际需求覆盖默认 role，<Message role="" />
* Balloon
  + 传入 id 以支持无障碍（为支持 SSR，所以不默认生成 id）
  + 默认 role="tooltip"，若有高优先级，请根据实际需求覆盖默认 role
  + 简单文案描述推荐使用 Balloon.Tooltip，focus 后会自动展开
  + 请务必设置可聚焦的 trigger 元素
* Dialog
  + 默认 role="dialog"，若有高优先级，请根据实际需求覆盖默认 role
  + 可通过autoFocus开启默认聚焦，焦点为弹层内容上第一个可聚焦元素 <Dialog autoFocus />
  + 请务必设置可聚焦的 trigger 元素
* Overlay
  + 可通过autoFocus开启默认聚焦，焦点为弹层内容上第一个可聚焦元素 <Overlay autoFocus />
  + 请务必设置可聚焦的 trigger 元素

1. **菜单与导航类**

* 菜单
  + Menu
    - 支持键盘导航
    - 默认 role="menu"，子元素为"menuitem"，菜单自定义单选子元素为"menuitemradio"，菜单自定义多选子元素为"menuitemcheckbox"
    - 使用selectMode属性 改变组件可选状态后，默认 role="listbox"，根节点是否多选改变aria-multiselectable的值，子元素为“listitem”
    - 请根据实际需求覆盖默认 role <Menu role="listbox" /> <Menu.Item role="listitem" />
  + Nav
    - 默认 role="listbox"，子元素为"listitem"
    - 请根据实际需求覆盖默认 role <Nav role="menu" /> <Nav.Item role="menuitem" />
  + Dropdown
    - 请务必设置可聚焦的 trigger 元素
    - 弹层展开后会自动聚焦，可通过设置 autoFocus={false}禁用
* 导航
  + Tab
    - 默认 role="tablist"，子元素为"tab"
    - 支持键盘导航
  + Pagination
    - 朗读格式默认为“上一页，当前${i}页”，可自定义文案，自定义方式参考国际化语言包的设置
  + Step、Breadcrumb
    - 默认aria-current标记当前步骤

1. **表单类**

* 推荐： Form Field Input CheckBox Radio Switch Select NumberPicker Button Range Rating DatePicker TimePicker Upload
* 不推荐：TreeSelect CascaderSelect Transfer
* Input
  + 自动根据禁用状态设置aria-disabled
  + 请使用者根据具体场景，设置aria-live、aria-label 参考文档
* CheckBox 基本无需使用者改动
  + 支持键盘导航
  + 根据选中状态更新aria-checked
* Radio 基本无需使用者改动
  + 支持键盘导航
  + 根据选中状态更新aria-checked
* Switch
  + 默认 role="switch"
  + 根据选中状态更新aria-checked
* Button SplitButton MenuButton
  + 默认使用<button>标签来渲染，可自定义标签类型<Button component="a">
  + 可通过配置设置自动聚焦，MenuButton与SplitButton组件设置menuProps属性，其中参数为对象{autoFocus: true}
* NumberPicker
  + 支持键盘上下按键，增加、减少数字。
  + 开发者可通过传递upBtnProps downBtnProps参数，自定义增加减少按钮的aria-label文案，参考文档
* Range
  + 使用左右方向键去改变数值
  + 默认 role=“slider”，滑动更新以下值aira-valuenow(定义当前值)，aria-valuetext(可读的替代文本)，aria-valuemax(最大值)，aria-valuemin(最小值)
* Rating
  + 传入 id 支持无障碍
  + 支持键盘操作
  + 通过 readAs 函数自定义展示/朗读的文案
* DatePicker
  + 使用 role 为 grid，rowgroup，row 和 cell，并设置aria-disabled(是否为禁选择状态)与aria-selected(是否被选择)。
  + 建议给予充分的aria-label提示，鼓励用户按照指定格式手动输入日期
* TimePicker
  + 建议给予充分的aria-label提示，鼓励用户按照指定格式手动输入日期
  + 使用 role 为 listbox 和 option，设置 aria-selected，tabIndex
* Upload
  + 设置 role 为 upload 和在 input 中设置 aria-hidden，不让屏幕阅读器读取
  + 建议至少保留一种点击上传的途径
* Transfer
  + 使用 role 为 menu，menuitemcheckbox 和 aria-checked，tabIndex 属性
  + 可自定义文案，自定义方式参考国际化语言包的设置。
  + 无障碍设计中，请尽可能地避免这一组件的使用
* TreeSelect CascaderSelect Cascader
  + 均支持方向键操作，上下按键为同级内容切换，左右按键为父子级切换.
  + 无障碍设计中，请尽可能地避免这些组件的使用

### 2.2.3. 步骤二：开发Checklist

对上节的知识了解后，我们为开发者提供一个改造的清单，我们所罗列的都是无障碍不可避免的问题，按照清单修改，页面基本支持无障碍，请开发者对如下规则进行阅读：

#### 2.2.3.1. 常规

[ ] 设置 lang 让屏幕阅读器选择语言去读取，lang 参数值遵循[ISO\_639-1 规则](https://zh.wikipedia.org/wiki/ISO_639-1)。

|  |
| --- |
| <html lang="en"></html>  或  <html xml:lang="en"></html> |

[ ] 标签属性的id需要全页面唯一

[ ] 页面上的文本，要按照书写习惯添加标点符号。屏幕阅读器在读取文本标点符号适时出现停顿

[ ] 当组件或元素聚焦时，要出现有聚焦的边框标志(比如按钮聚焦时，有蓝色边框)，使得与不聚焦状态做区别

[ ] 所有按钮文本描述都是可读取的，icon 类按钮要有aria-label或aria-labelledby来辅助描述

[ ] 视频需要以引入字幕文件的方式添加字幕，如下：

|  |
| --- |
| <video controls>  <source src="movie.mp4" type="video/mp4" />  <track  label="English Captions"  kind="captions"  srclang="eN"  src="captions.vtt"  default  />  <track  label="Subtitulos en español"  kind="captions"  srclang="es"  src="subs.vtt"  />  </video> |

[ ] 对于一些文本属性，屏幕阅读器读取属性优先级如下(只读取优先级最高的)：aria-labelledby，aria-label，alt，title

对于同一个元素 aria-labelledby，aria-label 只能使用一个(在同时使用时，用户代理在生成可访问的名称属性时将为 aria-labelledby 分配更高的优先级)。

[ ] 非语义标签的键盘事件捕获，要使用onkeydown ，IE浏览器不会为非字母数字键触发按键事件

[ ] 代码的逻辑顺序与显示的顺序要保持一致

[ ] 需要能跳过页面重复出现的菜单，请参考下文“跳过多个页面重复出现的导航栏”

[ ] 关于元素的隐藏，请参考详情

#### 2.2.3.2. 语义化标签

[ ] 务必使用语义化标签，并且body的直接子元素必须是以下几类标签，其次以下语义化标签与role为等价关系。对于更多语义化标签请查阅

|  |  |
| --- | --- |
| **语义化标签** | **role** |
| header | banner |
| nav | navigation |
| main | main |
| article | article |
| aside | complementary |
| footer | contentinfo |

[ ] 每个页面，有且仅有一个<main>标签或 role="main"的元素，并且为页面的主题内容

[ ] 列表结构尽量使用 ul、ol、dl 结构组织内容

[ ] 不使用语义化标签时，合理使用了aria-\*与role来描述元素，请参考

[ ] 要正确使用如下标签：

|  |
| --- |
| **b 粗体文本**  **em 着重文本**  **i 斜体字**  **strong 加重语气**  **ins 删除文本**  **del 下划线文本** |

[ ] 标题要使用 h1~h6，不能出现 h 标签跳跃，遵循从 h1 到 h6 顺序

#### 2.2.3.3. 图片

[ ] 图片要设置 alt 属性。

\* 对于有意义的图片，要设置 alt 并且描述文字简洁，不冗余，例如 logo：

|  |
| --- |
| <img src="logo.jpg" alt="this is logo" /> |

\* 对于无意义的图片，要设置 alt 为空，例如一些图标，装饰图片：

|  |
| --- |
| <img src="icon.jpg" alt="" /> |

[ ] 不要使用图片标题，使用文本标题。

#### 2.2.3.4. 表单

[ ] 表单元素的 label 要遵循约定关系[WCAG 用户输入目的](https://w3c.github.io/WCAG21-zh/index.html#input-purposes)的列表 遵循这些规范有助于浏览器自动回填数据、屏幕阅读器更好地传达信息

[ ] 表单元素要遵循如下规则：

使用 Tab 键可以顺序的切换表单元素。

对于表单中必填项要进行强调，设置aria-required属性，并通过视觉设计上的高亮提示用户。

|  |
| --- |
| // 对用户名设置 必填项  <label for="usrname"><span style="color:red">\*</span> username: </label>  <input type="text" name="usrname" id="usrname" aria-required="true"/> |

对于 label 描述表单控件，例如日期，提供具体的格式，方便用户输入。

|  |
| --- |
| // 为用户提供日期格式  <label for="date">日期（日-月-年）</label>  <input type="text" name="date" id="date" /> |

#### 2.2.3.5. 颜色和对比度

[ ] 不要依赖单一元素。例如：在展示图片、表格时，用户要加入纹理或图案。保证色盲用户能辨别，不用担心颜色会影响他对数据的理解。

### 2.2.4. 步骤三：页面其他内容

#### 2.2.4.1. 跳过页面重复出现的导航栏

每当页面载入时，读屏软件会从页面最顶部进行阅读，顶部往往是一些导航栏，这些导航栏在大多数页面中都是重复出现的。花费多余的时间阅读这些重复的内容会大大降低使用体验，因此我们需要设计一种方法，使得我们可以选择跳过这些重复的导航栏。

可以在https://webaim.org/ 、 https://github.com/ 等网站进行体验，网站首页加载完成后，按下 Tab 键，可以看到屏幕的左上角出现了：Skip to Main Content的按钮提示，通过回车或空格按下按钮，我们可以直接到达页面的主体内容区域位置，跳过导航栏。

根据页面的设计，我们分为两种情况进行无障碍的改进：

* 正常页面
* 单页面应用

1. **正常页面（即 anchor 可用的页面）**

这种情况下，我们可以用 HTML+Css 的方式，简单实现该功能。

* 原生标签<a href="#targetId"> 直接定位到页面的指定区域
* 为<a>添加 Css 样式，只在特定情况下显示跳转提示

我们在 HTML 中加入一个简单的超链接：

但是我们并不希望这个提示始终显示在页面的左上角，因此我们通过 CSS 对其进行隐藏：

|  |
| --- |
| <style>  .skip {  position: absolute;  top: -1000px;  left: -1000px;  height: 1px;  width: 1px;  text-align: left;  overflow: hidden;  }  a.skip:active,  a.skip:focus,  a.skip:hover {  left: 0;  top: 0;  width: auto;  height: auto;  overflow: visible;  }  </style>  <a href="#content" class="skip">Skip to content</a>  <div id="content">...</div>  当我们第一次按下 Tab 键时，这个超链接它会获得焦点，:focus 状态生效，因此出现在了页面的左上角。在包含正文内容信息的 div 中，我们会对其标注上 id="content" 。 |

1. **单页应用**

单页面应用（SinglePage Web Application，SPA）是一种从 Web 服务器加载的富客户端，单页面跳转仅刷新局部资源 ，公共资源(js、css 等)仅需加载一次，因此页面片段间的切换快，用户体验良好。但是由于单页面的 url 模式不同于多页面的 url 模式，对'domain.com/#content'进行了拦截，上述采用<a>标签锚点的方式的跳转部分会失效，我们可以通过 Js 的方式来解决。

对于单页面应用的解决方案，可以参考[我们的支持 fusion 主题配置的无障碍模板](https://fusion.design/template/62)，具体代码可见此[链接](https://github.com/alibaba-fusion/materials/blob/master/scaffolds/next-single-page-a11y/src/components/skip-to/index.jsx)。

|  |
| --- |
| <a  tabIndex="1"  className="skip-to-main"  onKeyDown={e => {  if ([KEYCODE.ENTER, KEYCODE.SPACE].indexOf(e.keyCode) > -1) {  const item = document.getElementById(dist || 'main');  const savedTabIndex = item.getAttribute('tabindex');  // this is important  item.setAttribute('tabindex', '-1');  item.focus();  item.setAttribute('tabindex', savedTabIndex);  }  }}  >  skip to main content  </a> |

只有以下元素是天然可以被 focus 的，其他元素例如<div>，只有在设置 tabindex 的情况下(哪怕为负数)才可以被 focus

* <a> elements with an href attribute
* <link> elements with an href attribute
* <button> elements
* <input> elements that are not type="hidden"
* <select> elements
* <textarea> elements

|  |
| --- |
| <a href="#main">jump to main</a>  <div>main<div>  <footer tabindex="-1">main<footer>  document.querySelector("a").focus(); // worked  document.querySelector("div").focus(); // not work  document.querySelector("footer").focus(); // worked |

#### 2.2.4.2. 元素的隐藏

1. **元素的隐藏**

我们可以将元素的隐藏分为以下两类：

* 对正常用户不可见
* 对读屏软件不可见

为什么会有这两种不同的隐藏目的呢？

因为有的元素，我们希望视觉正常的用户能看见，比如说广告，但是这部分信息对于使用读屏软件的视障用户而言，会降低他们的浏览体验，因此我们需要设置这些元素对读屏软件不可见。

而有的元素，我们希望读屏软件能看的见，而对于正常用户是隐藏的，比如说badge信息，正常视觉用户可以非常直观的从页面上的badge图标提取信息，比如说有多少未看的信息，但是对于视觉障碍用户而言，我们需要额外增加一个元素来描述badge，使得他们在使用读屏软件时，能从这个元素获取badge的这部分信息，此时虽然这个元素为视障用户提供了必要的信息，但是却给正常用户带来冗余的信息。因此需要对正常用户设置隐藏。

综上，在考虑元素的隐藏问题时，需要考虑两个方面：是否对正常用户隐藏？是否对读屏软件隐藏？接下来，我们将从这两个方面介绍一些隐藏元素的方法与他们之间的差异。

1. **隐藏元素的方法及其差异**

下表罗列了我们将要介绍的几种隐藏元素的方法及其差异。

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **方法** | **对正常用户可见** | **对读屏软件可见** |
| 1 | 文本缩进 | 否 | 是 |
| 2 | 绝对定位+高度塌陷 | 否 | 是 |
| 3 | 绝对定位+远离屏幕 | 否 | 是 |
| 4 | hidden属性 | 否 | 否 |
| 5 | aria-hidden="true" | 是 | 否 |
| 6 | display:none | 否 | 否 |
| 7 | visibility:hidden | 否 | 否 |

下面详细介绍一下表中罗列的几种隐藏元素的实现方法以及一些需要注意的事项。

**Text Indent 文本缩进**

|  |
| --- |
| .element-invisible {  text-indent: -9999em;  outline: 0;  } |

这种方法无法用在RTL(Right to Left)语言中。

**Position Absolute and Collapsed 绝对定位+高度塌陷**

|  |
| --- |
| .element-invisible {  height: 0;  overflow: hidden;  position: absolute;  } |

Position Absolute and Offscreen 绝对定位+远离屏幕

|  |
| --- |
| .element-invisible {  position: absolute;  top: -999999em;  left: auto;  width: 1px;  height: 1px;  overflow:hidden;  } |

**hidden属性**

|  |
| --- |
| <p hidden> hello a11y </p> // you can't see this, screen reader can't see this  <p> hello alibaba </p> // you can see this |

在使用hidden属性时，所有用户都看不见当前的元素。

**aria-hidden="true"**

|  |
| --- |
| <p aria-hidden="true"> hello a11y </p> // you can see this, screen reader can't see this |

aria-hidden对我们的无障碍页面优化很重要的一个特性，对于我们不想给展示给盲人的用户可以通过这个属性aria-hidden=“true”来隐藏，对盲人来讲相当于此元素不会被VoiceOver读出。

可以用来优化广告信息，或者对于盲人无用的信息。

**display:none**

|  |
| --- |
| .element-invisible {  display: none;  } |

使用display:none可以隐藏元素，这个元素同时也不能被读屏器访问，且隐藏后不占据空间。

**visibility:hidden**

|  |
| --- |
| .element-invisible {  visibility: hidden;  } |

使用visibility: hidden也可以隐藏元素，这个元素同时也不能被读屏器访问，但是与display: none不同的是，前者隐藏后占据空间。另外display: none和visibility: hidden可以对屏幕外的内容进行焦点管理，这部分内容的详细介绍在焦点管理一文中详细介绍。

1. **小结**

在上面罗列的7种隐藏元素的方法中，我们应该如何进行选择呢？其实非常简单，开发者可以直接根据是否对正常用户可见以及是否对读屏软件可见进行判断，从而在第2节开始的表格中直接选出适用的方法。

#### 2.2.4.3. 异步内容的更新及提醒

使用JavaScript，可以动态地更改页面的某些部分，而不需要重新加载整个页面，这些更改对于视觉正常的用户通常是可视的，但是对于使用读屏软件的用户来说，这一视觉上的变化并不能直接通过读屏软件传达。

考虑到无需重新加载页面即可更新的动态内容，通常是一个区域，为了能让这些内容的更新被使用读屏软件的用户知晓，我们将这些区域标记为live region。在实际应用中，live region一般用作消息提醒，比如打开Twitter，右键页面审查元素，你会发现：

|  |
| --- |
| <div id="sr-event-log" class="visuallyhidden" aria-live="polite">  <p>有新的推文，按 . 以查看它们。</p>  </div> |

虽然在网页中你看不到这段内容（因为class="visuallyhidden"，具体说明请见2.5.2 元素的隐藏），但是读屏软件（screen reader）可以在新Tweet出现时，朗读”有新的推文，按 . 以查看它们。“

1. **live region区块**

**aria-live属性**

ARIA提供了aria-live这个属性来设定在网页更新时，该如何通知读屏软件使用者，下面是aria-live属性三种可以选用的设定：

|  |  |
| --- | --- |
| **aria-live属性值** | **使用场景** |
| off | 默认值，表明一个区域不是实时的，不会报读变化 |
| polite | 更新内容应当在适当时刻报读，比如在用户停止输入时 |
| assertive | 立即向用户报读更新内容。由于这是突兀的，assertive值应当仅用于更新内容是重要的，应立即通知给用户的情况 |

我们推荐使用aria-live=polite，正如我们在上面的Twitter的例子中看到的那样，他们也使用了aria-live=polite的选项。

aria-live=assertive常用在一些错误提示的场景，有时也用于即时反馈信息，比如在[阿联酋航空的购票官网](https://www.emirates.com/cn/chinese/)上，有如下代码：

|  |
| --- |
| <div aria-live="assertive"  class="screen-reader-text"  id="search-flight-departure">  2 找到的结果  </div> |

这部分HTML代码对应的是购票官网上的出发地搜索框的提示信息，在aria-live=assertive模式下，使用者在输入地名进行搜索时，读屏器能即时给出反馈，朗读此时有几个找到的结果。

另外需要开发者注意的是，屏幕阅读器在阅读多条信息更新时，会根据不同的aria-live设定值做出不同的反馈，比如下面的字数统计场景：

|  |
| --- |
| <p aria-atomic="true" aria-live="polite">  There is <span id="num">0</span> characters.  <p> |

当在输入框输了三个字符abc后，aria-live=polite会给出如下提示：

|  |
| --- |
| abc  There is 3 characters. |

而aria-live=assertive则会给出如下提示：

|  |
| --- |
| a  There is 1 characters.  b  There is 2 characters.  c  There is 3 characters. |

**aria-atomic属性**

aria-atomic是live region的非必要属性，它的取值可以是true或者false。

当aria-atomic=true时，live region更新后，辅助技术将把整个live region作为整体进行通报；若aria-atomic=false时，辅助技术只通报live region中更改的部分，比如我们可以在Safari浏览器中测试：

|  |
| --- |
| <p aria-atomic="true" aria-live="polite">  Today is <span id="day">Monday</span>  <p>  // 若 Monday 变成了 Friday，  // aria-atomic=“true” 朗读为“ Today is Friday ”  // aria-atomic=“false” 朗读为“ Friday ” |

**aria-busy属性**

aria-busy也是live region的非必要属性，它的取值可以是true或者false。aria-busy=true时，区块的更新不会马上通知使用者，当aria-busy被置为false时，这些更新才会被通知给使用者。比如以下场景，ul有多个地方需要修改：

|  |
| --- |
| <ul aria-atomic="true"  aria-busy="true"  aria-live="polite"> |

我们可以先将aria-busy设为true, 等到全部内容更新完毕后再设成false，这样可以避免辅助工具在区域内容更新完毕前不断提醒使用者。

**aria-relevant屬性**

aria-relevant也是live region的非必要属性，用来表示要对live region的哪些变化做出反应，默认情况下aria-relevant="text additions"，它的取值可以是以下属性的集合，采用空格分割：

|  |  |
| --- | --- |
| **aria-relevant属性值** | **需要做出反应的变化** |
| additions | live region区块内有新增的DOM节点 |
| removals | live region区块内有减少的DOM节点 |
| text | live region区块中有文字变化 |
| all | 上面的所有变化 |

比如在下面的例子中，只有在live region区块中增加节点时，辅助技术才会告诉使用者，如果是移除节点或文字内容的变化，就不会告诉使用者：

|  |
| --- |
| <ul aria-relevant="additions"  aria-atomic="true"  aria-live="polite"> |

1. **实践例子**

在这一节我们将结合Fusion Design的组件来说明如何实现异步内容更新的提醒，点击此处浏览完整代码与页面demo。

|  |
| --- |
| <Form>  <FormItem label="输入内容:">  <Input.TextArea  placeholder="分享生活点滴"  name="contents"  id="contents"  aria-describedby="inputInfo"  aria-required="true"  onChange={this.sizeCount}  />  </FormItem>  <FormItem label=" ">  <span id="inputInfo" aria-live="polite">  已输入字符:  </span>  </FormItem>  </Form> |

通过设置aria-live，我们能使得用户输入完一段文字并停止输入后，读屏软件能自动读出下方的提示语句，提示用户当前已经输入了多少字符。

更多关于内容更新提醒的实践例子可以[点击此处](http://pauljadam.com/demos/aria-atomic-relevant.html)进行尝试。

#### 2.2.4.4. 焦点管理

1. **焦点简介**

网页设计中常说的焦点可以理解为当前操作的对象。打个比方：某超链接获得了焦点，则回车后，浏览器会打开这个超链；再比如输入框获得焦点，我们即可在这输入框中输入文字，如果失去了焦点，就不能在这个输入框输入文字了。

一些用户（比如使用读屏软件的用户）几乎全靠键盘或其他输入设备来操作计算机。对这些用户而言，焦点至关重要，因为这是他们到达所有屏幕元素的主要途径。因此在一些无障碍页面检查的checklist中会指出所有页面功能应该都能使用键盘来执行，除非是手绘图这种无法使用键盘执行的操作。

焦点管理做的比较好的网站，能让用户在tab键的帮助下，完全依靠键盘进行页面的浏览操作，比如像阿联酋航空的订票官网。

使元素获得焦点

在默认情况，只有如下元素可以通过tab键获得焦点：

* 有href属性的a
* button
* input
* select
* textarea

其他元素例如div、span等是无法通过tab键获得焦点的，那么怎么样才能获得焦点呢？我们可以为标签加入tabindex属性来使这些元素获得焦点：

|  |
| --- |
| <div tabindex="0">这是能获得焦点的div</div> |

tabindex属性的取值范围是0到32767，如果赋值为负数则表示没有焦点标记，通过下面的表格我们可以具体了解tabindex属性不同取值时的功能：

|  |  |
| --- | --- |
| **tabindex属性** | **是否可通过Tab键获取焦点** |
| 负数（比如tabindex="-1") | 不可以 |
| 0（tabindex="0"） | 可以，获取顺序对应于文档中的位置 |
| 正数（比如tabindex="2"） | 可以，数值越小越先获取 |

|  |
| --- |
| <span tabindex="0"> hello </span> // 鼠标点击与键盘控制均能获得焦点  <span> world ! </span> // 鼠标点击和键盘控制都不能获得焦点  <span tabindex="-1"> fusion design </span> // 鼠标点击可以获取焦点，但是键盘控制不能获取焦点  <input id="age" type="text" tabindex="0" /> // 鼠标点击与键盘控制均能获得焦点  <input id="name" type="text" /> // 鼠标点击与键盘控制均能获得焦点，因为input是隐式可聚焦元素  <input id="name2" type="text" tabindex="-1" /> // 鼠标点击可以获取焦点，但是键盘控制不能获取焦点 |

1. **焦点管理**

让我们先了解一些在焦点管理中，常用的对焦点的操作方法：

* 获得焦点元素
* 加载时自动获得焦点
* 焦点判断

**获得焦点元素**

我们可以通过document.activeElement记录下触发焦点的元素。知晓页面的焦点位置是十分有用的，在后面处理失焦时我们可以用到这一方法。

**加载时自动获得焦点**

autofocus是一个boolean属性，默认值为true，规定当页面加载时设置了该属性为true的元素自动获得焦点。

<button type="button" autofocus>按钮文本</button>

例如谷歌的首页，人们99%的时间都是用它来搜索，我们可以对搜索框的input元素设置autofocus属性，在页面加载时，光标就可以自动定位在输入框里，带来更好的用户体验。

**焦点判断**

我们可以通过document.hasFocus()判断用户与页面是否处在交互状态，比如在一些在线笔试考试中，可以使用这个方法，在考生切换页面时弹出提示，告诉考生不要浏览其他页面。

1. **失焦处理方案**

在单面应用中，我们会碰到页面跳转后失焦的问题，经过上面对焦点和焦点控制方法的介绍，我们有如下的方案来解决页面跳转后的失焦问题。

**通过焦点返回方法处理失焦**

焦点返回是个常用的操作，比如弹出一个对话框，关闭这个对话框后，我们需要焦点返回到之前触发的dom元素上，这样才能有一个良好的体验。这个方案的实现关键就是使用上面提到的document.activeElement：

|  |
| --- |
| // 获得当前的焦点位置  var lastfocus = document.activeElement;  // 打开对话框，关闭后聚焦到打开前的焦点  new dialog();  dialog.show();  dialog.hide(function(){  lastfocus.focus();  }); |

**通过autofocus属性处理失焦**

比如说我们在页面跳转后，总是希望能让焦点处于我们新页面的搜索框上，这个时候我们可以设置我们的搜索框的autofocus属性为true即可。

|  |
| --- |
| <input type="search" autofocus="true" placeholder="enter your question" > |

**添加快捷键处理失焦**

当页面跳转后，用户不知道焦点处于什么位置时，我们可以提供键盘快捷键的方式让用户快速聚焦到某一特定的元素上，这可以通过accesskey属性来实现。主菜单与导航菜单使用accesskey，通常是不错的选择。

|  |
| --- |
| <a href="http://webaim.org/" accesskey="w">WebAIM.org</a> |

可通过ctrl+alt+w 或 alt+w 或 alt+shift+w来完成键盘快捷键聚焦的过程，以下元素支持accesskey属性：a, area, button, input, label以及textarea。

#### 2.2.4.5. 设计建议

本文章主要指出一些我们应当避免的设计，并总结一些设计建议。

1. **应当慎重设计既可hover又可click的元素**

同一个元素可被hover、可被click，但触发的效果不同，会对使用读屏软件的用户带来麻烦。

比如对于登录按钮，当我的鼠标悬浮在登录按钮上时，可以出来登录菜单栏，当我们点击登录按钮时，可以进入个人中心，这样的设计对于视觉正常的用户而言，页面更加简洁明了。

但是对于需要使用读屏器的视觉障碍用户而言，体验感就不是那么好了，因为他们只能通过键盘进行操作，而键盘是无法模拟鼠标的hover这样的一个形态，因此上述页面对于使用读屏器的视觉障碍用户而言，就无法进入视觉用户可以通过hover而见到的登录菜单栏了，这样就直接影响了使用读屏器的这些用户的体验。

1. **尽量使用文本而不是文本图片**

当我们在浏览网页的时候，偶尔会遇到图片无法加载的情况，此时比较好的现象是，在未加载出来的图片处出现了替代的文本，让我们知道这张原本应该出现的图片的含义，这可以通过在img中添加alt属性完成。在我们进行网页无障碍开发过程中，也需要对图片添加alt属性的描述，这样我们的视觉障碍用户在使用读屏软件时，也能了解到页面上图片所代表的内容。

而考虑到读屏软件用户的体验，在填写图片的alt属性时，我们需要遵循以下几点规则：

* 精确的传达出图片的内容和功能。
* 尽可能的简洁。
* 不要提供重复信息。
* 不要使用一些诸如“…的图片”这样的冗余语句。
* 不要在alt中添加换行符。
* 对于无意义的图片，alt=""。
* WCAG规定不允许使用文字图片，若是英文图片，转化为相应的英文文字，对于图片描述，转化为相应文字。

因此图片+alt的方式只适用于少量文本，大量的文字内容还是需要放在文本的位置处。

## 2.3 开发工具

### 2.3.1 组件库推荐

#### 2.3.1.1 Fusion Next

#### 2.3.1.2 Bootstrap

说下做的是什么，给个链接，各自说下优点

#### 2.3.1.3 Material

### 2.3.2 辅助开发工具

开发测试由软件开发人员或工程师在软件开发生命周期的构建阶段执行。我们从代码静态分析、浏览器和视觉测试，对可访问性进行测试，降低软件的缺陷，因此从不同的角度进行测试：

#### 2.3.2.1. 静态分析工具

静态分析是指在不运行程序的条件下，对程序进行分析的方法，eslint-plugin-jsx-a11y是一款静态分析工具，在不运行程序条件下，运行此工具可以对程序进行静态分析：

 对JSX代码进行无障碍标准测试。

 eslint-plugin-jsx-a11y是静态扫描，无需侵入代码，能扫到事件缺失，其中定义30多条比较常见规则，请参考。

 可以在开发和集成测试中使用。

react-axe以及Access Lint也是不错的静态分析工具。

#### 2.3.2.2 浏览器检测工具

开发者在开发的过程中，可以通过浏览器插件对已经编写的代码进行可访问性测试，检测是否符合无障碍标准。Axe Chrome Plugin是其中的一款浏览器检测工具，介绍如下：

在浏览器中测试代码是否符合无障碍标准。

与eslint-plugin-jsx-a11y工具相比，此插件的无障碍检测规则更多，请参考

Axe可访问性检查器是一个快速、轻量级的可访问性测试工具，它基于业内最流行的开源无障碍测试库axe-core，安装成功后可在Chrome Dev Tools面板找到。也可以使用axe-core进行单元测试，请参考。

在控制台选择axe，就可以对网站进行测试，并列出不符合无障碍问题并给出解决方案。

#### 2.3.2.3 视觉测试工具

在全球范围内，存在很多视觉障碍的用户，比如红绿色盲，在设计和开发的过程中需要考虑到他们的使用体验，因此这里为开发者提供I want to see like the colour blind，介绍如下：

 在浏览器运行，对视觉进行测试。

 与前面两个工具相比，此工具只能在视觉方面进行测试。需要开发者根据选择不同类型的视觉用户进行测试。

 Chrome插件，开启可模拟红、绿、蓝、全色盲、色弱等用户眼中的站点，直接发现站点中存在的可辨别性问题。

# 三、无障碍检验及测试

### 3.1 测试样本选择

#### 3.1.1 结构化样本

选择一个视图和视图状态的样本，其中包括：

1）所有常见视图和视图状态：识别网站的通用网页，例如起始视图、用户登录界面等；

2）所有其他相关视图和视图状态；

3）覆盖网站或应用程序的基本功能。

#### 3.1.2 随机选择的样本

随机选择的视图和视图状态的数量至少占结构化样品的10%，且样本不属于之前结构化样本的一部分。随机选择的样本应来源于身体机能差异的评测人员随机访问并记录的视图和视图状态。

随机选取的视图和视图状态样本作为一个指标，用于验证结构化样本。

#### 3.1.3. 完整的过程

所选样本应包含该网站或应用程序一系列完整过程的所有视图和视图状态。

对于每个完整过程中的视图和视图状态，应将结构化样本和随机样本合并至完整过程的样本集，记录访问的视图序列和视图状态，标记其状态与默认顺序。

### 3.2 多层次测试检验

在开发完成以后，我们需要整体对项目进行无障碍测试，通过工具和人工检测，发现其中的问题，进行修复。这里按照测试的准确程度层层递进，可分为以下三个层次，业务可根据实际场景进行选择：

* 自动化检查
* 借助辅助技术进行自测
* 邀请真实用户进行测试

#### 3.2.1. 自动化检验

自动化检查可以检查出一些可访问性问题，如标题的结构和顺序、alt文本是否存在、页面标题和语言设置是否存在、颜色对比度是否合理、一定程度上的HTML结构是否语义化等。由于开发阶段和自动化检测工具的相似性，这里推荐按照工具的以下使用顺序进行检查，先对代码进行静态检测，然后通过axe进行测试，其次对于视觉测试，需要开发者根据不同类型的视觉用户进行测试：

* eslint-plugin-jsx-a11y
* Axe Chrome Plugin
* I want to see like the colour blind

**Benchmark**

在我们通过工具和人工测试后，怎么衡量我们测试是否达到一定的要求，这个时候我们需要寻找一些工具对我们的测试进行评估或生成生产测试报告，来对我们的测试给出有效的评估。[Lighthouse](https://developers.google.com/web/tools/lighthouse/)是Google开发的网站性能评测工具，主要用于改进网络应用的质量，已开源。内置在Chrome Dev Tools 中，也有插件版本。

无障碍的检测是该工具评测的一个子项，可以借助这个插件生成站点的无障碍评分报告。

#### 3.2.2. 借助辅助技术进行自测

手动测试将有助于识别准确性问题，例如标题结构是否与页面真实内容顺序相符、alt文本是否准确地传递了相关图像的信息、弹出的窗口、对话框或其他通知是否与展示一致并且明显等。

残疾人借助辅助技术（Assistive Technology）来操作软件产品，常用辅助技术有： 屏幕阅读器 - 用于读出屏幕上显示的文本 屏幕放大软件 - 用于放大显示器，使视力受损的用户轻松阅读。

**屏幕阅读器**

对于屏幕阅读器介绍，请参考，这里主要介绍通过屏幕阅读器如何测试，我们依据有障碍人群，浏览器和操作系统的不同使用，需要进行不同的测试，进行如下分类：

* 人群类型
  + 对于失明用户，我们在测试过程中需要开启屏幕阅读器和通过键盘去测试，对于视力障碍，我们推荐使用I want to see like the colour blind测试
* 浏览器
  + 在火狐和Chrome中我们使用axe插件进行测试(推荐)，在IE中使用Web Accessibility Toolbar测试，Safari使用Web Development Tools测试
* 操作系统
  + 推荐在widnow系统中我们可以在火狐或Chrome中使用axe插件进行测试，Apple系统下使用VoiceOver + Chrome中axe插件测试

**测试checklist**

对于网站的可访问性，我们参考如下的条目，若基本符合，网站基本支持网站的可访问性，如下：

[ ] 仅使用键盘，是否能使用站点的主要功能

使用Tab在页面的可聚焦元素上切换，使用 shift+Tab 切回上一聚焦元素。

[ ] 能否跳过导航栏，详情点击这里

跳过导航栏对有运动障碍的人很有意义

[ ] 是否支持内容缩放，详情点击这里

缩放后是否正常显示，这对于有视觉障碍的人(比如老人)很有意义

[ ] 是否通过屏幕阅读器，准确获取页面所传达的信息

针对完全失明用户

[ ] 开启色盲弱模拟或对比度插件，能否使用站点的主要功能

#### 3.2.3. 邀请真实用户进行测试

真实的有访问障碍的用户是无障碍的实际使用者，他们有不同的习惯和期望。有条件的话，可以邀请真实的存在访问障碍用户定期检查站点，以确保主要内容和功能对他们可用。