# 1. 简介

该规范的目的包括：

* 扩展可能由作者提供的无障碍信息
* 要求支持宿主语言提供全键盘支持，可以实现与设备无关的方法，比如，使用电话，手持设备，电子书阅读器和电视；
* 改善由脚本生成的动态内容的可访问性；并且
* 提供可以使用辅助工具交互的方式。

WAI-ARAI是一项技术规范，它提供了一个用来改善网站内容和应用的可访问性与交互能力的框架。本文档是主要面向开发者创建的自定义控件和其他的web应用组件。参见《WAI-ARIA概述》中针对其他受众的相关文档链接，比如WAI-ARIA Rrimer中介绍的开发者想要解决的无障碍问题，基本概念以及WAI-ARIA的技术方法。

这个草案目前处理两个方面的角色：用户界面方法和结构关系。更多的信息和用户案例查看《WAI-ARIA Primer》，它介绍了使用这角色使可交互的内容无障碍化。

角色分类的设计支持了一部分在平台无障碍接口中常用的角色。在动态生成的网站内容中引用这些分类中的角色可以用来支持辅助技术进行交互。

这种模式支持的标准被设计成是可扩展的，基于基本角色可创建自定义的角色。允许用户代理去支持最少的基本角色，并且用户代理可以为自定义的角色提供增强的访问支持。在《XML Schema [XSD]》中大多数都是正式的。但是，XSD中不会定义类似的角色，比如《baseConpcets》和更多的描述性定义。

* 《WAI-ARIA Primer》，一个W3C工作组记录，介绍了开发者在解决问题中的无障碍问题，基本概念和WAI-ARIA技术方案。
* 《WAI-ARIA Authoring Practices》， 一个计划性的W3C工作机记录，描述如何让网站开发者使用WAI-ARIA可以开发无障碍的富因特网应用。它提供了详细的建议和例子，主要面向网站应用开发人员，对用户代理和辅助技术的开发人员也适用。
* 《WAI-ARAI User Agent Implementation Guide》，一个计划性的W3C工作组记录，描述如何使浏览器和其他的用户代理支持WAi-ARIA；特别的描述了如何公开WAI-ARIA的特性给到平台的无障碍接口。
* 《WAI-ARIA路标》，一个计划性的W3C工作组记录，定义了使富网站内容无障碍的路径，包括已经采取的措施，未来剩余的步骤和时间点。

## 1.1富因特网应用可访问性

网站无障碍的领域定义了如何使网站内容让残障人士可以使用。某些残障类型使用辅助技术与内容进行交互。辅助技术可以将展示的内容转化为更适合用户的格式,并且可以允许用户以不同的方式进行交互。例如,用户通过方向键可能需要,或选择,与滑动控件交互,而不是使用鼠标拖拽。为了有效地做到这一点,软件需要理解内容的的语义。语义是*科学的以意义*;在这个例子中,用于指定的角色,和属性,应用在用户界面和内容元素中，像人类所能理解的那样。例如,如果一个段落语义识别,辅助技术可以将作为内容中的一个单元与之交互，知道这个段落确切的边界。一个可调范围滑块或可折叠的列表(又名tree控件)都是更复杂的例子,它的各个部分部件的语义需要让辅助技术正确识别支持并与其有效的交互。

新技术常常忽略无障碍方面需要的语义，一个新的开发实践经常滥用这些技术的语义。元素在语言中拥有一个定义的含义经常被一个不同的意义所替代去让用户理解。

比如，网站应用开发者利用HTML，CSS和javascript事件创建了可折叠的属性菜单，但是html并没有语义化的tree元素。对于没有残障的用户，在没有恰当语义的情况下，可以浏览和使用类似折叠属性菜单的控件，但树形菜单控件对于残障人士并没有可感知性或可交互性，因为辅助技术不能识别出角色

WAI-ARIA为作者提供了一种方式可以为自定义的控件提供适当的语义从而使这些控件对于残障人士是无障碍的，可用的以及可交互的。*本规范标识的各种控件和结构是通常被公认的无障碍产品*，通过提供一个可以添加给内容的相应角色本体论。*这些允许带有角色的元素可理解想一个特定的控件或者结构类型，从宿主语言中继承的无论任何种类的语义。*角色是平台无障碍接口的一个普通属性，使用辅助技术为用户提供有效的展示和交互。

这个角色的分类法包括交互控件和元素表示的文档结构。角色分类法描述了每一个角色支持的可继承的和详细的属性。《[WAI-ARIA User Agent Implementation Guide](http://www.w3.org/TR/2014/REC-wai-aria-implementation-20140320/) [[ARIA-IMPLEMENTATION](file:///C:\Users\rockywen\Desktop\references#ref_ARIA-IMPLEMENTATION)]》提供了关于角色映射无障碍接口的信息。

角色是元素类型且不会因为时间和用户动作改变。角色信息被辅助技术使用通过和用户代理交互，为指定的元素类型提供正常的处理。

状态和属性用来声明元素效果和描述交互的重要属性。当属性给客户端脚本动态的更改时，他们能够让用户代理和操作系统去适当的处理元素。例如，替代输入和输出技术，比如屏幕阅读软件和语音听写软件，需要能够使识别并且有效的操作以及传递各种交互状态（比如，disabled禁用、checked选中）给用户。

虽然他可能让辅助技术直接从DOM中访问这些属性，但是优先的机制是让用户代理映射状态和属性到操作系统的无障碍接口上。细节可参看《[WAI-ARIA User Agent Implementation Guide](http://www.w3.org/TR/2014/REC-wai-aria-implementation-20140320/) [[ARIA-IMPLEMENTATION](file:///C:\Users\rockywen\Desktop\references#ref_ARIA-IMPLEMENTATION)]》

图1.0阐述的是用户代理和无障碍接口以及辅助技术之间的关系。它描述由用户代理提供 “*contract*”给辅助技术，包括为我们很多的无障碍平台的图形用户界面提供无障碍接口中特定的无障碍信息方法（role角色、state状态、selection选择？、[event](file:///C:\\Users\\rockywen\\Desktop\\terms" \l "def_event)notification事件提醒、relationship information关系信息和描述）。DOM，通常是HTML，作为数据模型和视图在一个指定的模型-试图-控制器（MVC）关系中。Javascript作为控制器操控显示数据内容的样式和内容。用户代理传递相关信息给操作系统的无障碍接口，用于供任何辅助技术使用，就像屏幕阅读软件。

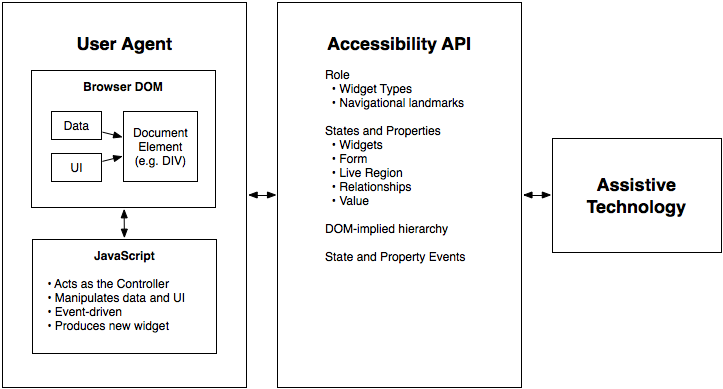


图1.0：contract模型无障碍接口

更多信息参看《[WAI-ARIA Primer](http://www.w3.org/TR/2010/WD-wai-aria-primer-20100916/) [[ARIA-PRIMER](file:///C:\Users\rockywen\Desktop\references#ref_ARIA-PRIMER)]》中的使用角色让交互内容无障碍。

除了*prose*以外， Web本体论语言（OWL）中提供了角色分类法，资源描述框架（RDF）中也有描述。工具可以使用这些在一个给定的内容文档中验证角色的实现。比如，一些角色的示例是期望作为一个特定父角色的子元素。此外，一些角色可能支持一个特定的状态或者属性，而另外的角色可能不支持。

*提示:使用RDF或者OWL作为一个正式的具有代表性的角色可能被使用作为支持未来的扩展。标准化的RDF/OWL机制可以用来定义新的角色，继承来自这个标准的角色。这个机制在一个彼此协作的方式中定义和使用角色的扩展。但是，并没有在该规范中定义。WAI-ARIA未来的某个版本会定义如何扩展角色。*

替代输入设备的用户需要键盘无障碍地访问内容。新语义，当把《WAI-ARIA开发实践》中提供的推荐的键盘交互结合之后，应该允许替代输入措施促进命令和控制，通过一个替代输入方案。

WAI-ARIA描述了导航路标通过分类法和XHTML角色路标，它可以帮助在灵活性和视觉有障碍的人士使用改良的键盘提供的导航。WAI-ARIA也可以给有认知学习障碍的人士使用。附加的语义允许作者重组和替代他们所需要的替代内容。

辅助技术需要有能力去支持替代输入来获取和设置控件状态和属性当前的值。辅助技术同样需要去确定对象是被选中以及管理允许多种选择的控件，就像列表盒子和网格。

基于语言的命令和控制系统可以获得来自WAI-ARIA的好处，比如角色属性辅助传递声音信息给用户。举个例子，通过确定一个元素拥有一个菜单角色并且他包含了3个带有menuitem元素，每一个元素包含文本内容表示不同的爱好，一个语音系统可以以这样的状态面对用户，“请在以下三种中选择一个：巧克力、草莓或者香草。”

WAI-ARIA的目的是用来为本地语言语义做一个补充，而不是替代。当宿主语言提供一个特性，这个特性提供和WAI-ARIA特性等价的可访问性，那就就使用宿主语言。WAI-ARIA应该只用于宿主语言缺乏所需的角色、状态和属性指示器这种状况。使用一个宿主语言尽可能和WAI-ARIA特性相似的特性，然后添加WAI-ARIA完成含义。例如一个可多选的网格需要实现的像一个表格，然后WAI-ARIA用来标识它是一个可交互的网格，而不仅仅是一个静态数据表格。这样就允许在不支持WAI-ARIA并且保存宿主语言语义的完整性的情况下给用户代理最好的回馈。

## 1.2目标受众

该规范为WAI-ARIA定义了基本的模型，包括 角色 状态 属性 和 值。它影响着几个受众：

* 用户代理 处理内容中包含的WAI-ARIA特性
* 辅助技术采用特殊的方式呈现内容给残障人士
* 创建内容的作者
* 开发工具帮助作者创建符合要求的内容，以及
* 适合的检测工具用来检测是否恰当的使用了WAI-ARIA

该规范应该适用与上述每一个受众的合理的要求

虽然这个标准适用于上面的受众，但它并不是特别的针对这些受众，也不是这些受众的唯一的信息来源。下面的文档也提供了重要的信息支撑：

* WAI-ARIA Authoring Practices记录了开发建议，以及开发者感兴趣的开发工具和适当的评测工具。
* WAI-ARIA User Agent Implementation Guide addresses用户代理和辅助技术的开发者

## 1.3用户代理支持

WAI-ARIA的特性依赖于用户代理的两个方面：

* 为了改善可访问性，主流的用户代理使用WAI-ARIA去改变宿主语言特性中暴露出的无障碍接口。《[WAI-ARIA User Agent Implementation Guide](http://www.w3.org/TR/2014/REC-wai-aria-implementation-20140320/)》中定义了该机制
* 辅助技术在一个无障碍接口里使用可用的增强的信息，或者使用WAI-ARIA标记直接通过DOM去传达语义和交互信息给用户。

除了使用WAI-ARIA标记去加强 暴漏给无障碍接口，用户代理的行为就像他们本地化一样。辅助技术对无障碍接口中额外的信息做出反应，*类似他们已经在非web内容上准备了相同的信息*。用户代理不是辅助技术，除了为无障碍接口提供适当的更新，其他什么也不需要做。

WAI-ARIA规范既不要求也不禁止用户代理基于WAI-ARIA提高本地显示和交互行为。主流的用户代理可能会暴漏WAI-ARIA的导航路标（比如一个对话框或者通过键盘命令）让所有用户更容易的导航。用户代理被鼓励最大限度的对用户提供有效性，包括残障用户。

WAI-ARIA试图提供*失去*的语义去将作者的意图传递给辅助技术。通常情况，作者使用WAI-ARIA会提供适当的展示和交互特性。过一段时间，宿主语言可能提交WAI-ARIA的等价物，比如一个新的表单控件，由用户代理实现成为标准的用户界面控件。允许作者使用他们去替换自定义的WAI-ARIA可用用户界面组件。在这种情况下，用户带来将支持原生的宿主语言特性。实施了WAI-ARIA的宿主语言开发者建议继续支持WAI-ARIA特性，当他们没有不利于与隐含的宿主语言语义冲突的时候。，如果宿主语言特性不能充分的达到作者的需求，WAI-ARIA语义会更加清晰的反映作者的意图

## 1.4 WAI-ARIA和宿主语言协同进化

WAI-ARIA意在增强像HTML和SVG这样的语言对语义化的支持，或者作为在没有明确支持ARIA在其他标记语言中的一个无障碍增强的技术。当作者通过样式和脚本创建新类型的对象，它为辅助技术明确阐述语义，不直接支持页面的语言，因为新的类型对象发明的比web语言中出现的支持的标准化对象要快。

当宿主语言提供了一个语义化元素为对象类型，它利用样式和脚本没有适当的创建对象。WAI-ARIA可以改善这些对象的可访问性，允许用户代理处理本地对象，提供最好的无障碍。例如，在HTML中使用h1元素比在div元素上使用heading角色要好得多。

随着时间的推移，预计宿主语言会逐步为当前只能使用WAI-ARIA声明的对象提供语义。这是自然并符合情理的，WAI-ARIA的目标之一是去帮助刺激更多的语义和可访问性标记诞生。当本地语义为一个给定的特性变得可用，作者应使用本地特性，并且停止对那个特性使用WAI-WRIA。剩下的内容可以继续使用WAI-ARIA，然而，用户代理仍然需要支持WAI-WRIA。

虽然随着时间的推进，WAI-ARIA的特性可能会失去重要性，WAI-ARIA一般情况下可能添加语义到网页中会被持续的需要。宿主语言可能不会实现所有WAI-ARIA提供 的语义，各种宿主语义可能会实现特性的不同子集。新类型的对象持续的被开发出来，WAI-ARIA目标之一是提供一个方法去使这样的对象无障碍化，因为网站开发实践进步的经常比与宿主语言标准要快。在这种方式中，WAI-ARIA和宿主语言双方以不同的速率一起进化。

一些宿主语言除了用户界面以外还存在为它们创建语义特性。例如，SVG在图形化对象背后表达出了语义，并没有这些对象能够表现的用户界面组件;XForm为控件提供了语义，却并没有提供广泛的用户接口特性。宿主语言像这些可能，在设计时没有提供映射到WAI-ARIA的本地语义。在这种情况下，WAI-ARIA可能作为一个长期的方法被采用，负责添加语义信息给用户界面组件。

## 1.5开发实践

### 1.5.1开发工具

很多需要被定义的WAI-ARIA角色，状态和属性可以被自动的检测，在开发进程中，类似其他使质量控制进程用来校验代码一样。用来帮助创建自定义控件的作者，*开发工具可以比较WAI-ARIA中支持的控件的角色，状态和属性，像他们支持相关的和相互对照的角色*，状态和属性一样好。开发工具可以告知作者在控件设计模式中的错误，以及可能会提示开发者不能单独通过上下文确定的信息。比如，一个脚本库可以确定树形项目在一个树形视图中的标识，但是可能需要提示作者去标识全部的树形。帮助作者呈现一个符合逻辑的无障碍结构，一个开发环境可能提供一个在线的基于WAI-ARIA标记的web资源视图。

在HTML中，tabindex是实现WAI-ARIA，支持键盘焦点导航浏览的一个重要方式；开发和调试工具可以检查确保tabindex的值是合理设置的。比如，错误的状况可能包括这些情况，树形中多于一个树形项目拥有一个大于或等于0的tabindex值，没有为任何的树形项目设置tabindex值；或者aria-activedescendant没有定义当树形角色有一个大于或者等于0的tabindex值。

### 1.5.2测试实践和工具

交互内容的可访问性不能仅通过静态校验所确定。交互内容的开发者应该为独立于设备访问控件和应用做测试，应该核实无障碍接口访问所有内容以及在用户的交互中的变化。

## 1.6辅助技术

以编程的方式可以访问无障碍语义与辅助技术是至关重要的。大多数的辅助技术与用户代理进行交互，像其他应用，通过一个公认的无障碍API。可感知对象在用户界面中暴露给辅助技术，像无障碍化对象一样，通过无障碍API接口来定义。恰当的这样去做，无障碍信息——角色、状态、属性像上下文一样，通过无障碍API准确的专递给辅助技术。当一个状态发生变化，用户代理提供一个恰当的时间通知无障碍API。上下文信息，在多个宿主语言中，像HTML，就可以从DOM它本身提供的一个上下文树形层级结构确定。

当一些辅助技术和这些无障碍APIs交互，另外一些可以直接从DOM获取信息。这些技术可以重组、简化、风格化或者回流内容去帮助不同类型的用户。常见的用例是为不同类型的适应性，可能是老龄化，认知障碍人群，在嘈杂的环境中使用他们的设备。例如，可用的区域导航路标可以允许为一个移动设备基于它的语义在任何一个时间只显示一部分内容的自适应。这可以减少用户在任何一个时间需要处理的信息量。在其他的情况中，它可能适当的替换一个自定义用户界面和其他一些东西，使得更容易使用一个键盘或者触摸屏幕设备来导航。

这些语义编程访问的要求与[User Agent Accessibility Guidelines: Programmatic operation of user agent user interface](http://www.w3.org/TR/2002/REC-UAAG10-20021217/guidelines.html" \l "tech-ui-access-api) 和 [Programmatic notification of changes](http://www.w3.org/TR/2002/REC-UAAG10-20021217/guidelines.html#tech-api-notify-change) ([[UAAG](http://www.w3.org/TR/wai-aria/references#ref_UAAG)])相似，除了适用于内容之外，也不仅仅适用于用户代理。

# 2. 使用WAI-ARIA

复杂的web应用变嘚不可访问，当辅助技术无法判断文档背后的语义，或者当用户无法用一个可用的方法有效的导航它的所有部分的时候，WAI-ARIA划分语义到role中（定义用户接口元素的类型）还有role支持的states和properties。

在一个生命周期中，作者需要将一个WAI-ARIA的role和适当的states和properties关联到文档中的元素中，除非元素的states和properties已经有个适当的隐性的WAI-ARIA语义。在这种情况下等效的宿主语言的状态和属性获得优先级以避免冲突而隐藏在宿主元素中的角色属性将优先考虑。

## 2.1. WAI-ARIA角色

利用role属性将一个WAI-ARIA角色定义在一个元素上，像《[Role Attribute](http://www.w3.org/TR/2011/WD-role-attribute-20110113/)》工作草案中的定义role属性一样。

<li role="menuitem">Open file…</li>

规范中的角色定义包含一组文档landmarks和[WAI-ARIA属性分类](http://www.w3.org/TR/wai-aria/roles" \l "role_definitions)。

角色在这个[分类](http://www.w3.org/TR/wai-aria/terms" \l "def_taxonomy)里，他们的预期行为利用 [RDF/OWL](http://www.w3.org/TR/2004/REC-owl-features-20040210/)建模。角色分类的特性为每一个角色规定了下列要求：

* 为角色提供一个信息描述
* 相关角色的分层信息（例如,一个[directory](http://www.w3.org/TR/wai-aria/roles" \l "directory)是[list](http://www.w3.org/TR/wai-aria/roles" \l "list)的一种类型）
* 角色的上下文（比如，一个[listitem](http://www.w3.org/TR/wai-aria/roles" \l "listitem)应该被包含在一个[list](http://www.w3.org/TR/wai-aria/roles" \l "list)中）
* 描述其他规范中的相关概念
* 使用OWL中规定的层次结构类型并允许语义继承（类似class的层次结构）；并且
* 为每一个role提供支持statas和properties（例如，一个[checkbox](http://www.w3.org/TR/wai-aria/roles" \l "checkbox)支持通过[aria-checked (state)](http://www.w3.org/TR/wai-aria/states_and_properties" \l "aria-checked)检查是否被check） .

附加一个关于如何处理每个元素的信息角色给辅助技术。

## 2.2. WAI-ARIA状态和属性

WAI-ARIA规定了一组states和properties用来支持各个平台上的平台accessibility APIs。辅助技术可以通过一个exposed（裸露的）用户终端dom或者通过一个映射到平台accessibility APIs来关联这些信息。当结合了roles，在任何时候用户终端会将带有用户接口的信息提供给辅助技术，然后传达给用户。States和properties的变化会通知给辅助技术，从而提示用户一个变化已经发生。

在接下来的例子中，一个list item（html中的li），用来创建了一个可以check（checkbox衍生）的菜单项，js事件捕获鼠标和键盘事件去切换[aria-checked](http://www.w3.org/TR/wai-aria/states_and_properties" \l "aria-checked)的值。一个role用来让一个简单的widget行为被用户终端理解。用户行为可改变的属性（比如aria-checked）被定义在[states and properties](http://www.w3.org/TR/wai-aria/states_and_properties)章节中。

<li role="menuitemcheckbox" aria-checked="true">Sort by Last Modified</li>

一些可访问性的states，被称作为状态管理，受用户终端的控制。比如状态管理包含键盘焦点和选择。状态管理常常有相应的css伪类（比如 :focus和:selection）去定义样式的变化。于此相反，states在这个规范中通常由作者控制被称之为不可管理状态。一些states有用户终端管理，比如[aria-posinset](http://www.w3.org/TR/wai-aria/states_and_properties" \l "aria-posinset)和[aria-setsize](http://www.w3.org/TR/wai-aria/states_and_properties" \l "aria-setsize)，但是，如果DOM是不完全的（incomplete）导致用户终端计算错误，作者可以覆盖他们。用户终端将影射managed和unmansged states两种到accessibility APIs平台。

绝大多数现代用户终端都支持css选择器，并且允许作者基于WAI-ARIA属性信息建立UI变化，尽量减少脚本的数量实现等效的功能。在下面的例子中css选择器基于aria-checked属性的值来确定一个标记check时，显示文字变粗和背景图片。

[aria-checked="true"] { font-weight: bold; }

[aria-checked="true"]:before { background-image: url(checked.gif); }

如果css不是作为check标记切换的视觉反馈，作者可以使用额外的标记和脚本来管理图片，从而表示menuitemcheckbox是否被check。

<li role="menuitemcheckbox" aria-checked="true">

<img src="checked.gif" role="presentation" alt="">

<!-- note: additional scripts required to toggle image source -->

Sort by Last Modified

</li>

## 2.3. 焦点管理

一个应用当使用的时候应该使一个元素获得焦点，像应用一样需要一个地方给用户进行输入。除非用户干预，否则建议作者不去破坏元素的焦点或者滚动它到屏幕以外。所有交互的对象应该是可以聚焦的。综合交互的所有部分都应该是可以聚焦的或者有一个文档的替代方法实现他们的功能，就像是键盘的快捷键。作者应该保持一个显而易见的可以被发现的方式，可以通过tabbing或者其他的标准导航技术。键盘用户可以将焦点移动到任何可交互的元素上。

当使用标准的HTML和基本的WAI-ARIA小部件，应用的开发者可以简单的操作tab的顺序或者使用脚本来给文档中的元素创建键盘的快捷键。使用更加复杂的小部件就需要作者去管理他们的焦点。SVG Tiny提供了一个类似与导航的*”ring”机制*，默认情况下遵循文档的顺序利用使用系统的输入设备来实施（绝大部分电脑都有tab键）。SVG作者可以操作焦点属性将元素布置在导航顺序中并且通过修改元素的导航属性，他们可以动态的制定导航的顺序。

WAI-ARIA包含大量的“容器管理”小组件，也称为“复合”小组件。在适当的时候，容器负责跟踪最后一个活跃的后裔（组件中的某个元素，比如某按钮之类，默认的情况下是这个容器中的第一个项目）。至关重要的是，一个容器要保持一个可用的和一致的策略。然而也会有例外的情况，当先前聚焦的元素重新获得焦点的时候，当前活动的后裔元素应该和这个容器最后获得焦点时候所激活的后裔元素是一样的。例外的情况是一个小组件中的内容发生了变化，比如一个menubar的小组件，当焦点离开menubar的时候，用户希望总是返回menubar里的第一个项目。举例，如果当时用户离开树形分组，而树形分组中的第二个项目是当前活动的后裔，那么当树形分组重新获得焦点的时候，它的第二个项目保持活动后裔状态。用户也可以通过点击来激活容器内部中的某一个后裔。

当容器或者他的活动后裔获得焦点，用户可以使用附件按键在容器中导航，比如，箭头键去改变当前的活动后裔。按任何主导航复加键（一般是tab键）就会移动出这个容器到下一个小组件。

举个例子，一个grid可以被看作一个带有成千上万的gridcell电子表格元素。*所有这些不可能在同一时间出现在文档中。*这就需要使用容器的aria-activedescendant属性来管理容器元素中的焦点，或者管理容器中子元素的tabindex并且设置焦点给适当的子元素。更多的信息，可以查看[Providing Keyboard Focus in WAI-ARIA Authoring Practices](http://www.w3.org/TR/2010/WD-wai-aria-practices-20100916/" \l "kbd_focus)。

内容作者需要去管理下列容器角色的中焦点：

* [combobox](file:///C:\Users\rockywen\Desktop\roles#combobox)
* [grid](file:///C:\Users\rockywen\Desktop\roles#grid)
* [listbox](file:///C:\Users\rockywen\Desktop\roles#listbox)
* [menu](file:///C:\Users\rockywen\Desktop\roles#menu)
* [menubar](file:///C:\Users\rockywen\Desktop\roles#menubar)
* [radiogroup](file:///C:\Users\rockywen\Desktop\roles#radiogroup)
* [tree](file:///C:\Users\rockywen\Desktop\roles#tree)
* [treegrid](file:///C:\Users\rockywen\Desktop\roles#treegrid)
* [tablist](file:///C:\Users\rockywen\Desktop\roles#tablist)

更多的焦点管理信息可以在[WAI-ARIA Authoring Practices](http://www.w3.org/TR/2010/WD-wai-aria-practices-20100916/)中的 [Using Tabindex to Manage Focus Among Widgets](http://www.w3.org/TR/2010/WD-wai-aria-practices-20100916/#focus_tabindex)章节中找到。

# 3. WAI-ARIA的规范性要求

*本章是规范的。*

*本规范表示这章节是否是规范性的或者是资料性的*。像适用于规范性的或者是资料性的章节一样把这个章节分类。陈述句“This section is normative”或者“This section is informative”适用于所有章节中的子章节。

规范的章节提供了一些作者、用户终端和辅助技术必须按照本规范的实施的要求。在这个文档里关键词MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT,SHOULD, RECOMMENDED, MAY, 和 OPTIONAL的解释和 [Keywords for use in RFCs to indicate requirement levels](http://www.rfc-editor.org/rfc/rfc2119.txt)一样。RFC-2119关键字要被格式化成大写字母并且包含在strong的标签里并且要带有class=”rfc2119”。当要显示上面所示的关键字，但是不同意这种格式，他们不会传达RFC 2119中的正式的信息，只是解释，即资料性的。在这个规范中尽可能的避免这种用法。

资料性的章节需要提供一些有用的信息来了解该规范。这些章节可能包含例子和推荐的练习，但不必为了符合本规范而遵循这些建议。

# 4. 主要术语

这个章节是资料性的。

这里定义的一些术语，在本文档中使用下列定义。

**Accessible API 无障碍接口**

操作系统和其他平台会提供一组接口将对象和事件的信息暴露给辅助技术。辅助信息利用这些接口去获取小组件的信息和与之交互。举个例子这些技术是 [Java Accessibility API](http://java.sun.com/javase/technologies/accessibility/index.jsp) [[JAPI](http://www.w3.org/TR/wai-aria/references#ref_JAPI)], [Microsoft Active Accessibility](http://msdn.microsoft.com/en-us/library/ms697270(VS.85).aspx) [[MSAA](http://www.w3.org/TR/wai-aria/references#ref_MSAA)], [the Mac OS X Accessibility Protocol](http://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/Accessibility/cocoaAXIntro/cocoaAXintro.html) [[AXAPI](http://www.w3.org/TR/wai-aria/references#ref_AXAPI)], the [Gnome Accessibility Toolkit (ATK)](http://library.gnome.org/devel/atk/) [[ATK](http://www.w3.org/TR/wai-aria/references#ref_ATK)], 和 [IAccessible2](http://www.linuxfoundation.org/collaborate/workgroups/accessibility/iaccessible2) [[IA2](http://www.w3.org/TR/wai-aria/references#ref_IA2)]

**Accessible Name 无障碍名称**

Accessible Name是用户接口元素的名字。每一个平台的accessibility APIs提供了Accessible Name属性。Accessible Name的值从可视化中衍生出来（比如按钮上的可视的文字），或者是用户接口元素中无形的属性（一个描述icon的替代）。

一个简单用于Accessible Name的属性可能是一个显示“OK”的按钮。文字“OK”就是Accessible Name。当按钮获得焦点的时候，辅助技术可以串联起平台的角色并带着Accessible Name描述。举个例子，屏幕阅读软件也许会读作“push-button”或者“Ok button”。串联的顺序和细节的描述（比如读作“button”，“push-button”，“clickable button”）取决于平台accessibility APIs或辅助技术。

**Assistive Technologies 辅助技术**

硬件和（或）软件

* 依赖由用户代理检索和渲染的web内容提供的服务
* 通过使用APIs和一个用户代理或者web内容本身共同工作，并且
* 提供超越用户代理所提供的服务，方便残障人士使用用户界面和web内容进行交互。

这个定义可能和使用在其他文档中的概念不同。

在该文档的上下文中重要的辅助技术如下：

* 屏幕放大镜，用于放大及改善渲染的文本和图片的视觉可读性；
* 屏幕阅读软件，最常用来通过语音合成或者一个盲文显示器传递信息；
* 文本转语音软件，用于将文本转化成合成语音；
* 语音识别软件用于允许语言控制和发出指令
* alternate input technologies (including head pointers, on-screen keyboards, single switches, and sip/puff devices), which are used to simulate the keyboard;
* 替代输入技术（包括头戴鼠标，）
* alternate pointing devices, which are used to simulate mouse pointing and clicking.

**Attribute 属性**

在这个规范中，属性的使用像它在标记语言中一样。属性是添加到元素中的结构化特性，从而提供关于元素表示的对象的states和properties的信息。

**Class 类**

一组具有类似特征的实例对象

**Element 元素**

在这个规范中，元素的使用和标记语言中的一样。元素在标记语言中是结构化的包括对象的数据档案。

**Event 事件**

一个编程信息，用来在计算机系统中一个对象的状态变化传递给另外的对象。用户在一个网页上输入，用一个抽象的事件作为媒介描述交互并提供文档对象状态改变的提示。在某些编程语言中，事件通常被称为通知。

**Hidden 隐藏**

指的是元素对任何用户是不可见和不可感知的。一个元素只在DOM中隐藏，如果它或者它的任何一个祖先元素有aria-hidden属性的话就设置被true。

*提醒：作者会注意到可以为所有css媒介类型设置visibility:hidden和display:none，因此，辅助技术都会通过DOM渲染引擎中访问到设置隐藏的内容。然而为了支持辅助技术直接访问DOM或者使用其他技术来隐藏内容（比如：透明和移除屏幕之外），作者要保证当元素显示或者隐藏的时候aria-hidden属性要相应的进行更新。*

**Informative 参考性**

内容以提供信息为目的，而不要求有一致性。符合要求的内容被称之为normative。

**Keyboard Accessible 键盘可访问性**

让用户更加容易的使用键盘或者利用辅助技术模拟键盘输入，像是一个sip and puff tube。详情参考[WCAG 2 Guideline 2.1; "Make all functionality available from a keyboard"](http://www.w3.org/TR/WCAG20/" \l "keyboard-operation)

**Landmark 路标**

用户希望快速访问的页面上的一类区域。用户目的相关的内容在页面中的不同区域。比如：导航、搜索、仔细阅读主要信息。

**Live Region 活动区域**

活动区域是网页中的可感应区域通，当用户将聚焦到某个地方的时候，常用来更新一个外部事件的结果。这些区域不总是更新用户交互的结果。这种做法已经越来越多的用户在使用ajax的地方。举个例子，活动区域是个聊天记录，股票行情或者是一个体育比分，需要定期更新反馈统计信息。由于这些移步更新的区域在用户的焦点的之外，屏幕阅读器一样的技术并不知道他的存在或者不能给用户处理他们。WAI-ARIA提供了一组peoperties从而允许作者去辨别这些活动区域并且告知如何去处理他们：aria-live，aria-relevant，aria-atomic，aria-busy。自定义的活动区域列表参见[Choosing Between Special Case Live Regions](http://www.w3.org/WAI/PF/aria-practices/#chobet)。

**Primary Content Element 主要内容元素**

一个实现宿主语言主要内容的元素，就像html中的body元素。

**Managed State 状态管理**

与accessibility APIs相关的state（状态）但是他们的值的阅读和设置是在用户代理端进行。应用的作者不能总是管理这些state，但是要让用户代理可以处理。在一些情况下应用的作者可以很好的管理这些state。一般的managed state（状态管理）包括focus（焦点）和selection（选择）。

**Normative 一致性**

符合一致性的，与之相反，内容被认定为informative或者non-normative是不符合一致性。

**Object 对象**

在用户界面的上下文中，在可感知的用户体验中的、通过一个或多个元素中的标记语言中呈现的并且由用户代理渲染的一个项目。

在程序的上下文中，一个或一个以上的类实例化的，并且定义了一般特性相似的对象。一个对象在一个无障碍接口中可能表示一个或多个DOM对象。无障碍接口以及各定义了来自DOM接口中的不同的接口。

**Ontology 本体论**

一个描述类的特征和他们之间如何关联。

**Operable 可操作的**

用户用可用的方式去控制。详细相关内容可参照[WCAG 2 Principle 2; content must be operable](http://www.w3.org/TR/WCAG20/#operable)。也可参看[Keyboard Accessible](http://www.w3.org/TR/wai-aria/terms#def_keyboard_accessible)。

**Owned Element 从属元素**

一个“owned element”是元素的任何一个DOM后裔，任何一个元素都会像通过aria-owns子元素一样被指定，或者拥有子元素的任何一个DOM后裔。

**Perceivable 可感知的**

可感知是用户获得感觉的一种方式。相关信息可查看[WCAG 2 Principle 1; content must be perceivable](http://www.w3.org/TR/WCAG20/#perceivable)。

**Property 属性**

属性是一个给定对象的基本的特性，或者是和对象的相关数值。改变属性可能会显著的影响对对象的描述和意思。某些属性（比如，aria-multiline）比state（状态）改变的要少。一些属性（比如：aria-valuenow、aria-activedescendant、aria-valuetext）可能经常会变。

**Relationship 关系**

将两个不同的可分音节的东西连接起来。关系可以支持多种类型去指定一个对象标记另外一个，控制一个等等。

**Role 角色**

主要的指示类型。这个语义关联允许工具去呈现和支持和用户所希望相一致的对象交互方式。

**Semantics 语义**

使人类理解一些事情，定义了计算机可以处理一个对象，想元素和属性并且可靠的代表对象使得各种人类相互了解对象

**State 状态**

状态是一个动态的属性表示对象的特点，可能会改变相应用户动作或自动化流程。状态不会影响对象的基本性质，但是是受到对象相关数据或用户交互的影响。

**Taxonomy 分类法**

分类是一个定义各种类的特点彼此相关的层次结构，继承父类的属性的层次结构。一个分类可以组成正式定义的Ontology。

**Understandable 可理解的**

给用户一个漂亮的方式，他们可以构造一个适当的含义。可参看[WCAG 2 Principle 3; Information and the operation of user interface must be understandable](http://www.w3.org/TR/WCAG20/#understandable)。

**User Agent 用户代理**

任何软件都会为用户检索和显示网站内容，比如 网页浏览器，媒体播放器，插件和其他程序，包括辅助技术。

**Value 值**

一个文字信息用来表达一个状态、属性或者文字内容。

**acrWidget无障碍的富控件（accessibility rich Widget？）**

用户可以进行交互的独立的用户接口对象。小组件的范围从一个一个值或操作的简单的对象（比如复选框和菜单项目）到包含很多可以管理自对象的复杂对象（比如 树形结构和电子表格）。