1Article accessibilité

Contenu

[Contenu 1](#_Toc480282309)

[1 Comment rendre une application web accessible ? 2](#_Toc480282310)

[Présentation du contexte 2](#_Toc480282311)

[Les auteurs 2](#_Toc480282312)

[Comment écrire du HTML accessible. 2](#_Toc480282313)

[La page HTML 2](#_Toc480282314)

[1 Problématique technique 3](#_Toc480282315)

[2 Mise en place dans le code. 4](#_Toc480282316)

[Les stratégies. 4](#_Toc480282317)

[Créer Un site totalement accessible. 4](#_Toc480282318)

[1 Créer un site alternatif accessible. 4](#_Toc480282319)

[Entête de la page 5](#_Toc480282320)

[1 Structurer le document 5](#_Toc480282321)

[2 Les images 6](#_Toc480282322)

[Le titre de la page 6](#_Toc480282323)

[Les menus 6](#_Toc480282324)

[Les titres 7](#_Toc480282325)

[3 Les textes et les paragraphes 7](#_Toc480282326)

[4 Cacher du texte aux utilisateur voyants 8](#_Toc480282327)

[5 Les images. 8](#_Toc480282328)

[6 Les champs de formulaires. 9](#_Toc480282329)

[7 Les éléments clickables 9](#_Toc480282330)

[8 Les tableaux 10](#_Toc480282331)

[9 Mise à jour de contenu 10](#_Toc480282332)

[3 Conclusion. 11](#_Toc480282333)

[4 Exemple 1 11](#_Toc480282334)

# Comment rendre une application web accessible ?

## Présentation du contexte

Ce document est destiné aux développeurs, web designer et webmasters, qui souhaitent rendre leurs sites ou applications web accessibles, au plus grand nombre d’utilisateurs. Même si ce document traite avant tout de l’accessibilité pour les déficients visuels, il peut s’appliquer à tout type de handicap.

## Les auteurs

**Fabrice LUNEAU**, je suis développeur, Java/J2E. Je suis titulaire d’un BTS d’informatique de gestion option développeur et d’une licence de concepteur développeur informatique. JE suis devenu non voyant à la suite d’une maladie. Je connaissais déjà là problématique de l’accessibilité de l’outil informatique, car j’intervenais bénévolement dans une association de déficient visuels, et j’ai pu échanger avec des adhérents sur cette problématique. Après avoir perdu la vue, j’ai pu constater les problèmes d’accessibilité de l’outil informatique et de la fonction de développeur pour les déficients visuels, au cours de ma reconversion professionnelle.

**V H**

Introduction

**Dans ce document nous expliquerons essentiellement comment écrire du HTML accessible. D’une manière générale il convient d’utiliser le HTML de façon sémantique et non visuelle. C’est-à-dire qu’une balise doit être utilisée pour son rôle et non pour son rendu à l’écran. Car les lecteurs d’écran s’appuie sur le rôle ou la fonction d’une balise HTML, pour retourner les informations aux utilisateurs non-voyants.** Si l’on souhaite agir sur la présentation il convient d’utiliser les propriétés CSS.

**On peut s’appuyer sur la norme ARIA(Accessible Rich Internet Application)du W3C et HTML5, qui sont des formidables avancées en matière d’accessibilité.**

**Afin de restituer l’information à l’utilisateur le lecteur d’écran, crée une copie de la page, qui s’appuie sur le code HTML, pour créer une version accessible à plat.**

## La page HTML

Lorsque l’internaute non-voyant arrive sur une page, un certain nombre d’informations lui sont annoncées :

Liste de 3 éléments

* Le titre de la page
* le nombre de cadre
* Le nombre de liens
* Le nombre de titres
* le nombre de zones, qui composent le document

Fin de liste

Ensuite le lecteur d’écran commence la lecture du document depuis le début.

En général l’utilisateur, stop la lecture et cherche à aller au contenu principal via une touche de raccourcis. En allant au premier titre de niveau un de la page, ou en cliquants sur un lien d’évitement « aller au contenu » ou « skip to main content ». Cela évite de faire une recherche hasardeuse dans la page.

Ensuite il est possible de naviguer avec la touche tabulation pour aller d’élément focusable en éléments focusable (liens, éléments de formulaire,….). Mais le fait d’appuyer sur la touche tabulation saute la lecture des textes, qui ne sont pas focusable.

Il est aussi possible de naviguer entre les différentes zones de la page, si la page utilise cette possibilité.

Il est possible d’obtenir la liste de tous les éléments classés par catégories et de demander à se déplacer à un élément par son type.

En pratique, les lecteurs d’écran permettent d’obtenir :

Liste de 3 éléments

* la liste des liens hyper texte

Grâce à elle, il est possible de parcourir la liste pour rechercher le lien désiré. C’est pourquoi les liens doivent avoir un texte évocateur.

* + un lien nommé : « cliquer ici » ne veut rien dire !
* la liste des titres par niveau. Celle-ci permet de se déplacer dans le document pour rechercher le contenu désiré. Le lecteur d’écran annonce « titre de niveau *n* » et après lit le texte. . C’est pourquoi les niveaux de titres doivent être utilisés pour leurs niveaux et non pour leur apparence.

Il est important de noté, que la mise en formes n’est pas annoncées :

* taille du texte
* couleur du texte, même un text transparent sera lu
* aspect de la police italic, gras, souligné,…
* la liste des zones de formulaire

Pour les formulaires, il est possible d’obtenir toute la liste des éléments et de naviguer directement, parmi ces derniers.

Liste de 4 éléments

* + champs
  + zone de texte
  + case à cocher
  + boutons radios
  + bouton
* les zones composant la page.

Fin de liste

Fin de liste

## Problématique technique

Les lecteurs d’écrans vocalisent tous les éléments textuels et foc usable, or un certain nombre d’éléments ne disposent pas par défaut de texte, comme les images et les champs de formulaires.

De plus le texte fournit doit avoir du sens dans le contexte de la page et hors du contexte. Car la navigation sur une page n’est pas forcément linéaire. Elle peut être interrompue et lors de la reprise l’information fournie peu être incompréhensible hors de son contexte, si l’on arrive directement sur un élément. Par exemple un lien intitulé « ici », qui est précédé du texte « pour télécharger le logiciel cliquer ici », n‘a pas de sens si l’on arrive directement dessus avec la touche tabulation Il faut réunir les deux informations.

Il ne faut aussi ne pas oublier que l’internaute non-voyant n’utilise pas de souris, et donc des éléments non foc usable au clavier seront inaccessibles.

Par exemple une image simulant un bouton de validation de formulaire , activable seulement à la souris ; ne sera pas accessible.Et sera reconnu comme une image et non comme un bouton par le lecteur d’écran

## Mise en place dans le code.

Les solutions proposé dans ce document, , requiert peu de technique. Elles reposent en grandes partie sur la mise à disposition d’untexte accessible et à sa mise en valeur par le code HTML. Et dans un deuxième temps sur l’utilisation des propriétés ARIA et en particulier « rôle « .

### Les stratégies.

Il existe deux stratégies possibles.

### Créer Un site totalement accessible.

La première approche est de créer un site totalement accessible. Cependant on peut être confronté à des solutions techniques antagoniques. Néanmoins cela n’est pas un objectif impossible à tenir.

Une première approche est d’utiliser du HTML standard, sans élément dynamiques, limiter l’utilisation du JavaScript et d’élément dessinés en CSS.

### Créer un site alternatif accessible.

La deuxième approche est de proposer des outils directement intégrés dans les pages.

* Des feuilles de styles pour les malvoyants, avec des contrastes moins agressifs ou négatifs (vert sur noir, blanc sur noir).
* Créer un site alternatif en HTML simple ou sans les éléments non-accessibles, remplacer les éléments dynamiques par des éléments HTML statiques. Remplacer les menus pliables par des listes.
* Il est possible de créer des éléments seulement accessible et visible par les lecteurs d’écrans.

### Entête de la page

Pour se comprendre le mieux est de parler la même langue. Ainsi il est important d’indiquer dans le code de la page la langue du contenu.

<html lang="fr">

Sinon à défaut il se peut, que le lecteur d’écran essaye de lire la page en français avec une voix anglaise, ce qui est pratiquement incompréhensible.

Le lecteur d’écran peut détecter automatiquement la langue et certaindialect du texte , dans la pratique cette solution n’est pas toujours fiable.

Sans la balise <body> ajouter la propriété ARIA role avec la valeur

* « Document » par défaut
* »application » si la page n’est pas destiné à être manipulé par un navigateur web ou lu par un utilisateur humain.

### Structurer le document

Si le document est composé de plusieurs parties, il faut les identifier explicitement.

* L’entête header

<header role="banner">…

</header>

* le menu ou les menus

<nav role="navigation">

…

</nav>

* Le contenu principal du document, il peut en y avoir qu'un par page.

<main role="main">

…

</main>

* Le pied de page footer

<footer role="contentinfo">

…

</footer>

\*\*\* faire une note sur la compatibilité avec les navigateur\*\*\*\*

\*\*\* la norme aria n’est pas implémentée par tous les navigateur\*\*\*

### Le titre de la page

Le titre de la page doit annoncer la fonction de la page et rappeler sur quel site on se trouve. Car si l’on change de fenêtre ou d’onglet, il doit être possible de s’appuyer sur lui, pour savoir où l’on se trouve.

Une page intitulé « Accueil », ne fournit pas assez d’informations, il faut une information complémentaire, comme le nom du site. « Accueil – de Développez.net ».

Une page intitulé « Forum sur l’accessibilité des application web », fournit assez d’informations.

Il est possible de rajouter le nom du site, à chaque fois. Mais de préférence à la fin du titre. Car dans de nombreux cas l’information est superflue. Si on sait où l’on se trouve, il est possible d’intérrompre la lecture .

Les pages de résultat de recherches paginées doivent indiquer le numéro de la page en cours et le nombre de page totale.

### Les menus

Il est possible de proposer un ou plusieurs menus , pour la navigation, la recherche,…

Il faut les identifier comme vu précédemment avec la balise «<nav> » et la propriété «  rôle « nav », et utiliser les balise <ul> et <li> pour structurer les éléments du menu.

chaque éléments du menu doit être marqué avec la propriété role="menuitem">.

<nav role=*"navigation"*>

<ul role=*"menu"*>

<li role="menuitem">Accueil</li>

…

</ul>

</nav>

### Les titres

La balise <H> est reconnue naturellement. Cependant il faut bien utiliser les titres. Le lecteur d’écran annoncera le titre par son niveau puis par son texte. C’est pourquoi les niveaux de titres doivent être utilisés pour leur sens et non pour leur aspect, comme on le ferai avec un traitement de texte. Cela permet de structurer le document et de guider la lecture.

Pour changer l’aspect d’un titre, il convient d’utiliser les propriétés CSS.

La page doit comprendre de préférence un seul titre de niveau 1 , dont le contenu doit être proche du titre de la page, c’est-à-dire du contenu de la balise <title>. Puisque on s'attend à trouver le contenu principal de la page en dessous de ce dernier.

On peut utiliser plusieurs titre de niveau 1, si on a structurer la page en plusieurs parties, mais toujours un seul titre de niveau 1 par zone.

### Les textes et les paragraphes

Le texte est naturellement accessible, il suffit de lancer la lecture ou de naviguer avec les touches de flèches. Mais le texte n’est pas focussable. Il est néanmoins possible de naviguer de paragraphe en paragraphe, c'est-à-dire de balise <p> en balise <p>, cependant il ne s’agit pas d’un mode de navigation pertinent.

Il peut être judicieux de placer le focus, à l’endroit où doit commencer la lecture, avec une <div> par exemple, ou d'utiliser un lien d'évitement « aller au contenu », « skip to main content »,.

Les listes sont aussi naturellement accessibles, elles sont annoncées par « début de liste » et « fin de liste ». Puis ensuite les éléments avec leur présentation puce ou numéros. Elles permettent de structurer un menu par exemple, comme on l’a déjà vu avec la balise <nav>.

### Cacher du texte aux utilisateur voyants

En théorie il est possible de faire du responsive design à l'attention des déficient visuels avec les propriétés CSS suivantes :

* aural CSS 2.0) / speech (CSS 2.1)Synthèses vocales
* braillePlages braille
* embossedImprimantes braille

Mais en pratique ces propriétés ne sont pas implémenté dans les navigateurs, donc inutilisables !

Pour afficher un texte seulement pour les utilisateurs de lecteurs d’écran, il est possible d'utiliser une <div> invisible, avec les propriétés CSS suivantes par exemple :

.visuallyhidden {

blindZone {    border: 0;

    clip: rect(0 0 0 0);

    height: 1px;

    margin: -1px;

    overflow: hidden;

    padding: 0;

    position: absolute;

    width: 1px;

}

Cette solution peut poser des problèmes de référencement, mais est plus cohérente qu’afficher un texte en blanc sur blanc. Une solution de ce type es utilisé par Google sur la page de résultat, un text « si vous utiliser un lecteur d’écran cliqué ici … », qui n’est pas visible par les autres utilisateurs.

### Les images.

Les images sont focussable au clavier. Quand le lecteur d’écran arrive sur une image, s’il ne dispose pas d’information, il lit le nom du fichier de l’image, ce qui n’est pas d’une grande aide et très perturbant, surtout quand il s’agit d’un numéro. Il faut Proposer au minimum une alternativ avec la propriété alt et éventuellement une description avec la propriété description. Il est possible de proposer un lien vers une description avec la propriété « longdesc ».

C’est pourquoi il est impératif de bien choisir le texte en fonction du contexte de la page.

<img alt= » » description= » »>

### Les champs de formulaires.

Les textes décrivant les éléments de formulaire sont indépendants, de ces derniers. C’est pourquoi Il faut définir la propriété id, dans la balise de l’élément de formulaire. Cela permet de les associer avec la balise <label>

<label for=’nom ‘>Nom :</label><input type=’text’ type=’text’ id=’nom’/>

Ainsi lorsque le lecteur d’écran entrera dans un élément de formulaire il lira le label associé, le type de l‘éléments, puis le contenu, éventuel, du champ. Il est donc possible de savoir quoi saisire.Les options de HTML5, sont également annoncées par le lecteur d’écran. ARIA propose aussi les options « required »  et « invalid », qui peuvent sembler être en doublon, mais il est préférable d'utiliser les deux, car l’implémentation varie d’un navigateur à l’autre et utiliser les deux augmente les chances que l‘information soit accessible, car il est probable qu’au moins une des deux solution soit prise en compte.

* Saisie obligatoire « role= »required».
* Valeur incorrecte « role= »invalid ».

\*\*\*ajouter les propriétés html5 correspondantes\*\*

### Les éléments cliquables

Il est de plus en plus courant de substituer les boutons HTML par des SPAN ou d'utiliser des images pour simuler des boutons ou des éléments cliquables. Dans la mesure du possible les bouton HTML devraient être préférés.

Néanmoins si vous souhaité utiliser un bouton personnalisé, pour le rendre accessible vous devez l’adapter Selon une des solution suivantes.

#### Les boutons html hors formulaire

Les « <input type= »bouton » … » utilisé en dehors d’un formulaire et associés à un code HTML sont accessibles, et sont actionnable avec la touche « entré et « espace ».

Ils ne demandent pas d’adaptation, c’est pourquoi ils devraient être préférés.

<input type=*"button"* value=*"bouton standard"* onClick="alert('clicked')" />

#### Les boutons sur des liens

On peut ajouter la propriété « role= ‘button’ » sur un lien, mais il ne faut pas oublier qu’un lien est activé seulement pour la touche entré et pas par la barre espace, il faut donc ajouter un gestionnaire de clavier.

#### Les boutons <span> et <div>

* Il faut leur attribuer une propriété « rôle » « . avec la valeur « button ».
* les rendre foccusable, si il ne le sont pas.

leur attribuer un nom accessible à l’aide des attributs aria-label ou aria-labelledby, si il n’y a pas de texte à l’intérieur de la balise.

* Ajouter un gestionnaire de clavier JavaScript , car l'évènement onClick concerne seulement la sourie, il faut gérer la touche espace et entré.

<input type=*"button"* value=*"bouton standard"* onClick="alert('clicked')" />

Valider

</span>

#### Les boutons interrupteur à deux ou trois état.

Il faut utiliser les propriétés ARIA

* « role= »button » » .
* Utiliser un gestionnaire de clavier.

La propriété « aria-pressed » permet de fixer l’état :

• « false », le bouton n’est pas activé.

* « True », le bouton est activé.

« mixed », le bouton est dans un état indéterminé (bouton de Schrödinger).

Il faut gérer l’état avec le JavaScript, et changer le rendu visuel en conséquence.

<span role=*"button"* tabIndex=*"0"* aria-pressed=*"true"* onClick="switchButton(this)" onKeyUp="if(isValidated(event)) {switchButton(this)}">

Interrupteur

</span>

#### Les boutons images

Il faut s’assurer qu’ils ont une alternative textuelle, surtout quand ils n’ont pas de texte comme pour un bouton qui simulera une icône. Il faut utiliser la propriété ARIA « label » ou « labelledby » pour fournir un texte descriptif .

Globalement ils se gèrent comme un bouton à deux états, il faut juste changer les images et le texte associé à l’image.

### Les tableaux

Les tableaux ne doivent pas être utilisés pour mettre en forme un contenu, comme un formulaire car le lecteur d’écran lira à la fois les informations du tableau et du formulaire : il y aura trop d’informations lues à l’utilisateur.

La navigation dans un tableau peut se faire de cellule en cellule dans tous les sens ou bien de ligne en ligne. Il est possible de demander le titre de la colonne. Le lecteur d’écran peut aussi rappeler automatiquement le titre de la colonne et de la ligne lorsque on entre dans une cellule .C’est pourquoi il est important d’utiliser des entêtes pour les titres de colonnes et de lignes avec <th>

\*\*comparé avec aria et html5\*\*

### Mise à jour de contenu

Lorsqu'une partie du document est mise à jour, il faut le notifier à l'utilisateur. Dans notre cas au lecteur d'écran, qui lira l'information à l'utilisateur, sans modifier le focus et la position de son curseur.

Il faut :

* définir la zone concernée, avec le « role= »region » » .
* déterminer la portée de la modification, faut-il relire toute la zone ou seulement la dernière information ajoutée, comme pour un flux d’informations par exemple ?

\*\*écrire un exemple et définir les propriétés ARIA\*\*

<div role="region" aria-live="polite" aria atomic="true" aria-busy="false">

Zone de texte changeante

</div>

|  |  |
| --- | --- |
| Propriété | Définition |
| Role= »region » | Définit la zone live |
|  | Après une modification :   * « true », il faut relire toute la zone * « false », seulement la modification * juste ce qui a été ajouté, |
| aria-live="polite" |  |
| busy="false" |  |
|  |  |
| " aria- |  |

## Exemple 1 : Page structurée

<!DOCTYPE html>

<html lang="fr">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>Hidden text</title>

<link href="css/style.css" rel="stylesheet" type="text/css">

</head>

<body role="document">

<header role="banner">

Header informations

</header>

<div class="blindZone">

If you can read this text without a screen reader in your web browser, you are an Alien !

In this DIV, you can put informations to screen reader user.

Like a Lin to mute music, or disturbing options to screen reader user.

</div></div>

There is a secret message to scrren reader users only.

              <main role="main">

              Main content of the ppage.

              </main>

               <footer role="contentinfo">

              Footer informations

              </footer>

</body>

</html>

--------------------------------------------------------------------------------

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Gestionnaire de clavier

//Check if the keys space or enter have been pressed

**function** isValidated(event) {

event || window.event;

**if** (event.keyCode === 32 || event.keyCode === 13) {

**return** **true**;

} **else** {

**return** **false**;

}

}

## Gestionnaire d’état pour le bouton à état

// Switch the button to other state

**function** switchButton(button) {

**var** buttonState = button.getAttribute("aria-pressed");

**if**(buttonState == 'true') {

buttonState = **false**;

} **else** {

buttonState = **true**;

}

button.setAttribute("aria-pressed", buttonState);

}

## Bouton switch

Switch bouton

              <!DOCTYPE html>

              <html lang="en">

<head>

              <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

              <title>Switch button</title>

              <link href="css/style.css" rel="stylesheet" type="text/css">

              <script src="JavaScript/accessibilityFacilities.js" type="text/javascript"></script>

              </head>

              <body role="document">

                             <main role="main">

                             <h1>Switch buttons</h1>

                             <h2>Description</h2>

                             If you create a mutiple state button, you should :

                             <ul>

                             <li>Add aria property : role="button""".</li>

                             <li>Add "" tabIndexproperty to set focus enabled

              with tab key.</li>

                             <li>add aria property"aria-pressed" with values "true", "false", or "mixed"</li>

                             <li>manage the change ofaria property "aria-pressed" in JavaScript code</li>

                             </ul>

                             <h2>Example</h2>

                             <img src="img/selected.png" alt="Selected" role="button" tabIndex="0" aria-pressed="true" onClick="switchButton(this)" onKeyUp="if(isValidated(event))

{switchButton(this)}" id="button1"/>

                             <h2>Code</h2>

                             &lt;img src="img/selected.png" alt="Selected" role="button" tabIndex="0" aria-pressed="true" onClick="switchButton(this)"

onKeyUp="if(isValidated(event)) {switchButton(this)}" id="button1"/&gt;

                             </main>

              </body>

              </html>

## Tableau div avec titretitre

tableau div 2 avec titres colonnes et titres lignes

<!DOCTYPE html>

<html lang="fr">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>Table with titles on columns and rows</title>

<link href="css/style.css" rel="stylesheet" type="text/css">

</head>

<body role="document">

              <main role="main">

<h1>Table with titles on columns and rows</h1>

<h2HTML version</h2>

<table>

<tr>

<th></th>

<th>Juanuary</th>

<th>February  </th>

<th>March</th>

</tr>

<tr>

<th>France</th>

<td>1</td>

<td>2</td>

<td>3</td>

</tr>

<tr>

<th>Germany</th>

<td>4</td>

</td>5</td>

<td>6</td>

</tr>

</table>

<h2><DIV and SPAN version</h2>

<ul>

<li>You can use attribute "columnheader" to describe lines</li>

</ul>

                             <div role="grid">

<div role="row">

<span role="gridcell"></span>

<span role="columnheader">January</span>

<span role="columnheader">February  </span>

<span role="columnheader">March</span>

</div>

<div role="row">

<span row="rowheader">France</span>

<span role="gridcell">1</span>

<span role="gridcell">2</span>

<span role="gridcell">3</span>

</div>

<div role="row">

<span row="rowheader">Germany</span>

<span role="gridcell">4</span>

<span role="gridcell">5</span>

<span role="gridcell">6</span>

</div>

              </div>

</main>

</body>

</html>

--------------------------------------------------------------------------------

Tableau DIV simple

Tableau div 1

<!DOCTYPE html>

<html lang="fr">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>simple table with column titles</title>

<link href="css/style.css" rel="stylesheet" type="text/css">

</head>

<body role="document">

              <main role="main">

<h1>Table</h1>

Simple table with column header.

<h2Classic html table</h2>

<ul>

<li>Prefer simple HTML table with tags table, td, th...</li>

</ul>

<table>

<tr>

<th>January</th>

<th>febraruy</th>

<th>March</th>

</div>

<tr>

<td>1</td>

<td>2</td>

<td>3</td>

</tr>

<tr>

<td>4</td>

<td>5</td>

<td>6</td>

</tr>

</table>

<h2>With DIV and Span</h2>

Use role attribut to give meaning to div and span

<ul>

<li>use "grid" value to asign  role as table to div</li>

<li>use "row" to div containing row< as tr/li>

<li>use "columnHeader" to span contaning column title as th</li>

<li>use "gridCell" to span containg data as td</li>

<tab

</ul>

                             <div role="grid">

              <div role="row">

<span role="rowheader">January</span>

<span role="rowheader">Febraruy</span>

<span role="rowheader">March</span>

</div>

<div role="row">

<span role="gridcell">1</span>

<span role="gridcell">2</span>

<span role="gridcell">3</span>

</div>

<div role="row">

<span role="gridcell">4</span>

<span role="gridcell">5</span>

<span role="gridcell">6</span>

</div>

              </div>

              <h2>Code</h2>

lt;div role="grid"gt;

              &lt;div role="row"&gt;

&lt;span role="rowheader"&gt;January&lt;/span&gt;

&lt;span role="rowheader"&gt;Febraruy&lt;/span&gt;

&lt;span role="rowheader"&gt;March&lt;/span&gt;

&lt;/div&gt;

&lt;div role="row"&gt;

&lt;span role="gridcell"&gt;1&lt;/span&gt;

&lt;span role="gridcell"&gt;2&lt;/span&gt;

&lt;span role="gridcell"&gt;3&lt;/span&gt;

&lt;/div&gt;

&lt;div role="row"&gt;

&lt;span role="gridcell"&gt;4&lt;/span&gt;

&lt;span role="gridcell"&gt;5&lt;/span&gt;

&lt;span role="gridcell"&gt;6&lt;/span&gt;

lt;/divgt;

              lt;/divgt;

              <h2gt;Code<;/h2gt;

lt;div role="grid"gt;

              &lt;div role="row"&gt;

&lt;span role="rowheader"&gt;January&lt;/span&gt;

&lt;span role="rowheader"&gt;Febraruy&lt;/span&gt;

&lt;span role="rowheader"&gt;March&lt;/span&gt;

&lt;/div&gt;

&lt;div role="row"&gt;

&lt;span role="gridcell"&gt;1&lt;/span&gt;

&lt;span role="gridcell"&gt;2&lt;/span&gt;

&lt;span role="gridcell"&gt;3&lt;/span&gt;

&lt;/div&gt;

&lt;div role="row"&gt;

&lt;span role="gridcell"&gt;4&lt;/span&gt;

&lt;span role="gridcell"&gt;5&lt;/span&gt;

&lt;span role="gridcell"&gt;6&lt;/span&gt;

&lt;/div&gt;

              &lt;/div&gt;

              &lt;h2&gt;Example&lt;/h2&gt;

                                           &lt;div role="grid"&gt;

              &lt;div role="row"&gt;

&lt;span role="rowheader"&gt;January&lt;/span&gt;

&lt;span role="rowheader"&gt;Febraruy&lt;/span&gt;

&lt;span role="rowheader"&gt;March&lt;/span&gt;

&lt;/div&gt;

&lt;div role="row"&gt;

&lt;span role="gridcell"&gt;1&lt;/span&gt;

&lt;span role="gridcell"&gt;2&lt;/span&gt;

&lt;span role="gridcell"&gt;3&lt;/span&gt;

&lt;/div&gt;

&lt;div role="row"&gt;

&lt;span role="gridcell"&gt;4&lt;/span&gt;

&lt;span role="gridcell"&gt;5&lt;/span&gt;

&lt;span role="gridcell"&gt;6&lt;/span&gt;

&lt;/div&gt;

              &lt;/div&gt;

</main>

</body>

</html>

images bouton

--------------------------------------------------------------------------------

This message is for the designated recipient only and may contain privileged, proprietary, or otherwise confidential information. If you have received

it in error, please notify the sender immediately and delete the original. Any other use of the e-mail by you is prohibited. Where allowed by local law,

electronic communications with Accenture and its affiliates, including e-mail and instant messaging (including content), may be scanned by our systems

for the purposes of information security and assessment of internal compliance with Accenture policy.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

www.accenture.com

notselected.png

selected.png

images bouton

## images bouton

--------------------------------------------------------------------------------