CSS in Action: Invisible Content with Screen Readers

http://www.webaim.org/techniques/css/invisiblecontent/

Hiding Text from Sighted Users

Fortunately, there are ways of resolving the conflicts between the needs and desires of visual users and those of screen reader users. This paper examines a few circumstances in which hiding text from visual users can be beneficial, and proposes a solution which allows HTML to be hidden without compromising the accessibility or semantic integrity of the document, and which works across browsers and platforms.

The essence of the technique proposed in this document is to hide the content above the viewable area of the browser and to also shrink the content to a height and width of 1 pixel. The combination of moving the content and shrinking it is what allows this technique to work across a wide range of browsers and platforms.

Sample Code 1

The following code should appear in the style sheet:

```
.hidden
{position:absolute;
left:0px;
top:-500px;
width:1px;
height:1px;
overflow:hidden;}
```

The CSS class should then be referenced from within the tag of the element being hidden, as shown:

```
<div class="hidden">This text is hidden.</div>
```

Sample Code 2

The following code should appear in the style sheet:

```
.hidden
{
position:absolute;
left:0px;
top:-500px;
width:1px;
height:1px;
overflow:hidden;
}

h1
{
height:30;
width:60;
background-image:url(h1.jpg);
}
```

The CSS class should then be referenced from within the tag of the element being hidden, as shown:

```
<h1><img src="h1" alt="" height="30" width="60">This heading text is hidden.</h1>
```

Sample Code 3

The following code should appear in the style sheet:

```
#skip a, #skip a:hover, #skip a:visited
{
position:absolute;
left:0px;
top:-500px;
width:1px;
height:1px;
overflow:hidden;
}

#skip a:active
{
position:static;
width:auto;
height:auto;
}
```

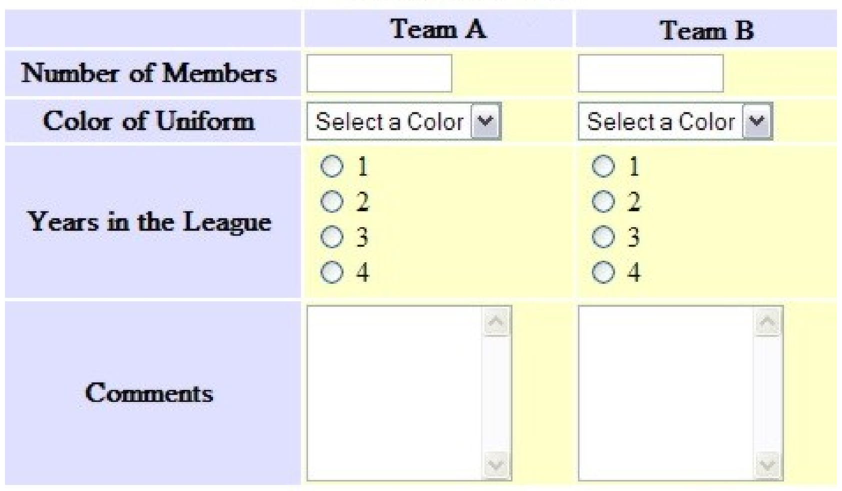
The CSS class should then be referenced from within the tag of the element.

```
<div id="skip"><a href="#content">Skip to Main Content</a></div>
```

Forms within data tables

To a visual user, table header cells can perform the dual function of organizing table content and also providing labels for the form elements within that table, as seen in the screenshot of a form within a data table below.

Table 1 (screenshot)



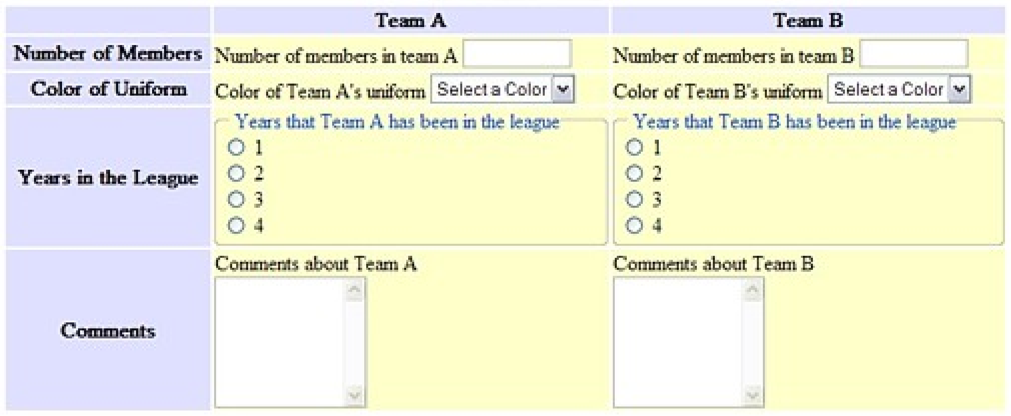
	Team A	Team B
Number of Members		
Color of Uniform	Select a Color	Select a Color
Years in the League	○ 1 ○ 2 ○ 3 ○ 4	○ 1 ○ 2 ○ 3 ○ 4
Comments		

Figure 1. Data table used to provide "labels" for form elements

To a screen reader user, the table row and column headers are somewhat useful in terms of understanding the layout of the table, but the headers do not act as form labels. When screen reader users tab from one form element to another, they will not hear the table headers read to them. In fact, they will not hear any label at all. Screen readers require text labels. Ideally, these labels should be wrapped in the <label> tag, as recommended by WCAG 1.0. Additional labeling and grouping can be accomplished by using the <fieldset> and <legend> tags.

In this particular instance, however, visual users will not receive any added benefit from the visual text labels. To them, such text labels would be redundant with the table headers, since, in a visual sense, these headers already provide adequate labels for the form elements. Here is how the same table would look to visual users if standard text labels were added, with the <label> tag, <fieldset> tag, and <legend> tag:

Table 2 (screenshot)



	Team A	Team B
Number of Members	Number of members in team A	Number of members in team B
Color of Uniform	Color of Team A's uniform Select a Color	Color of Team B's uniform Select a Color
Years in the League	Years that Team A has been in the league ○ 1 ○ 2 ○ 3 ○ 4	Years that Team B has been in the league ○ 1 ○ 2 ○ 3 ○ 4
Comments	Comments about Team A	Comments about Team B

Figure 2. Form with labels within a data table.

Though screen reader users will be happy with this version of the table, most sighted users will find the additional text to be a distraction. To visual users, the table has just become more crowded, wordy, and harder to understand at a glance. This is a situation in which the addition of markup intended to benefit screen reader users interferes with the accessibility, or at least the user-friendliness of the content to visual users.

Sample Code 4

The following code should appear in the style sheet:

```
.hidden
{
position:absolute;
left:0px;
top:-500px;
width:1px;
height:1px;
overflow:hidden;
}
```

The CSS class should then be referenced from within the tag of the element being hidden, as shown:

```
…
<label for="amembers" class="hidden">Number of members in team A</label>
…
```

Multiple form elements that "share" a single label

Another example of apparent incompatibility between the needs of screen readers users and visual users occurs when developers create multiple form elements that seem as though they ought to belong to the same label. A common example of this is when two, or more text input elements are used for phone numbers.

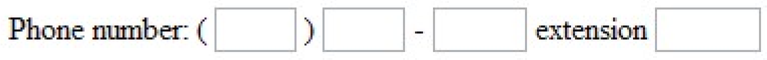


Figure 3. Form labels that apply to more than one form element.

In the screenshot above, most visual users in North America will understand that the individual text input areas correspond to the different sections of standard phone numbers. Screen reader users may attempt to enter the entire phone number in the first box. When they discover that the box limits them to only 3 characters, this will likely lead to some confusion. Some users will be able to figure out the entire context after experimenting with it, but this kind of experimentation takes time, and is unnecessary.

The most obvious workaround for this particular problem would be to combine all of the text input boxes into a single text input box, and then provide the appropriate label. This may be the best solution in most circumstances in almost every way. Nevertheless, the CSS technique can be applied to this situation also.

Sample Code 5

```
.hidden
{
position:absolute;
left:0px;
top:-500px;
width:1px;
height:1px;
overflow:hidden;
}
```

The CSS class should then be referenced from within the tag of the element being hidden, as shown:

```
<form method="post" action="">
<p>Phone number:
(
<label for="area" class="hidden">Area code</label>
<input name="area" type="text" size="3" maxlength="3" id="area" />
)
<label for="first" class="hidden">first 3 digits</label>
<input name="first" type="text" size="3" maxlength="3" id="first" />
-
<label for="last" class="hidden">last 4 digits</label>
<input name="last" type="text" size="4" maxlength="4" id="last" />
<label for="ext" class="hidden">extension</label>
<input name="ext" type="text" size="5" maxlength="5" id="ext" />
</p>
<p><input type="submit" name="Submit" value="Submit" /></p>
</form>
```

Sample Code 6

The following code should appear in the style sheet:

```
.hidden
{ position:absolute;
left:0px;
top:-500px;
width:1px;
height:1px;
overflow:hidden;
}
```

The CSS class should then be referenced from the <label> tag, as shown:

```
<div class="hidden">Begin main menu.</div>
...
<div class="hidden">End main menu.</div>
```