Web design and development II (CW2)

Class 9

Creating a custom alert window using CSS and a JavaScript onclick event

JavaScript alert windows can be very useful, but they are not especially elegant.

Although alert windows can't be customized, a solution is to design and position an alert window using CSS, and use JavaScript event to show and hide it using events.

Procedures :

1. Create the web page.

2. Create and position the custom alert window using CSS.

3. Use ***display: none;*** to hide the window.

4. Create a JavaScript event which will call a function responsible of changing the display of the hidden window (for *block* or *flex*, for instance).

   Another function will have to be created to change the display back to none.

CSS :

```
div#popup {
        position: absolute;
        display: none;
        align-items: center;
        justify-content: center;
        top: 25%;
        left: 35%;
        width: 300px;
        height: 200px;
        background: red;
        color: white;
        font-size: 2rem;
        font-weight: bold;
        border-radius: 30px;
        border: double 10px white;
}
```

HTML :

```
<div id="popup">My alert window</div>

<header>

<nav>
<a href="#" onclick="popup()"><div>Link</div></a>
<a href="#"><div>Link</div></a>
<a href="#"><div>Link</div></a>
</nav>

</header>
```

JavaScript :

```
function popup(){
document.getElementById("popup").style = "display:-
flex";
}
```

Pseudo-class :target

Pseudo-class *:target* allows to hyperlink contents and to give those different CSS styles.

In the example beside, hyperlinks are related to contents stylized using ***id*** (*text1* and *text2*).

Pseudo-class *:target* defines the styles applied to targets.

HTML:

```
<p><a href="#text1">Jump to text 1</a></p>
<p><a href="#text2">Jump to text 2</a></p>

<p id="text1"><b>New content 1...</b></p>
<p id="text2"><b>New content 2...</b></p>
```

CSS:

```
:target {
        border: 2px solid black;
        background-color:lightblue;
}
```

Create a cunstom alert window using CSS and :target

We have seen it was possible to create and show/hide a custom alert window using JavaScript. But it is perfectly possible to do so using CSS only.

Une solution consiste à créer une fenêtre d'alert (pop-up) à l'aide des CSS et de les afficher à l'aide d'un événement JavaScript qui lancera des fonctions simples.

Marche à suivre :

1. Créer la page web de base.

2. Créer la fenêtre en position absolue

   (ou autres) et la cacher à l'aide de *display: none*.

3. Créer un événement qui lance la fonction permettant de changer le *display* et ainsi de faire apparaître la fenêtre.

CSS :

```
.popup {
        position: absolute;
        display: flex;
        align-items: center;
        justify-content: center;
        top: 25%;
        left: 35%;
        width: 300px;
        height: 200px;
        background: red;
        color: white;
        font-size: 2rem;
        font-weight: bold;
        border-radius: 30px;
        border: double 10px white;
}

.popup:target {
        display: none;
}

.closebtn {
        position: absolute;
        right: 10px;
        top: 10px;
        text-decoration: none;
        font-size: 35px;
        font-weight: bold;
        color: #fff;
}
```

HTML :

```
<nav>
        <a href="#link">Lien 1</a>
</nav>

<div class="popup" id="link">
        <a href="#" class="closebtn">x</a>
        My window
</div>
```

Create content hidden in tabs using CSS and :target

The procedure is the same as with the custom alert window.

In the example beside, different contents are stored and hidden into three containers, each of them having a different id.

Using pseudo-class :target, it ensures the content will be revealed if a hyperlink targets them (using the proper id).

CSS:

```
.tab div {
        display: none;
}

.tab div:target {
        display: block;
}
```

HTML:

```
<div class="tab">
<a href="#link1">Link 1</a>
<a href="#link2">Link 2</a>
<a href="#link3">Link 3</a>

        <div id="link1">
         <h1>Content 1</h1>
         <p>Bla, bla, bla...</p>
        </div>

        <div id="link2">
        <h1>Content 2</h1>
        <h4>Bla, bla, bla...</h4>
        </div>

        <div id="link3">
         <h1>Content 3</h1>
        <p>Bla, bla, bla...</p>
        </div>

</div>
```

Create a drop-down menu using CSS

Drop-down menus are made of first level links containing, if needed, second level links.

In the example beside, placed between <nav> tags, a first pair of <ul> tags groups a first level link followed by a second pair of <ul> tags (styled with class *submenu*) grouping the second level links.

So every of these groups can be aligned horizontally, *display:inline-block* is used.

Selector ***ul ul*** allows to position second level links, bringing them in foreground (using *z-index*) and hiding them (using *display:none*).

Selector ***ul:hover .submenu*** allows changing display for *block* when hovering the first level link.

HTML :

```
<nav>
        <ul>
                <a href="#">Lien 1</a>
                <ul class="submenu">
                        <a href="#">A</a>
                        <a href="#">B</a>
                        <a href="#">C</a>
                </ul>
        </ul>
</nav>
```

CSS :

```
ul {
        position: relative;
        display: inline-block;
        margin: 0px;
        padding: 0px;
        background: lightyellow;
}

ul ul {
        display: none;
        position: absolute;
        z-index: 1;
}

ul:hover .submenu {
        display: block;
        width: 100%;
}

ul a {
        display: inline-block;
        padding: 10px 20px;
}

ul ul a {
        display: inline-block;
        padding: 10px 20px;
        width: 100%;
        border: solid black 1px;
}
```