La liaison d'événements

Temps de lecture : 6 minutes

Pour l'instant nous n'avons vu de la liaison de données que dans un seul sens : d'un composant vers un élément.

Il est temps d'aborder la liaison de données de l'élément vers le composant : la liaison d'événements (ou *event binding*).

La seule manière de savoir qu'une action de l'utilisateur s'est produite est d'écouter certains événements tels que les mouvements de la souris, les frappes sur le clavier ou les clics.

Voyons un premier exemple :

```
<button (click)='sauvegarde()'>Sauvegarder</button>
```

Dans cet exemple nous écoutons l'événement `click` sur l'élément `button`, et nous exécutons la méthode `sauvegarde()` sur notre composant.

Nous pourrions également utiliser une autre notation, appelée canonique :

```
<button on-click='sauvegarde()'>Sauvegarder</button>
```

1) L'événement cible

La liste des événements disponibles peut se trouver à cette adresse : lien

Il s'agit des événements du DOM HTML qui permettent au JavaScript de réagir lorsqu'ils surviennent. Par exemple, il est possible comme ci-dessus d'écouter le clic sur un bouton. L'événement clic est un événement du DOM HTML qui sera écouter par Angular grâce à notre liaison.

Il suffit de rajouter des parenthèses, comme ci-dessus avec `click()`.

2) `$event`

Dans une liaison d'événements, Angular met en place un gestionnaire d'événements pour l'événement cible.

Lorsque l'événement survient, le gestionnaire exécute l'expression du template. L'expression du template implique généralement un récepteur, qui effectue une action en réponse à l'événement, comme le stockage d'une valeur.

La liaison transmet des informations sur l'événement, y compris les valeurs de certaines données, à travers un objet événement appelé `$event`.

Si l'événement est un événement natif, alors l'objet `$event` est un objet DOM avec des propriétés natives comme `target` et `target.value`.

Etudions un exemple :

Dans cet exemple, nous définissons la valeur de l'input en liant `value` à la propriété `nom` (grâce à `[value]`).

Pour écouter les modifications apportées à cette valeur (*lorsque l'utilisateur rentre quelque chose dans l'input*), nous lions l'événement `input` (grâce à `(input)`) à l'expression `nom=$any($event.target).value`.

C'est-à-dire que nous récupérons l'input de l'utilisateur grâce à `$event.target.value`, et nous l'assignons à `nom`.

`$any()` permet de désactiver le contrôle de type TypeScript pour cet exemple. En effet, nous sommes en mode strict dans le cours écrit et il faut donc désactiver le contrôle de type pour pouvoir utiliser `value` dans l'expression.