Bootstrap 4 : le Scrollspy

Ce plugin permet de relier automatiquement des éléments de navigation avec des zones HTML en utilisant un défilement. Pour information « spy » signifie espionnage, autrement dit on va espionner le défilement de la page pour le synchroniser avec la navigation.

Le principe

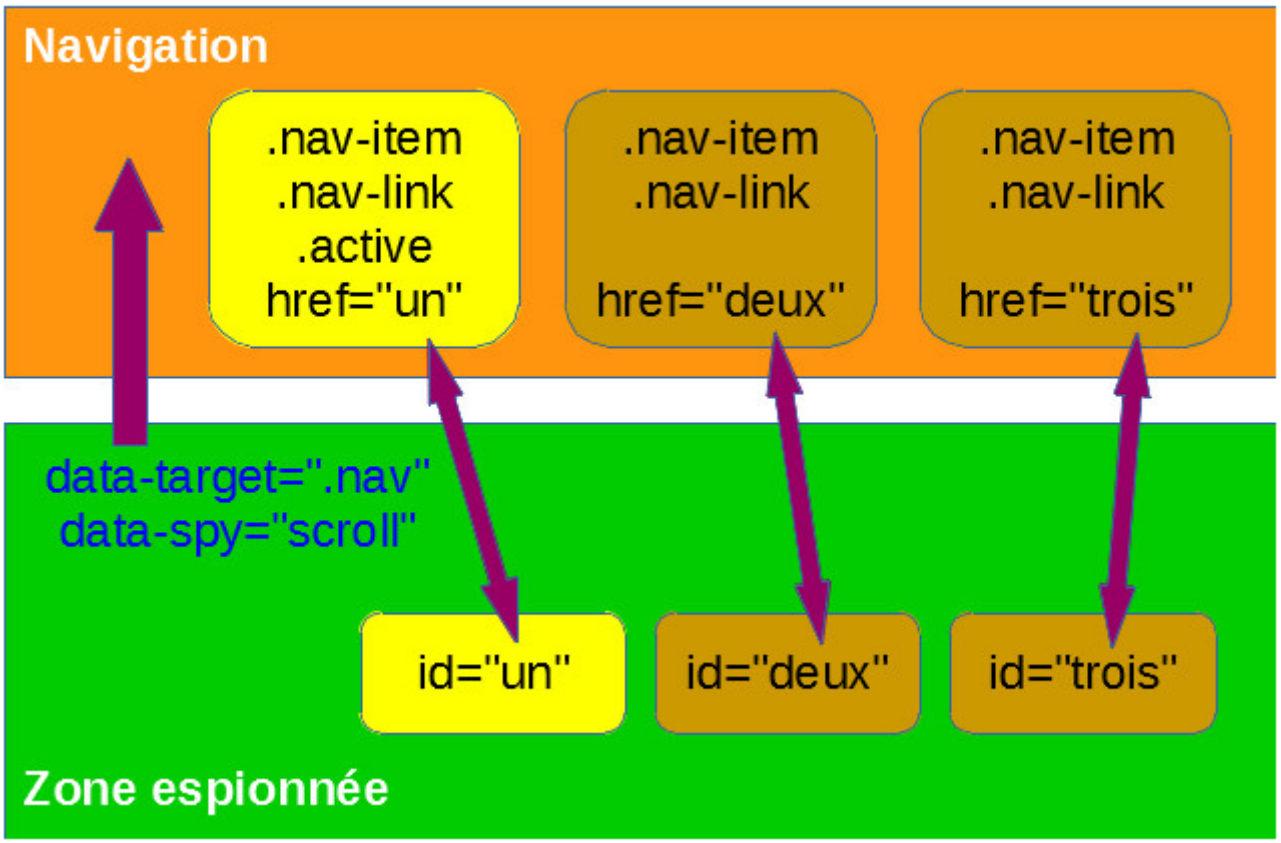
Pour faire fonctionner ce plugin il faut :

- une navigation avec le composant **nav** (ou une liste groupée),
- une zone à espionner (spying) qui doit comporter l'attribut **data-spy= »scroll** ».

D'autre part il faut relier la navigation et l'élément à espionner, dans celui-ci on utilise l'attribut **data-target** pour pointer la navigation.

Il faut aussi relier chaque élément de la navigation à son correspondant dans la zone à espionner, mais ça ce n'est pas particulier au plugin.

Ce qui peut ainsi se schématiser :



Un exemple simple

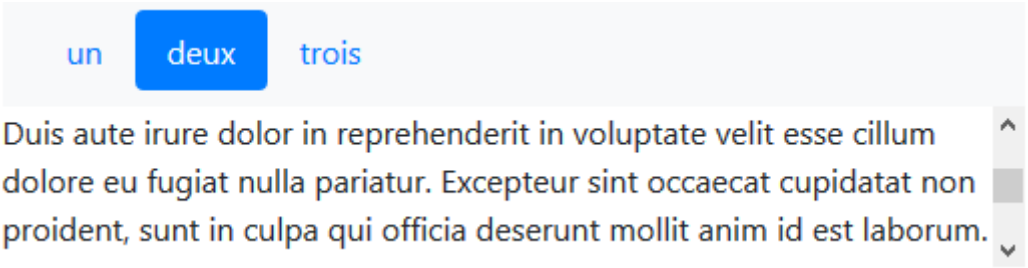
Pour illustrer ce fonctionnement prenons un exemple simple :

```
<style>
  #content {
    height: 80px;
    overflow-y: auto;
    position: relative;
  }
</style>

...

<nav id="nav" class="navbar navbar-light bg-light">
  <ul class="nav nav-pills">
    <li class="nav-item">
      <a class="nav-link active" href="#un">un</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#deux">deux</a>
    </li>
    <li class="nav-item">
```

```
      <a class="nav-link" href="#trois">trois</a>
    </li>
  </ul>
</nav>
<div id="content" data-target="#nav" data-spy="scroll" data-offset="0">
  <p id="un">Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
  <p id="deux">Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
  <p id="trois">Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.</p>
</div>
```

J’ai prévu un peu de style pour le contenu pour limiter la hauteur et avoir une barre de défilement. D’autre part il faut absolument une position relative.

Tester en ligne



Lorsqu’on fait défiler le texte on active l’item correspondant à l’identifiant de la partie de texte visible.

Mise en page

Ce plugin est essentiellement utilisé pour les mises en page de sites avec une barre de navigation et plusieurs zones à afficher, une stratégie de plus en plus utilisée. Comme un exemple est plus efficace qu’une explication, surtout pour ce genre de

fonctionnement, en voici un :

```
<!DOCTYPE HTML>
<html>

  <head>
    <meta charset="utf-8">
     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link href="css/bootstrap.css" rel="stylesheet">
                                <link       rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">
    <style>
      body {
        position: relative;
        padding-top: 50px;
      }
    </style>
  </head>

   <body  data-target=".navbar-nav"  data-spy="scroll"  data-offset="50">

  <nav class="navbar navbar-expand-lg fixed-top navbar-dark bg-dark">
     <a  class="navbar-brand"  href="#accueil">La  boutique  en délire</a>
      <button  class="navbar-toggler"  type="button"  data-toggle="collapse" data-target=".collapse">
     <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item">
          <a class="nav-link" href="#accueil">Accueil</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#produits">Produits</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#nouvelles">Nouvelles</a>
        </li>
```

```
        <li class="nav-item">
                                        <a    class="nav-link"
href="#localisation">Localisation</a>
        </li>
      </ul>
    </div>
  </nav>

  <section id="accueil" class="bg-primary text-center text-white
py-3">
    <div class="container">
      <h1 class="py-md-2">Bienvenue dans notre espace !</h1>
       <p>"Sed ut perspiciatis unde omnis iste natus error sit
voluptatem accusantium doloremque laudantium, totam rem aperiam,
eaque ipsa quae ab illo inventore veritatis et quasi architecto
beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia
voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur
magni dolores eos qui ratione voluptatem sequi nesciunt. Neque
porro quisquam est, qui dolorem ipsum quia dolor sit amet,
consectetur, adipisci velit, sed quia non numquam eius modi
tempora incidunt ut labore et dolore magnam aliquam quaerat
voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem
ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi
consequatur? Quis autem vel eum iure reprehenderit qui in ea
voluptate velit esse quam nihil molestiae consequatur, vel illum
qui dolorem eum fugiat quo voluptas nulla pariatur?"
      </p>
    </div>
  </section>

  <section id="produits" class="bg-dark text-center text-white
py-3">
    <div class="container">
      <h1 class="py-md-2">Nos produits attractifs !</h1>
       <p>"Sed ut perspiciatis unde omnis iste natus error sit
voluptatem accusantium doloremque laudantium, totam rem aperiam,
eaque ipsa quae ab illo inventore veritatis et quasi architecto
beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia
voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur
magni dolores eos qui ratione voluptatem sequi nesciunt. Neque
porro quisquam est, qui dolorem ipsum quia dolor sit amet,
consectetur, adipisci velit, sed quia non numquam eius modi
tempora incidunt ut labore et dolore magnam aliquam quaerat
```

voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?"

      <hr>

          <button type="button" class="btn btn-light btn-block">

            <h3 class="pt-1">Outillage</h3>
          </button>

          <button type="button" class="btn btn-light btn-block">

            <h3 class="pt-1">Incendie</h3>
          </button>

          <button type="button" class="btn btn-light btn-block">

            <h3 class="pt-1">Habitation</h3>
          </button>

      <hr>
  </section>

  <section id="nouvelles" class="bg-primary text-center text-white py-3">

      <h1 class="py-md-2">Les nouvelles de nos activités !</h1>
       "Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat

voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?"

      <hr>

          <button type="button" class="btn btn-light btn-block">

            <h3 class="pt-1">Sonorisation</h3>
          </button>

          <button type="button" class="btn btn-light btn-block">

            <h3 class="pt-1">Voyage</h3>
          </button>

          <button type="button" class="btn btn-light btn-block">

            <h3 class="pt-1">Transport</h3>
          </button>

      <hr>

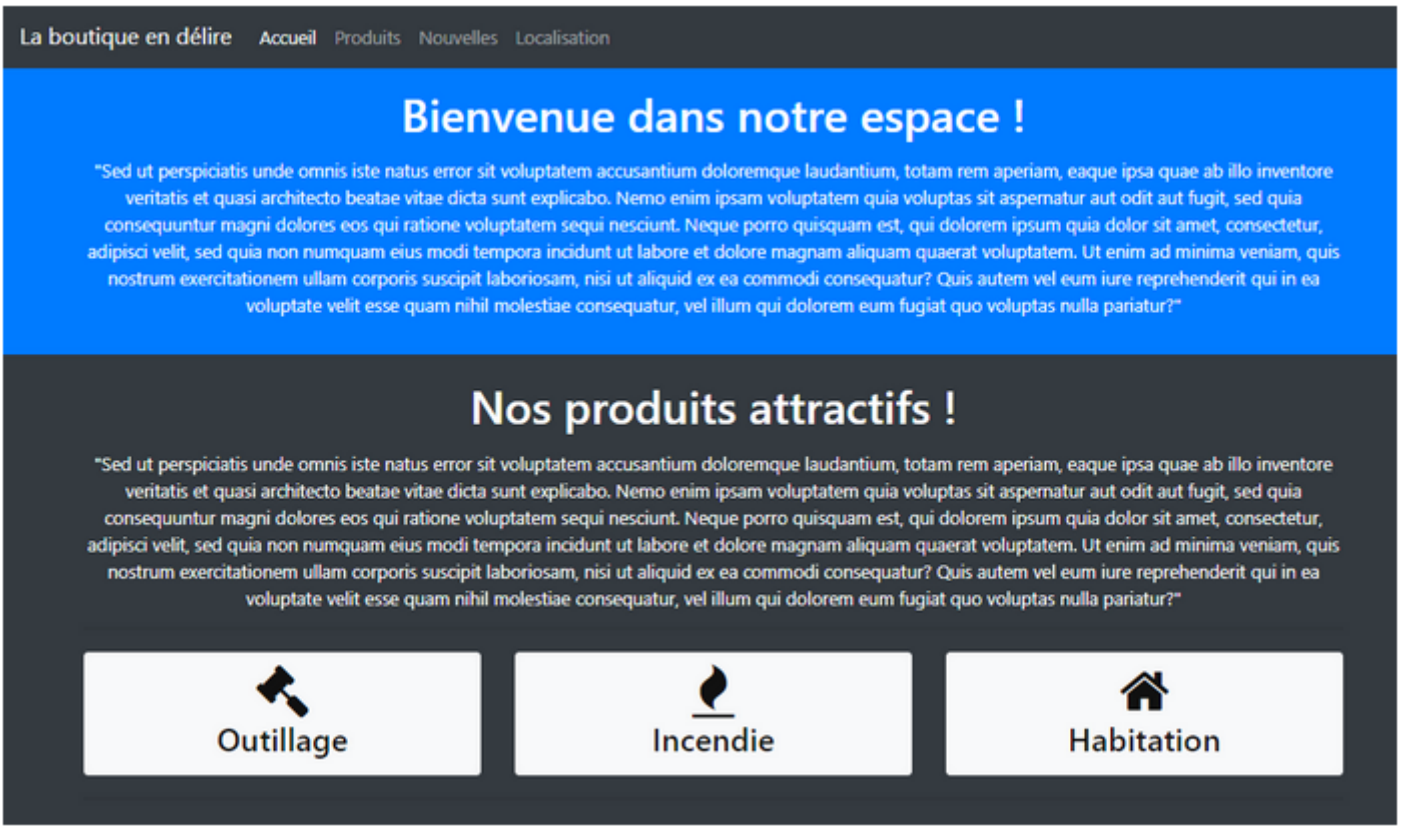
  </section>

  <section id="localisation" class="bg-dark text-center text-white py-3">

      <h1 class="py-md-2">L'emplacement de nos boutiques !</h1>
       "Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi

```
tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?"
      </p>
      <hr>
      <div class="row">
        <div class="col-md-4 my-1">
          <button type="button" class="btn btn-light btn-block">
            <span class="fa fa-road fa-3x"></span>
            <h3 class="pt-1">Routes</h3>
          </button>
        </div>
        <div class="col-md-4 my-1">
          <button type="button" class="btn btn-light btn-block">
            <span class="fa fa-cutlery fa-3x"></span>
            <h3 class="pt-1">Restaurants</h3>
          </button>
        </div>
        <div class="col-md-4 my-1">
          <button type="button" class="btn btn-light btn-block">
            <span class="fa fa-television fa-3x"></span>
            <h3 class="pt-1">Télévision</h3>
          </button>
        </div>
      </div>
      <hr>
    </div>
  </section>

                                                    <script
src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
  <script src="js/bootstrap.js"></script>
  </body>
</html>
```

Tester en ligne



Lorsqu’on fait défiler la page l’item du menu correspondant à la zone affichée s’active :

Un mouvement fluide

L'exemple fonctionne correctement, mais on peut regretter la brutalité du mouvement lorsqu'on utilise le menu. La zone demandée s'affiche spontanément et on ne comprend pas vraiment qu'il se produit un défilement. Il serait plus élégant d'avoir un mouvement fluide pour passer d'une zone à une autre. Ce n'est pas prévu au niveau du plugin, mais avec quelques lignes de Javascript on peut réaliser cela :


```
$('a.navbar-brand, a.nav-link').on('click', function(e) {
  e.preventDefault()
  $('html, body').animate({
    scrollTop: $(this.hash).offset().top - 60
  }, 1000)
})
```

[Tester en ligne]

Activation par Javascript

On a vu dans les exemples que le plugin peut être simplement mis en œuvre avec des attributs :

```
<body data-target=".nav" data-spy="scroll">
```

Il est aussi possible de le faire avec Javascript. Pour tester cette possibilité, il faut retirer ces attributs et ajouter cette ligne de code :

```
$('body').scrollspy({ target: '.nav' });
```

Vous disposez aussi de l'événement **activate.bs.scrollspy** qui est déclenché lorsqu'une nouvelle zone est affichée. L'utilité de cet événement ne saute pas aux yeux, mais si vous en avez éventuellement besoin la syntaxe est la suivante :

```
$('#monScrollspy').on('activate.bs.scrollspy', function () {
  // Action
})
```

Un souci peut se présenter si vous modifiez dynamiquement le menu au niveau du DOM parce que le plugin ne sera pas au courant de ces changements. dans ce cas il faut commander un rafraîchissement :

```
$('[data-spy="scroll"]').each(function () {
  var $spy = $(this).scrollspy('refresh')
})
```

En résumé

- Le Scrollspy permet de relier automatiquement des éléments de navigation avec des zones HTML en utilisant un défilement.
- C'est un effet de plus en plus utilisé sur les sites web.

Bootstrap 4 : info-bulles (Tooltips) et Popovers

Une info-bulle (Tooltip) est une information textuelle sommaire qui apparaît dans une bulle lorsqu'on survole ou clique un élément de la page. Un Popover peut être considéré comme une grosse info-bulle avec plus de possibilités. Ces deux plugins font appel à une

librairie externe pour le positionnement, en l'occurrence popper.

Principes communs

*Ces plugins nécessitent la librairie **popper.js** (ou la version **bundle** de Bootstrap).*

D'autre part, contrairement à ce qu'on a vu jusqu'à présent, il faut systématiquement initialiser ces plugins avec jQuery. Le plus simple est de faire un tir groupé avec toutes les info-bulles et popovers présents sur la page :

```
$(function () {
  $('[data-toggle="tooltip"]').tooltip()
  $('[data-toggle="popover"]').popover()
})
```

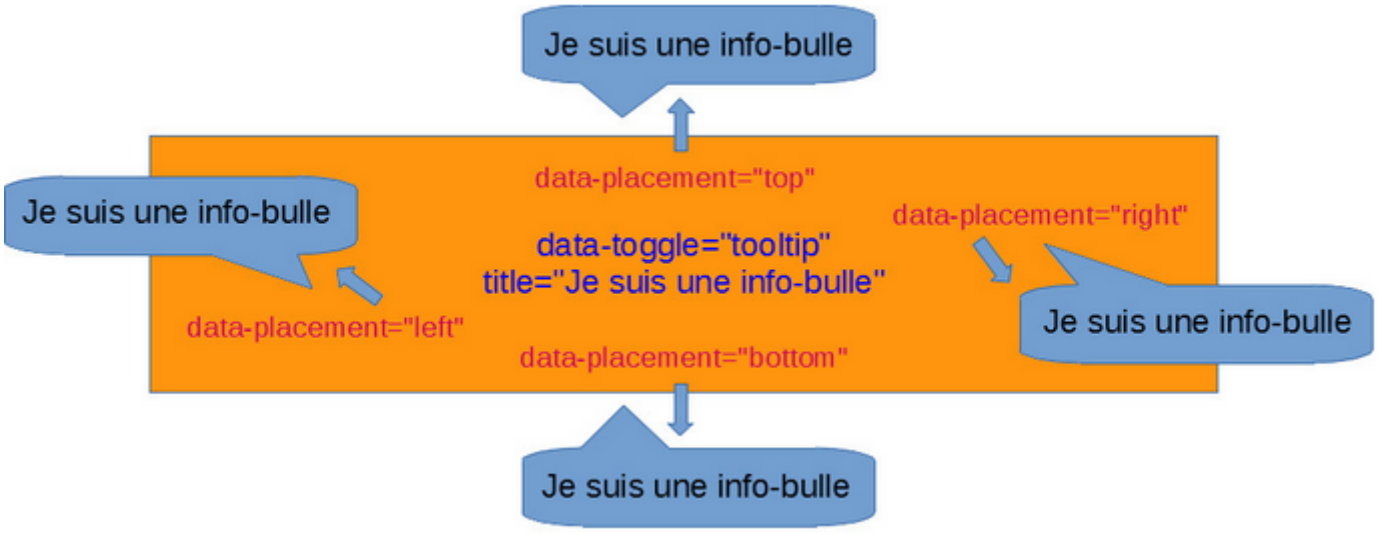
Le principe

Info-bulle (tooltip)

Pour mettre en place une info-bulle il faut :

- l'attribut **data-toggle= »tooltip »**,
- l'attribut **title** pour définir le texte de l'info-bulle,
- optionnellement l'attribut **data-placement** avec comme valeurs possibles **top**, **right** ,**bottom** et **left** pour déterminer l'emplacement. Si cet attribut est absent la valeur par défaut est **top**.

Ce qu'on peut schématiser ainsi :



Popover

Pour mettre en place un Popover il faut :

- l'attribut **data-toggle= »popover »**,
- l'attribut **data-content** pour définir le contenu,
- optionnellement l'attribut **data-placement** avec comme valeurs possibles **top**, **right**, et **left** pour déterminer l'emplacement. Si cet attribut est absent la valeur par défaut est **right**,
- optionnellement l'attribut **data-container** pour éviter de perturber le contenu du Popover avec du style d'un élément englobant.

Ce qu'on peut schématiser ainsi :

U

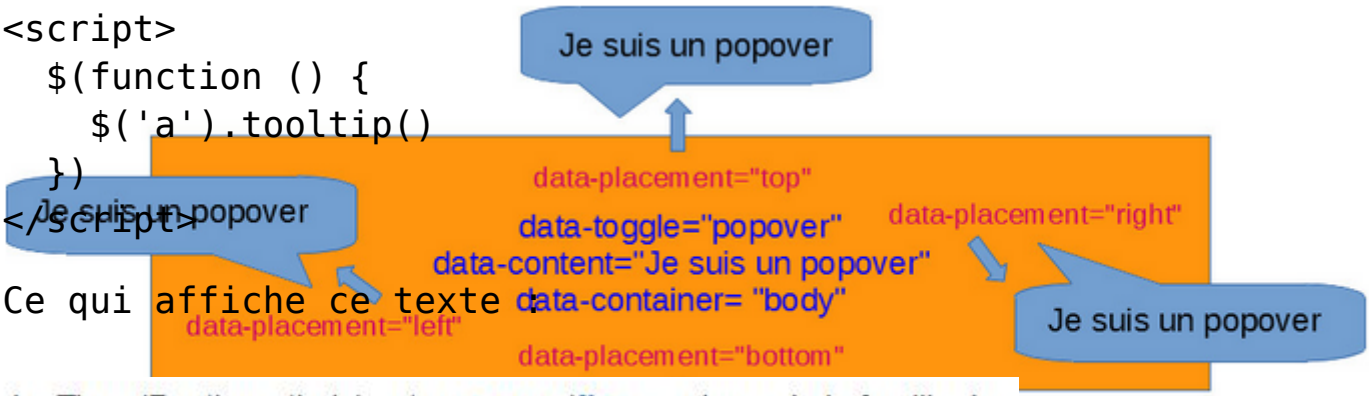
n exemple

Info-bulle (tooltip)

Considérez ce code :

```
<p>Le Tigre (Panthera tigris) est un <a data-toggle="tooltip" data-placement="bottom" title="Classe de vertébrés">mammifère</a> carnivore de la famille des félidés (Felidae) du genre Panthera. Aisément reconnaissable à sa fourrure rousse rayée de noir, il est le plus grand félin sauvage et l'un des plus grands <a data-toggle="tooltip" data-placement="top" title="Synonyme de carnassier"> carnivores</a> du monde.</p>

...

<script>
  $(function () {
    $('a').tooltip()
  })
</script>
```

Ce qui affiche ce texte :



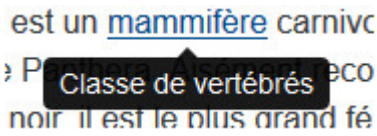
Le Tigre (Panthera tigris) est un mammifère carnivore de la famille des félidés (Felidae) du genre Panthera. Aisément reconnaissable à sa fourrure rousse rayée de noir, il est le plus grand félin sauvage et l'un des plus grands carnivores du monde.

Comme il est prévu deux info-bulles, si on passe le curseur de la souris au-dessus

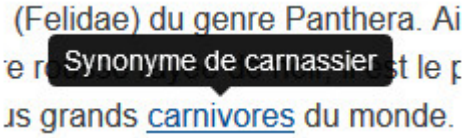
des deux liens on déclenche l'apparition de ces info-bulles :

Tester en ligne

Une info-bulle positionnée en bas :



Une info-bulle positionnée en haut :



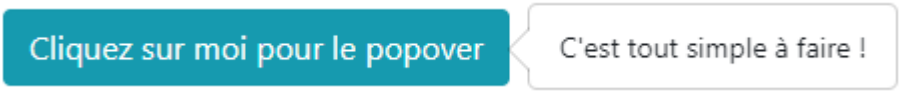
On pourrait se passer de l'attribut **data-placement= »top »** pour la deuxième info-bulle puisque c'est la valeur par défaut.

Popover

Considérez ce code :

```
<a href="#" class="btn btn-info" data-toggle="popover" data-content="C'est tout simple à faire !">Cliquez sur moi pour le popover</a>

...

<script>
  $(function () {
    $('a').popover()
  })
</script>
```

Tester en ligne

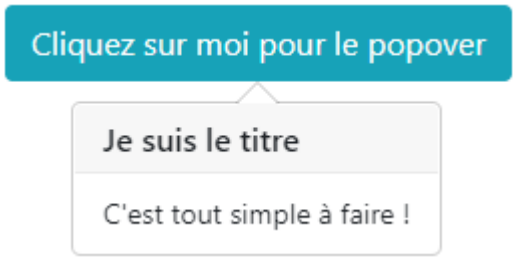


Il est possible d'ajouter un titre avec l'attribut **title** et de modifier la position avec l'attribut **data-placement** :

```
<a href="#" class="btn btn-info" data-toggle="popover" data-placement ="bottom" data-content="C'est tout simple à faire !"
```

```
title="Je suis le titre">Cliquez sur moi pour le popover</a>
```

Tester en ligne



Positionnement avec JQuery

On peut obtenir le même positionnement avec jQuery plutôt que par l'attribut **data-placement**, voici un exemple avec des info-bulles (le code est identique pour les Popovers) :

```
<p>Le Tigre (Panthera tigris) est un <a href="#" data-toggle="tooltip" title="Classe de vertébrés">mammifère</a> carnivore de la famille des félidés (Felidae) du genre Panthera. Aisément reconnaissable à sa fourrure rousse rayée de noir, il est le plus grand félin sauvage et l'un des plus grands <a href="#" data-toggle="tooltip" title="Synonyme de carnassier">carnivores</a> du monde.</p>

...

<script>
  $(function () {
   $('a:first-child').tooltip({ placement:'bottom' })
   $('a:last-child').tooltip({ placement:'top' })
  })
</script>
```

Mais évidemment dans ce cas il faut différencier les info-bulles au niveau du Javascript, ce qui ne présente pas vraiment d'intérêt dans cet exemple.

Déclenchement

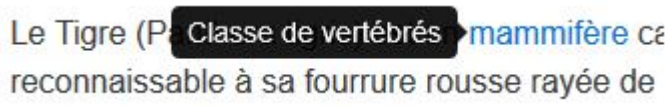
Info-bulle

Par défaut une info-bulle est déclenchée lorsque le curseur de la souris survole l'élément concerné ou si celui-ci a le focus. On peut modifier ce comportement avec l'option **trigger**. Les possibilités sont **click**, **hover**, **focus** et **manual**. Voici un exemple d'utilisation avec le même code HTML :

```
$(function (){
   $('a:first-child').tooltip({ placement: 'left', trigger:'click' })
   $('a:last-child').tooltip({ placement: 'right', trigger:'click focus' })
})
```

Maintenant la première info-bulle se déclenche seulement avec un clic et est positionnée à gauche :

Tester en ligne



La seconde info-bulle est positionnée à droite et est activée par clic et tant qu'on a le focus :

En cas d'apparition avec seulement un clic il faut un autre clic pour la disparition.

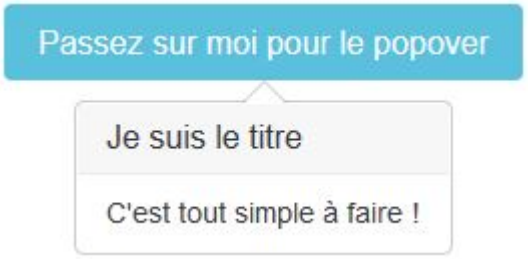
Popover

Pour un popover le comportement par défaut est un déclenchement sur un clic. Mais on dispose comme pour l'info-bulle des 4 possibilités : **click**, **hover**, **focus** et **manual**.

Voici un l'exemple vu ci-dessus mais cette fois on désactive le clic du bouton et on déclenche le popover au survol avec l'attribut **data-trigger** :

```
<a href="#" class="btn btn-info" data-toggle="popover" data-trigger="hover" data-placement ="bottom" data-content="C'est tout simple à faire !" title="Je suis le titre">Passez sur moi pour le popover</a>
```

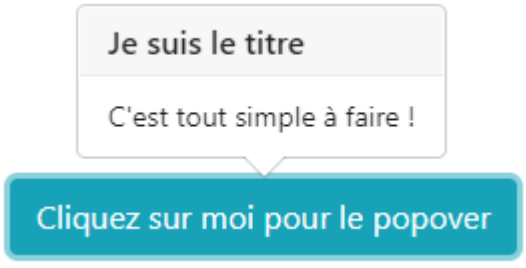
Tester en ligne



Par défaut pour refermer le popover dans le cas classique d'un déclenchement par un clic il faut recliquer. Si on veut une fermeture sur un clic ailleurs sur la page il faut utiliser l'option **trigger** avec la valeur **focus** :

```
<a href="#" class="btn btn-info" data-toggle="popover" data-trigger="focus" data-placement ="top" data-content="C'est tout simple à faire !" title="Je suis le titre">Passez sur moi pour le popover</a>
```

Tester en ligne

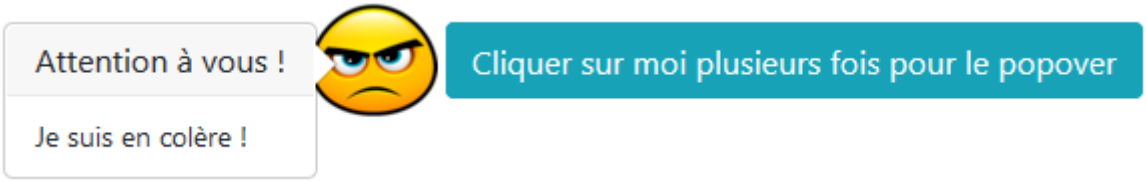


Avec l'option **manual** on peut définir une action quelconque pour déclencher le popover que l'on veut (c'est aussi applicable à une info-bulle). Regardez cet exemple :

```
<img  class="ml-5" data-toggle="popover" data-offset="20, -15" data-content="Je suis en colère !" title="Attention à vous !" src="images/visage.png">
<a href="#" id="pop" class="btn btn-info" >Cliquer sur moi plusieurs fois pour le popover</a>
```

```
...

<script>
  $(function () {
    $('img').popover({trigger:'manual'})
    $('a').click(function() {
       $('img').popover('toggle')
    })
  })
</script>
```

Tester en ligne



On initialise le popover sur l'image en déclenchement manuel :

```
$('img').popover({trigger:'manual'})
```

On utilise l'événement **click** du bouton pour lancer la fonction **popover** avec **toggle** comme paramètre :

```
$('img').popover('toggle')
```

J'ai introduit un décalage pour approcher la bulle de la bouche du personnage, j'explique ce point un peu plus loin dans ce chapitre.

Délai de déclenchement

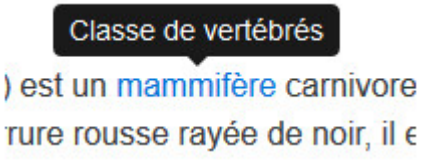
Par défaut une info-bulle (c'est identique pour un popover) s'affiche immédiatement et disparaît avec la même célérité. On peut modifier ce comportement avec l'option **delay**. On peut soit définir un délai identique pour l'apparition et la disparition, soit différencier les deux. Voici un exemple d'utilisation avec le même code HTML :

```
$(function (){
   $('a:first-child').tooltip({ delay: 400 })
   $('a:last-child').tooltip({ delay: { show: 400, hide: 200 } })
```

```
});
```

Ici on utilise l'option **delay** en la définissant à 400 ms pour le premier lien pour l'affichage et la disparition. On différencie avec 400 ms pour l'affichage et 200 ms pour la disparition pour le second.

Tester en ligne



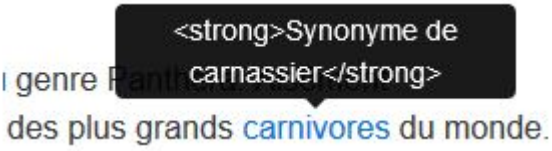
On peut aussi affecter la valeur avec un attribut :

```
<a data-toggle="tooltip" data-delay="400" title="Classe de vertébrés">
```

Contenu avec HTML

Par défaut le titre d'une info-bulle (et le contenu du popover) accepte uniquement du texte. Regardez cet exemple :

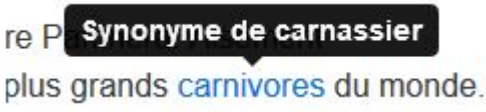
```
<p>Le Tigre (Panthera tigris) est un mammifère carnivore de la famille des félidés (Felidae) du genre Panthera. Aisément reconnaissable à sa fourrure rousse rayée de noir, il est le plus grand félin sauvage et l'un des plus grands <a href="#" data-toggle="tooltip" title="<strong>Synonyme de carnassier</strong>">carnivores</a> du monde.</p>
```



Je comptais afficher le titre en gras mais c'est raté, pour que ça fonctionne il faut que j'affecte la valeur **true** à l'option **html** :

```
<a data-toggle="tooltip" data-html="true" title="<strong>Synonyme de carnassier</strong>">
```

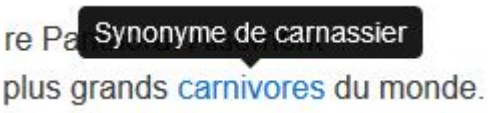
Tester en ligne



Décalage

On a vu qu'une info-bulle (c'est identique pour un popover) se positionne automatiquement par rapport à l'élément. Notre seule latitude est de choisir la position (haut, bas, gauche ou droite). Mais on peut aussi décaler cette position en jouant à la fois sur l'abscisse et l'ordonnée. Voici notre exemple de base sans décalage :

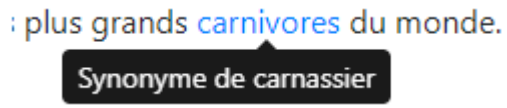
```
<p>Le Tigre (Panthera tigris) est un mammifère carnivore de la famille des félidés (Felidae) du genre Panthera. Aisément reconnaissable à sa fourrure rousse rayée de noir, il est le plus grand félin sauvage et l'un des plus grands <a href="#" data-toggle="tooltip" title="Synonyme de carnassier"> carnivores</a> du monde.</p>
```



On va maintenant faire un décalage en utilisant l'option **offset** :

```
<a href="#" data-toggle="tooltip" data-offset="-20px, -1px" title="Synonyme de carnassier">
```

Tester en ligne



Vous pouvez ainsi positionner avec précision une info-bulle.

Les événements

Vous disposez de ces 4 principaux événements pour ces deux plugins (pour le popover il suffit de remplacer le nom tooltip) :

Tooltips	Popovers	Description

show.bs.tooltip	**show.bs.popover**	Déclenché dès l'appel à la méthode show
shown.bs.tooltip	**shown.bs.popover**	Déclenché quand la bulle est devenue visible
hide.bs.tooltip	**hide.bs.popover**	Déclenché dès l'appel à la méthode hide
hidden.bs.tooltip	**hidden.bs.popover**	Déclenché quand la bulle est devenue invisible

Si vous avez besoin de l'un de ces événements la syntaxe est celle-ci :

```
$('#pop').on('show.bs.popover', function () {
  // Action
})
```

En résumé

- Les info-bulles permettent de faire apparaître un petit texte informatif pour un élément, les popovers également mais autorisent un contenu plus riche.
- On peut positionner précisément une info-bulle ou un popover.
- On peut déclencher une info-bulle ou un popover au clic, au survol, a focus, ou avec n'importe quel événement.
- On peut imposer des délais à l'affichage ou à la disparition d'une info-bulle ou d'un popover.
- On peut mettre du HTML dans une info-bulle ou un popover.

Bootstrap 4 : caroussel

(Carousel)

Un carrousel permet de faire défiler automatiquement ou manuellement des images à la manière d'un diaporama. La transition des images peut se faire par substitution ou par translation. Dans ce chapitre nous allons voir comment mettre en œuvre celui proposé par Bootstrap.

Le principe

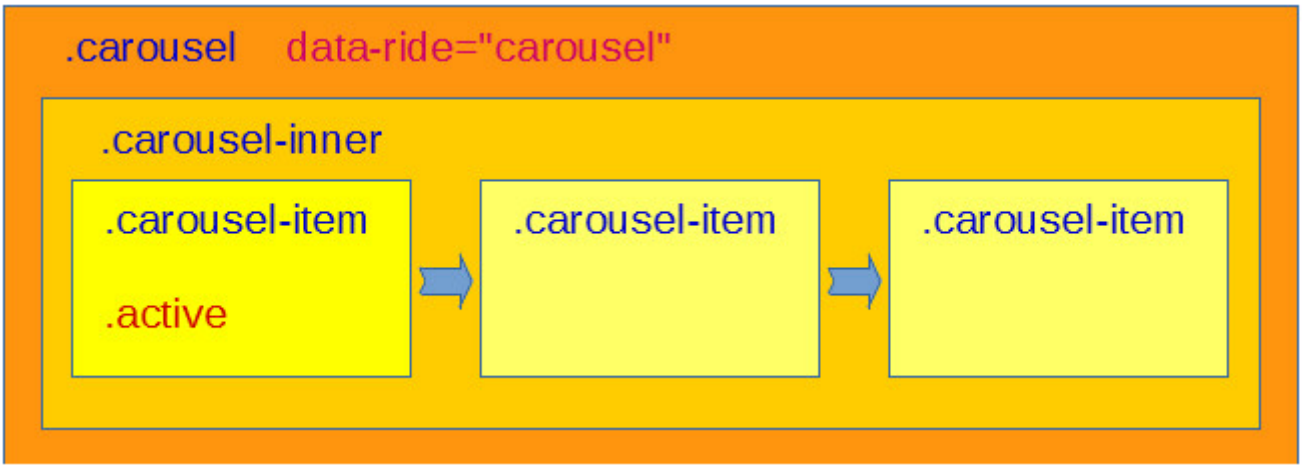
Le carrousel doit être englobé avec :

- une classe **carousel** ,
- un attribut **data-ride= »carousel »**.

Le contenu doit être englobé avec une classe **carousel-inner** si on veut disposer des contrôles.

Chaque image doit être englobée avec une classe **carousel-item**. Pour que le carrousel fonctionne il faut absolument que l'une des images soit activée avec la classe **active**.

Ce qui nous donne le schéma suivant :



Enfin il faut prévoir les classes **d-block** et **w-100** pour les images.

*La classe **d-block** (**d** pour **d**isplay) fait partie d'une série de classes bien pratiques pour la visualisation que nous avons déjà*

vues dans une précédente partie, on a par exemple ***d-none*** *pour ne rien afficher, ou* ***d-inline****. On a aussi les variations en fonction des supports, par exemple* ***d-md-block****.*

Ce qui donne le code minimal suivant pour 3 images :

```
<div class="carousel" data-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
       <img class="d-block w-100" src="images/t1.jpg" alt="Un tigre">
    </div>
    <div class="carousel-item">
       <img class="d-block w-100" src="images/t2.jpg" alt="Un tigre">
    </div>
    <div class="carousel-item">
       <img class="d-block w-100" src="images/t3.jpg" alt="Un tigre">
    </div>
  </div>
</div>
```

Tester en ligne

Si on veut un changement d'image par translation il faut ajouter la classe **slide** :

```
<div class="carousel slide" data-ride="carousel">
```

Par défaut les images se succèdent toutes les 5 secondes. On peut changer cette valeur avec l'attribut **data-interval** :

```
<div class="carousel slide" data-ride="carousel" data-interval="2000">
```

La valeur est exprimée en millièmes de secondes, donc ici on a 2 secondes.

Des titres dans les images

Il est possible de faire apparaître un titre pour chaque image dans la partie inférieure avec la classe **carousel-caption** :

```
<div class="carousel slide" data-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item active">
       <img class="d-block w-100" src="images/t1.jpg" alt="Un
tigre">
      <div class="carousel-caption d-none d-md-block">
        <h1>Un tigre</h1>
      </div>
    </div>
    <div class="carousel-item">
       <img class="d-block w-100" src="images/t2.jpg" alt="Un
tigre">
      <div class="carousel-caption d-none d-md-block">
        <h1>Un autre tigre</h1>
      </div>
    </div>
    <div class="carousel-item">
       <img class="d-block w-100" src="images/t3.jpg" alt="Un
tigre">
      <div class="carousel-caption d-none d-md-block">
        <h1>Encore un tigre</h1>
      </div>
    </div>
  </div>
</div>
```

Tester en ligne

On peut utiliser n’importe quelle typographie pour le contenu textuel.

Il est conseillé d’utiliser des classes utilitaires d’affichage pour éviter d’avoir le titre qui apparaît sur les petits supports. Ici on a utilisé ***d-md-block*** *donc on limite l’affichage du titre aux écrans moyens et grands.*

Notez que le titre vient au-dessus de l’image et qu’il faut donc adapter le style aux couleurs présentes sur l’image pour que le titre soit lisible.

Un indicateur-sélecteur de l’image affichée

Il est possible de faire apparaître un indicateur-sélecteur pour savoir quelle image est affichée et aussi pour accéder directement à une image particulière. Il faut :

- utiliser une liste ordonnée équipée de la classe **carousel-indicators**,
- identifier dans chaque élément de la liste le carrousel avec l’attribut **data-target**,

- utiliser l'attribut **data-slide-to** pour identifier numériquement la diapositive concernée.

```
<div class="carousel slide" data-ride="carousel">
  <ol class="carousel-indicators">
        <li  data-target=".carousel"  data-slide-to="0"
class="active"></li>
    <li data-target=".carousel" data-slide-to="1"></li>
    <li data-target=".carousel" data-slide-to="2"></li>
  </ol>
  <div class="carousel-inner">
    <div class="carousel-item active">
       <img class="d-block w-100" src="images/t1.jpg" alt="Un
tigre">
      <div class="carousel-caption">
        <h1>Un tigre</h1>
      </div>
    </div>
    <div class="carousel-item">
       <img class="d-block w-100" src="images/t2.jpg" alt="Un
tigre">
      <div class="carousel-caption">
        <h1>Un autre tigre</h1>
      </div>
    </div>
    <div class="carousel-item">
       <img class="d-block w-100" src="images/t3.jpg" alt="Un
tigre">
      <div class="carousel-caption">
        <h1>Encore un tigre</h1>
      </div>
    </div>
  </div>
</div>
```

Tester en ligne

On peut sélectionner la diapositive à afficher en cliquant sur l'indicateur.

Des boutons de défilement

On peut afficher des boutons à droite et à gauche pour permettre un défilement manuel des images. Il faut identifier le carrousel pour le référencer dans les liens avec **href**. Les boutons doivent être équipés :

- de la classe **carousel-control-prev** ou **carousel-control-next**,
- de l'attribut **data-slide** avec la valeur **prev** ou **next**.

```
<div id="carousel" class="carousel slide" data-ride="carousel">
  <ol class="carousel-indicators">
        <li  data-target=".carousel"  data-slide-to="0"
class="active"></li>
    <li data-target=".carousel" data-slide-to="1"></li>
    <li data-target=".carousel" data-slide-to="2"></li>
  </ol>
  <div class="carousel-inner">
    <div class="carousel-item active">
       <img class="d-block w-100" src="images/t1.jpg" alt="Un
tigre">
      <div class="carousel-caption">
        <h1>Un tigre</h1>
```

```
      </div>
    </div>
    <div class="carousel-item">
       <img class="d-block w-100" src="images/t2.jpg" alt="Un
tigre">
      <div class="carousel-caption">
        <h1>Un autre tigre</h1>
      </div>
    </div>
    <div class="carousel-item">
       <img class="d-block w-100" src="images/t3.jpg" alt="Un
tigre">
      <div class="carousel-caption">
        <h1>Encore un tigre</h1>
      </div>
    </div>
  </div>
   <a  class="carousel-control-prev"  href="#carousel"  data-
slide="prev">
    <span class="carousel-control-prev-icon"></span>
  </a>
   <a  class="carousel-control-next"  href="#carousel"  data-
slide="next">
    <span class="carousel-control-next-icon"></span>
  </a>
</div>
```

Tester en ligne

Accessibilité

Rôles

Si on ajoute des boutons de défilement il faut leur adjoindre le rôle **button** puisque la balise **<a>** n’est pas sémantiquement pertinente :

```
<a class="carousel-control-prev" href="#carousel" role="button" data-slide="prev">
```

Propriétés

Si on ajoute des boutons de défilement il faut prévoir la propriété **aria-hidden** avec la valeur **true** dans la balise qui va accueillir l’icône :

```
<span class="carousel-control-prev-icon" aria-hidden="true"></span>
```

Lecteur d’écran

Si on ajoute des boutons de défilement il faut prévoir une information pour les lecteurs d’écran :

```
<span class="sr-only">Précédent</span>
...
<span class="sr-only">Suivant</span>
```

Résumé pour les boutons de défilement

Voici donc la syntaxe de base pour les boutons de défilement :

```
<a class="carousel-control-prev" href="#carousel" role="button" data-slide="prev">
     <span  class="carousel-control-prev-icon"  aria-hidden="true"></span>
  <span class="sr-only">Précédent</span>
</a>
<a class="carousel-control-next" href="#carousel" role="button" data-slide="next">
     <span  class="carousel-control-next-icon"  aria-hidden="true"></span>
  <span class="sr-only">Suivant</span>
</a>
```

Activation avec Javascript

Dans les exemples précédents on a lancé le carrousel avec **data-ride= »carousel** ». Si on veut changer un peu le fonctionnement de base il faut le lancer de façon explicite à partir du Javascript. Donc dans la balise englobante on prévoit juste ce code :

```
<div id="carousel" class="carousel slide">
```

En l'état le carrousel ne fonctionne plus de façon automatique mais on peut le lancer avec jQuery :

```
$('.carousel').carousel();
```

Quel intérêt ?

Si on ne veut rien faire de spécial aucun mais nous allons envisager des cas où ça devient utile.

Cycle et pause

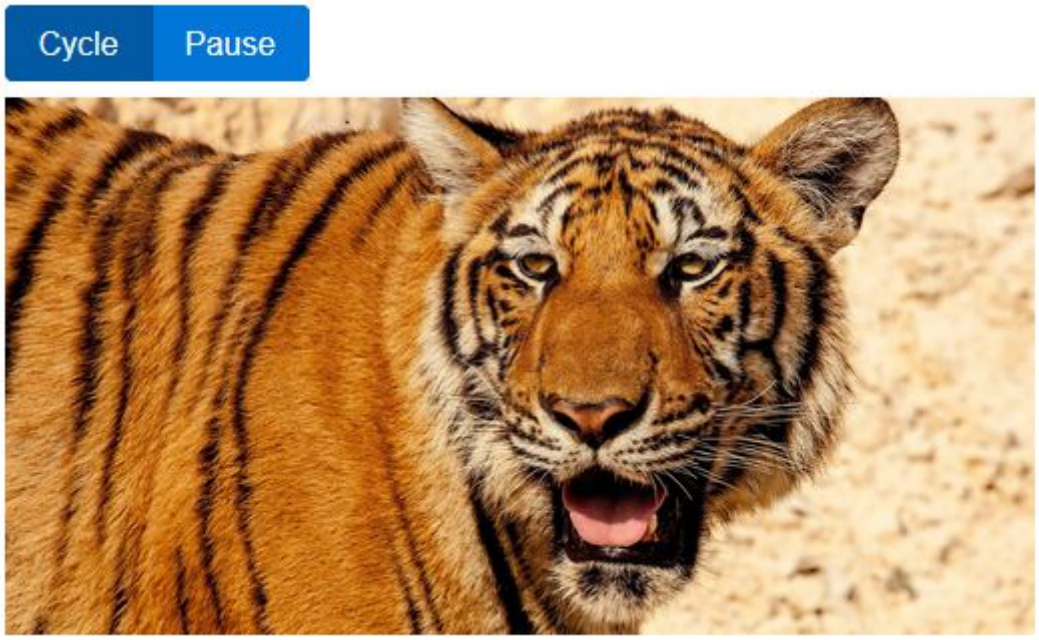
On peut avec jQuery commander le démarrage du cycle ou la pause :

```
<div class="container">
  <div class="row">
```

```
    <div class="col-md-3">
          <div  class="btn-group  btn-group-toggle"  data-
toggle="buttons">
        <label id="cycle" class="btn btn-primary active">
          <input type="radio" checked>Cycle
        </label>
        <label id="pause" class="btn btn-primary">
          <input type="radio">Pause
        </label>
      </div>
    </div>
    <div class="col-md-9">
      <div class="carousel slide">
        <div class="carousel-inner">
          <div class="carousel-item active">
            <img class="d-block w-100" src="images/t1.jpg" alt="Un
tigre">
          </div>
          <div class="carousel-item">
            <img class="d-block w-100" src="images/t2.jpg" alt="Un
tigre">
          </div>
          <div class="carousel-item">
            <img class="d-block w-100" src="images/t3.jpg" alt="Un
tigre">
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

<script
src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script src="js/bootstrap.js"></script>
<script>
  $(function () {
    $('.carousel').carousel({ interval: 2000 })
    $('#cycle').click(function() {
      $('.carousel').carousel('cycle')
    })
    $('#pause').click(function() {
      $('.carousel').carousel('pause')
```

```
    })
  })
</script>
```

Tester en ligne



Les deux boutons commandent le diaporama. Une bonne occasion pour revoir comment configurer des boutons de type « radio » avec Bootstrap. Remarquez aussi qu'on peut transmettre des options au carrousel lorsqu'on le commande comme je l'ai fait ici en affectant la valeur 2000 à la propriété **interval** pour avoir un défilement au rythme de 2 secondes.

Autres commandes

Voici maintenant un exemple plus complet avec une palette de boutons de commande :

```
<div class="container">
  <div class="row">
    <div class="col-md-12 pb-2">
      <div class="carousel slide">
        <div class="carousel-inner">
          <div class="carousel-item active">
            <img class="d-block w-100" src="images/t1.jpg" alt="Un
tigre">
          </div>
```

```
          <div class="carousel-item">
            <img class="d-block w-100" src="images/t2.jpg" alt="Un
tigre">
          </div>
          <div class="carousel-item">
            <img class="d-block w-100" src="images/t3.jpg" alt="Un
tigre">
          </div>
        </div>
      </div>
    </div>
    <div class="col-md-12 text-center">
          <div  class="btn-group  btn-group-toggle"  data-
toggle="buttons">
        <label id="first" class="btn btn-success">
             <input type="radio"><span class="fa fa-fast-
backward"></span>
        </label>
        <label id="previous" class="btn btn-success">
             <input type="radio"><span class="fa fa-step-
backward"></span>
        </label>
        <label id="pause" class="btn btn-success">
          <input type="radio"><span class="fa fa-pause"></span>
        </label>
        <label id="play" class="btn btn-success active">
            <input type="radio" checked><span class="fa fa-
play"></span>
        </label>
        <label id="next" class="btn btn-success">
             <input type="radio"><span class="fa fa-step-
forward"></span>
        </label>
        <label id="last" class="btn btn-success">
             <input type="radio"><span class="fa fa-fast-
forward"></span>
        </label>
      </div>
    </div>
  </div>
</div>

<script
```

```
src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script src="js/bootstrap.js"></script>
<script>
  $(function () {
    $('.carousel').carousel({ interval: 2000 })
    $('#first').click(function() { $('.carousel').carousel(0) })
                    $('#previous').click(function()     {
$('.carousel').carousel('prev') })
                       $('#pause').click(function()     {
$('.carousel').carousel('pause') })
    $('#play').click(function() { $('.carousel').carousel('cycle')
})
    $('#next').click(function() { $('.carousel').carousel('next')
})
                        $('#last').click(function()     {
$('.carousel').carousel($('.carousel-item').length - 1) })
  })
</script>
```

Tester en ligne

Les événements

Vous disposez de 2 événements pour ce plugin :

- **slide.bs.carousel** : se déclenche dès l'appel à la méthode **slide**
- **slid.bs.carousel** : se déclenche lorsque la translation est

terminée

Complétons l’exemple précédent en prévoyant l’affichage de l’index de la diapositive en cours ainsi que le nombre total de diapositives. Voici le code avec ce complément :

```
<div class="container">
  <div class="row">
    <div class="col-md-12 pb-2">
      <div class="carousel slide">
        <div class="carousel-inner">
          <div class="carousel-item active">
            <img class="d-block w-100" src="images/t1.jpg" alt="Un
tigre">
          </div>
          <div class="carousel-item">
            <img class="d-block w-100" src="images/t2.jpg" alt="Un
tigre">
          </div>
          <div class="carousel-item">
            <img class="d-block w-100" src="images/t3.jpg" alt="Un
tigre">
          </div>
        </div>
      </div>
    </div>
    <div class="col-md-12 text-center">
          <div class="btn-group btn-group-toggle" data-
toggle="buttons">
        <label id="first" class="btn btn-success">
             <input type="radio"><span class="fa fa-fast-
backward"></span>
        </label>
        <label id="previous" class="btn btn-success">
             <input type="radio"><span class="fa fa-step-
backward"></span>
        </label>
        <label id="pause" class="btn btn-success">
          <input type="radio"><span class="fa fa-pause"></span>
        </label>
        <label id="play" class="btn btn-success active">
            <input type="radio" checked><span class="fa fa-
play"></span>
```

```
        </label>
        <label id="next" class="btn btn-success">
             <input type="radio"><span class="fa fa-step-
forward"></span>
        </label>
        <label id="last" class="btn btn-success">
             <input type="radio"><span class="fa fa-fast-
forward"></span>
        </label>
      </div>
      <p><span class="badge badge-info"></span></p>
    </div>
  </div>
</div>

<script
src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script src="js/bootstrap.js"></script>
<script>
  var total;
  function affichage() {
    var current = $('.carousel-item.active').index() + 1
    $('.badge').text(current + ' / '+ total)
  }
  $(function () {
    total = $('.carousel-item').length
    affichage()
    $('.carousel').carousel({ interval: 2000 })
    $('#first').click(function() { $('.carousel').carousel(0) })
                    $('#previous').click(function()     {
$('.carousel').carousel('prev') })
                       $('#pause').click(function()     {
$('.carousel').carousel('pause') })
    $('#play').click(function() { $('.carousel').carousel('cycle')
})
    $('#next').click(function() { $('.carousel').carousel('next')
})
    $('#last').click(function() { $('.carousel').carousel(total -
1) })
    $('.carousel').on('slid.bs.carousel', function () {
      affichage()
    })
  })
```

```
</script>
```

Tester en ligne

Pour déterminer le nombre de diapositives, on compte le nombre de fois où on trouve la classe **carousel-item**. Pour la diapositive en cours, on recherche la classe **active**.

En résumé

- Le carrousel permet de faire défiler des images par substitution ou translation.
- On peut ajouter une légende par image.
- On peut ajouter des boutons de défilement et des indicateurs de l'image affichée.
- On peut régler la vitesse de défilement.
- Le plugin comporte des commandes utilisables avec jQuery et des événements qui permettent de commander efficacement l'affichage.

Bootstrap 4 : les onglets (Tab)

On a vu ce que nous propose Bootstrap pour la navigation. Dans ce chapitre nous allons utiliser les classes de bases déjà vues en ajoutant de l'interactivité pour créer des onglets.

Le principe

Ce plugin nécessite 2 groupes éléments :

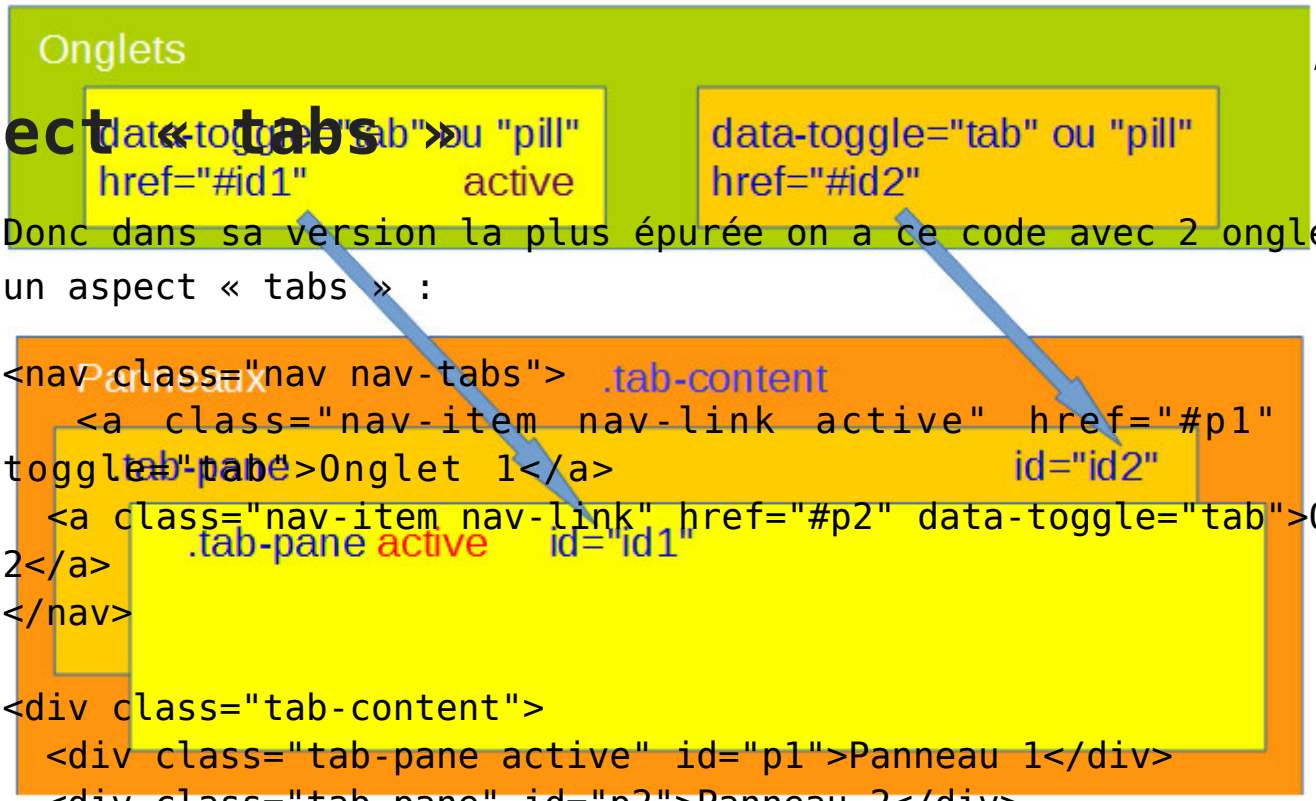
- des onglets avec chacun l’attribut **data-toggle= »tab** » ou **data-toggle= »pill** », selon l’aspect désiré, on ajoute la classe **active** à l’onglet actif au chargement,
- un groupe de panneaux avec la classe **tab-content**.

Pour chacun des panneaux on utilise la classe **tab-pane**, en prévoyant la classe **active** pour le panneau qui doit apparaître au chargement.

Il faut établir un lien entre chaque onglet et le panneau qui lui correspond. Cela se fait en attribuant à chaque panneau un identifiant référencé par un attribut **href** au niveau de l’onglet.

Ce qu’on peut schématiser ainsi :

Aspect « tabs »



Donc dans sa version la plus épurée on a ce code avec 2 onglets et un aspect « tabs » :

```
<nav class="nav nav-tabs">
  <a class="nav-item nav-link active" href="#p1" data-toggle="tab">Onglet 1</a>
  <a class="nav-item nav-link" href="#p2" data-toggle="tab">Onglet 2</a>
</nav>

<div class="tab-content">
  <div class="tab-pane active" id="p1">Panneau 1</div>
  <div class="tab-pane" id="p2">Panneau 2</div>
</div>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.0/umd/popper.min.js"></script>
<script src="js/bootstrap.min.js"></script>
```

*Le plugin fonctionne aussi avec des balises **ul** et **li** à la place de **nav**.*

Aspect « pills »

Voici le code avec aspect « pills » :

```
<nav class="nav nav-pills">
   <a class="nav-item nav-link active" href="#p1" data-toggle="tab">Onglet 1</a>
  <a class="nav-item nav-link" href="#p2" data-toggle="tab">Onglet 2</a>
</nav>

<div class="tab-content">
  <div class="tab-pane active" id="p1">Panneau 1</div>
  <div class="tab-pane" id="p2">Panneau 2</div>
</div>
```

Quelques effets

Transition progressive

Pour obtenir un effet progressif, il suffit d'ajouter la classe **fade** pour chacun des panneaux. D'autre part il faut prévoir la classe **show** pour le panneau qui doit s'afficher au chargement de manière à faire apparaître le panneau et avoir une animation à ce moment là également :

```
<nav class="nav nav-tabs">
   <a class="nav-item nav-link active" href="#p1" data-toggle="tab">Onglet 1</a>
  <a class="nav-item nav-link" href="#p2" data-toggle="tab">Onglet
```

```
2</a>
</nav>

<div class="tab-content">
  <div class="tab-pane fade show active" id="p1">Panneau 1</div>
  <div class="tab-pane fade" id="p2">Panneau 2</div>
</div>
```

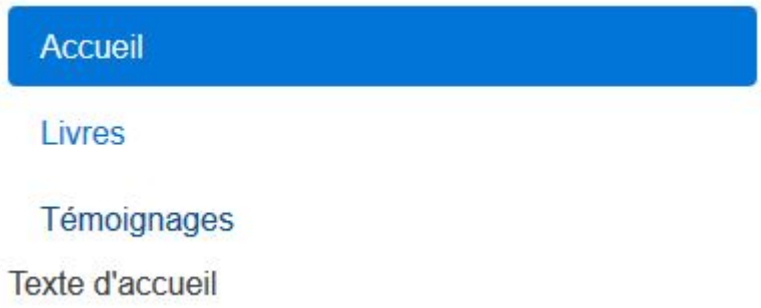
Tester en ligne

Empilage

Si la disposition en ligne ne vous convient pas vous pouvez empiler les onglets en utilisant **flexbox** avec la classe **flex-column** :

```
<nav class="nav nav-pills flex-column">
  <a class="nav-item nav-link active" href="#paccueil" data-toggle="tab">Accueil</a>
   <a class="nav-item nav-link" href="#livres" data-toggle="tab">Livres</a>
  <a class="nav-item nav-link" href="#temoignages" data-toggle="tab">Témoignages</a>
</nav>

<div class="tab-content">
      <div class="tab-pane active" id="accueil">Texte d'accueil</div>
   <div class="tab-pane" id="livres">Tous les livres</div>
      <div class="tab-pane" id="temoignages">Tous les témoignages</div>
</div>
```

Tester en ligne



Désactiver un onglet

Vous pouvez marquer un onglet comme inactif avec la classe **disabled** :

```
<nav class="nav nav-pills flex-column">
   <a class="nav-item nav-link active" href="#accueil" data-toggle="tab">Accueil</a>
    <a class="nav-item nav-link" href="#livres" data-toggle="tab">Livres</a>
   <a class="nav-item nav-link" href="#temoignages" data-toggle="tab">Témoignages</a>
</nav>

<div class="tab-content">
      <div class="tab-pane active" id="accueil">Texte d'accueil</div>
    <div class="tab-pane" id="livres">Tous les livres</div>
      <div class="tab-pane" id="temoignages">Tous les témoignages</div>
</div>
```

Menu déroulant

Pour avoir un menu déroulant dans un onglet, il faut utiliser le plugin **dropdown** que nous avons déjà vu :

```
<nav class="nav nav-tabs">
   <a class="nav-item nav-link active" href="#accueil" data-toggle="tab">Accueil</a>
  <div class="dropdown">
      <a class="nav-item nav-link dropdown-toggle" data-
```

```
toggle="dropdown" href="#">Livres</a>
    <div class="dropdown-menu">
        <a class="dropdown-item" href="#policiers" data-
toggle="tab">Policiers</a>
        <a class="dropdown-item" href="#romans" data-
toggle="tab">Romans</a>
        <a class="dropdown-item" href="#contes" data-
toggle="tab">Contes</a>
    </div>
  </div>
   <a class="nav-item nav-link" href="#temoignages" data-
toggle="tab">Témoignages</a>
</nav>

<div class="tab-content">
  <div class="tab-pane active" id="accueil">Texte d'accueil</div>
    <div class="tab-pane" id="temoignages">Tous les
témoignages</div>
   <div class="tab-pane" id="policiers">Tous les livres
policiers</div>
  <div class="tab-pane" id="romans">Tous les romans</div>
  <div class="tab-pane" id="contes">Tous les contes</div>
</div>
```

Tester en ligne

*Pour que le menu déroulant soit actif il faut prévoir la librairie **popper.js** (ou utiliser la version **bundle** de Bootstrap).*

On peut évidemment prévoir un menu déroulant avec l'aspect « pills » :

```
<nav class="nav nav-pills">
   <a class="nav-item nav-link active" href="#accueil" data-
toggle="tab">Accueil</a>
  <div class="dropdown">
```

```
      <a class="nav-item nav-link dropdown-toggle" data-toggle="dropdown" href="#">Livres</a>
    <div class="dropdown-menu">
        <a class="dropdown-item" href="#policiers" data-toggle="tab">Policiers</a>
         <a class="dropdown-item" href="#romans" data-toggle="tab">Romans</a>
         <a class="dropdown-item" href="#contes" data-toggle="tab">Contes</a>
    </div>
  </div>
   <a class="nav-item nav-link" href="#temoignages" data-toggle="tab">Témoignages</a>
</nav>

<div class="tab-content">
  <div class="tab-pane active" id="accueil">Texte d'accueil</div>
    <div class="tab-pane" id="temoignages">Tous les témoignages</div>
   <div class="tab-pane" id="policiers">Tous les livres policiers</div>
  <div class="tab-pane" id="romans">Tous les romans</div>
  <div class="tab-pane" id="contes">Tous les contes</div>
</div>
```

Tester en ligne

Accessibilité

Un groupement d'étiquettes équipées d'un mécanisme pour afficher des panneaux doit comporter les rôles **tablist** et **tab**. Il faut le compléter en repérant les panneaux avec le rôle **tabpanel**. D'autre part il faut compléter avec des attributs **aria-***.

Voici un exemple de mise en œuvre :

```
<nav class="nav nav-pills" role="tablist">
   <a class="nav-item nav-link active" id="accueil-tab" href="#accueil" data-toggle="tab" role="tab" aria-controls="accueil" aria-expanded="true">Accueil</a>
  <a class="nav-item nav-link" id="livres-tab" href="#livres" data-toggle="tab" role="tab" aria-controls="livres">Livres</a>
    <a class="nav-item nav-link" id="temoignages-tab" href="#temoignages" data-toggle="tab" role="tab" aria-controls="temoignages">Témoignages</a>
</nav>

<div class="tab-content">
     <div class="tab-pane active" id="accueil" role="tabpanel" aria-labelledby="accueil-tab">Texte d'accueil</div>
     <div class="tab-pane" id="livres" aria-labelledby="livres-tab">Tous les livres</div>
        <div class="tab-pane" id="temoignages" aria-labelledby="temoignages-tab">Tous les témoignages</div>
</div>
```

Activation avec Javascript

Un exemple

Nous avons ci-dessus activé le plugin avec la propriété **data-toggle= »tab** ». Il est aussi possible d'utiliser le plugin directement avec du Javascript. Voici un exemple sans la propriété renseignée :

```
<nav class="nav nav-tabs">
  <a class="nav-item nav-link active" href="#accueil">Accueil</a>
  <a class="nav-item nav-link" href="#livres">Livres</a>
  <a class="nav-item nav-link" href="#temoignages">Témoignages</a>
</nav>

<div class="tab-content">
       <div class="tab-pane active" id="accueil">Texte d'accueil</div>
    <div class="tab-pane" id="livres">Tous les livres</div>
```

```
      <div class="tab-pane" id="temoignages">Tous les témoignages</div>
</div>
```

Les onglets sont devenus inactifs. Pour que ça fonctionne il faut ajouter un peu de Javascript :

```
$('a').click(function (e) {
  e.preventDefault()
  $(this).tab('show')
})
```

Tous les onglets doivent être activés, c’est pour cette raison que j’ai choisi pour ma page le sélecteura qui est sans ambiguïté.

Les événements

Vous disposez également de 4 événements pour ce plugin :

- **show.bs.tab** : se déclenche dès l’appel à la méthode show avant que le panneau n’apparaisse
- **shown.bs.tab** : se déclenche lorsque le panneau devient visible
- **hide.bs.tab** : se déclenche lorsqu’un panneau va devenir invisible
- **hidden.bs.tab** : se déclenche lorsqu’un panneau est devenu invisible

D’autre part il est possible de connaître l’onglet actif avec **event.target** et l’onglet précédemment sélectionné avec **event.relatedTarget**. On va utiliser tout cela pour afficher le nom de l’onglet actuel et du précédent à chaque changement. On va donc ajouter le code HTML pour accueillir l’information :

```
<nav class="nav nav-tabs">
  <a class="nav-item nav-link active" href="#accueil">Accueil</a>
  <a class="nav-item nav-link" href="#livres">Livres</a>
  <a class="nav-item nav-link" href="#temoignages">Témoignages</a>
</nav>

<div class="tab-content">
      <div class="tab-pane active" id="accueil">Texte
```

```
d'accueil</div>
    <div class="tab-pane" id="livres">Tous les livres</div>
       <div class="tab-pane" id="temoignages">Tous les
témoignages</div>
</div>

<hr>
<p><strong>Onglet actif </strong>: <span id='actif'></span></p>
<p><strong>Onglet précédent </strong>: <span
id='precedent'></span></p>
```

Il ne reste plus qu'à prévoir le code Javascript pour déclencher le plugin et mettre en place l'écoute de l'événement et agir en conséquence :

```
$('a')
.click(function (e) {
  e.preventDefault()
  $(this).tab('show')
})
.on('shown.bs.tab', function (e) {
  $('#actif').text($(e.target).text())
  $('#precedent').text($(e.relatedTarget).text())
})
```

Tester en ligne

En résumé

- Les onglets permettent sur un même espace d'afficher au choix plusieurs contenus avec un simple clic.
- Les onglets sont faciles à organiser : horizontaux, empilés, simples liens ou boutons.
- On peut aussi intégrer un menu déroulant dans un onglet.

Bootstrap 4 : les boutons

On a déjà vu qu'on peut avec quelques classes régler facilement l'aspect des boutons et leur disposition. Dans ce chapitre nous allons voir comment les rendre visuellement dépendants d'un état.

Boutons bascule, « checkbox » et « radio »

Bouton bascule

On a parfois besoin de boutons à 2 états stabilisés : repos et appuyé. Le plugin permet de réaliser cela facilement. Il suffit de créer le bouton en prévoyant **data-toggle= »button »** :

```
<a class="btn btn-success" data-toggle="button">Simple Bascule</a>
```

Tester en ligne

Boutons effet « checkbox »

On peut grouper des boutons avec les classes **btn-group** et **btn-group-toggle** et les faire fonctionner comme des checkbox avec **data-toggle= »buttons »**. Les boutons restent indépendants dans leur fonctionnement mais ont l'avantage d'être groupés. Pour réaliser cela, il faut utiliser des contrôles **input** de type :

```
<div class="btn-group btn-group-toggle" data-toggle="buttons">
  <label class="btn btn-success active">
    <input type="checkbox" checked>Un
  </label>
  <label class="btn btn-success">
```

```
    <input type="checkbox">Deux
  </label>
  <label class="btn btn-success">
    <input type="checkbox">Trois
  </label>
</div>
```

Tester en ligne

Notez que si vous voulez qu'un bouton soit actif au chargement il faut prévoir :

- la classe **active** dans la balise **<label>**,
- **checked** dans la balise **<input>**.

Et vous pouvez évidemment en activer plusieurs puisque ce sont de bouton « checkbox ».

Boutons effet « radio »

On peut grouper des boutons avec les classes **btn-group** et **btn-group-toggle** et les faire fonctionner comme des boutons radio avec **data-toggle= »buttons** ». Les boutons sont maintenant alors liés dans leur fonctionnement, un seul peut être activé. Pour réaliser cela, il faut utiliser des contrôles **input** de type **radio** :

```
<div class="btn-group btn-group-toggle" data-toggle="buttons">
  <label class="btn btn-success active">
    <input type="radio" checked>Un
  </label>
  <label class="btn btn-success">
    <input type="radio">Deux
  </label>
  <label class="btn btn-success">
    <input type="radio">Trois
  </label>
</div>
```

Tester en ligne

C’est le même principe que vu ci-dessus pour l’activation d’un bouton au chargement (et évidemment un seul dans ce cas), mais cette fois ça s’impose parce que par principe dans un groupement « radio » il y a toujours un élément sélectionné.

Activation avec Javascript

Il n’y a pas trop d’intérêt d’activer les boutons avec Javascript. Par contre nous allons voir un cas assez courant d’utilisation dynamique d’un bouton qu’on peut facilement traiter avec jQuery.

Parfois on clique sur un bouton pour réaliser un processus qui demande un certain temps, comme par exemple un envoi de fichier.

D’abord on crée un bouton standard :

```
<button type="button" class="btn btn-primary">Cliquez !</button>
```

Ensuite on gère le clic :

```
$(function(){
  $('button').on('click', function() {
    var button = $(this)
    if(!button.hasClass('disabled')) {
      button.addClass('disabled').html('Chargement <span class="fa
fa-spinner fa-pulse"></span>');
      setTimeout(function () {
        button.removeClass('disabled').text('Cliquez !')
      }, 4000)
    }
  })
})
```

Donc lorsqu’on clique sur le bouton on le rend inactif, on ajoute une icône animée et on change le texte :

[Tester en ligne]

Et au bout du délai créé ici artificiellement on remet le bouton dans son état initial :

Maintenant voyons un cas d'utilisation un peu plus réaliste que notre minuterie. Nous avons une image lourde à charger, et nous voulons que le bouton change d'état le temps de chargement de l'image. Voilà le bouton et la balise prête à recevoir l'image :

```
<button type="button" class="btn btn-primary">Chargez l'image !</button>
<img id="mon_image">
```

Tester en ligne

Et voici le code Javascript pour gérer l'effet :

```
$(function(){
  $('button').on('click', function() {
    var button = $(this)
    if(!button.hasClass('disabled')) {
      button.addClass('disabled').html('Chargement en cours <span class="fa fa-spinner fa-pulse"></span>')
      var image = new Image()
      image.onload = function() {
        button.removeClass('disabled').text("Chargez l'image !")
        $("#mon_image").attr({ src:"images/legumes.jpg" })
      }
      image.src = "images/legumes.jpg"
    }
  })
})
```

Pendant le temps de chargement de l'image on a cet aspect :

Et quand l’image a fini de charger on retrouve le bouton normal et évidemment l’image :

En résumé

- On peut créer des boutons « bascule », ou avec effet « radio » ou « checkbox ».
- On peut gérer l’aspect d’un bouton avec jQuery par exemple pour attendre la fin d’un processus.

Bootstrap 4 : fenêtre modale (Modal)

Une fenêtre modale apparaît au-dessus de la page et est en général utilisée comme une boîte de dialogue avec l’utilisateur. Elle comporte généralement un formulaire. Nous allons voir dans ce chapitre comment créer et paramétrer une fenêtre modale.

Le principe

Ce plugin nécessite 2 éléments :

- un déclencheur avec l'attribut **data-toggle= »modal »**,
- un contenu avec la classe modal.

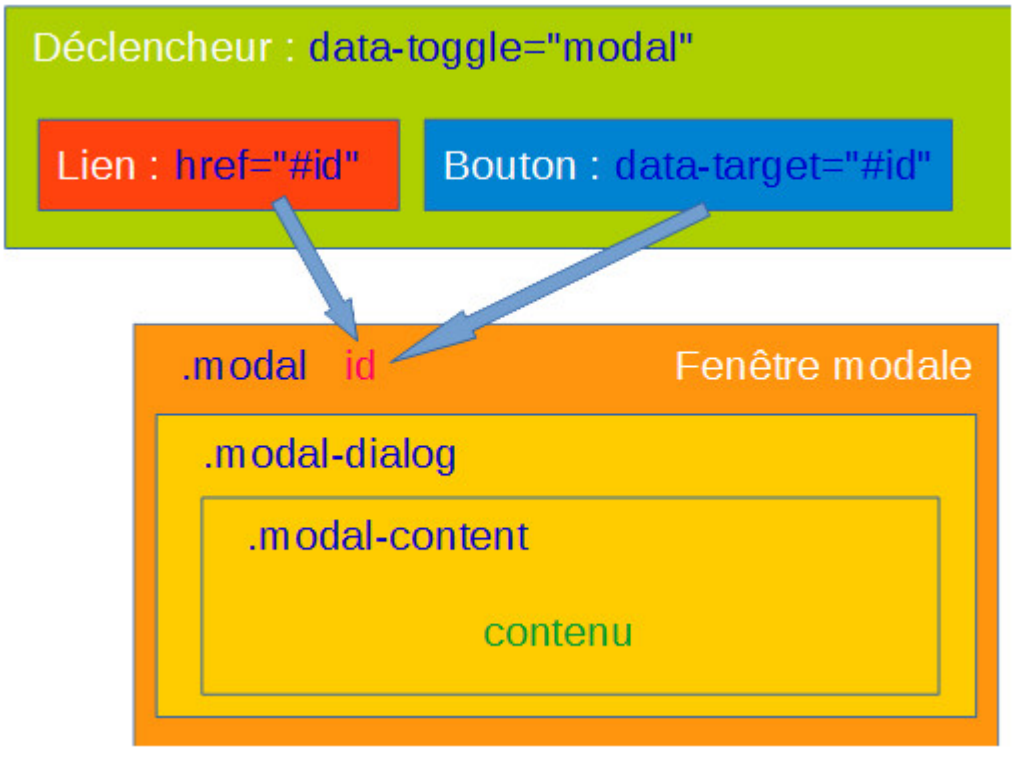
Mais comme il n'y a pas de contenant il faut établir un lien entre le déclencheur et le contenu. Cela se fait en attribuant au contenu un identifiant qui sera référencé par le déclencheur avec deux cas :

- **un lien** : la référence se fait avec un attribut **href**,
- **un bouton** : la référence se fait avec un attribut **data-target**.

Mais il nous faut aussi des classes pour styliser la fenêtre modale avec une largeur, un positionnement, un fond, une bordure… Pour réaliser tout ça on va utiliser deux classes :

- **modal-dialog** : fixe la position, la largeur et les marges,
- **modal-content** : fixe le fond et les bordures.

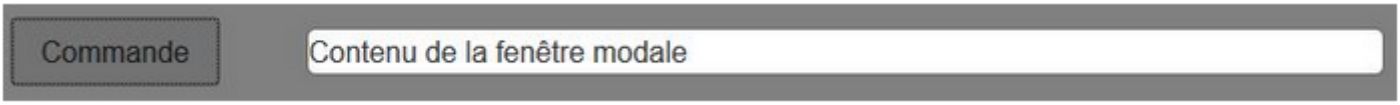
Ce qu'on peut schématiser ainsi :



Donc dans sa version la plus épurée on a ce code avec un bouton comme

déclencheur :

```
<button type="button" data-toggle="modal" data-target="#infos"
class="btn btn-secondary">Commande</button>
<div class="modal" id="infos">
  <div class="modal-dialog">
    <div class="modal-content">
      Contenu de la fenêtre modale
    </div>
  </div>
</div>
```



Entête, corps et pied

Maintenant que nous avons vu le principe de fonctionnement prenons un exemple avec :

- une entête avec la classe **modal-header**, avec possibilité d'un titre avec la classe **modal-title**,
- un corps avec la classe **modal-body**,
- un pied de page avec la classe **modal-footer**.

```
<button type="button" data-toggle="modal" data-target="#infos"
class="btn btn-primary">Informations</button>
<div class="modal" id="infos">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h4 class="modal-title">Plus d'informations</h4>
      </div>
      <div class="modal-body">
         Le Tigre (Panthera tigris) est un mammifère carnivore de
la famille des félidés...
      </div>
      <div class="modal-footer">
        <em>Informations sous réserve</em>
      </div>
    </div>
  </div>
```

```
</div>
```

Tester en ligne

Ce qui peut se schématiser ainsi :

Commande de fermeture

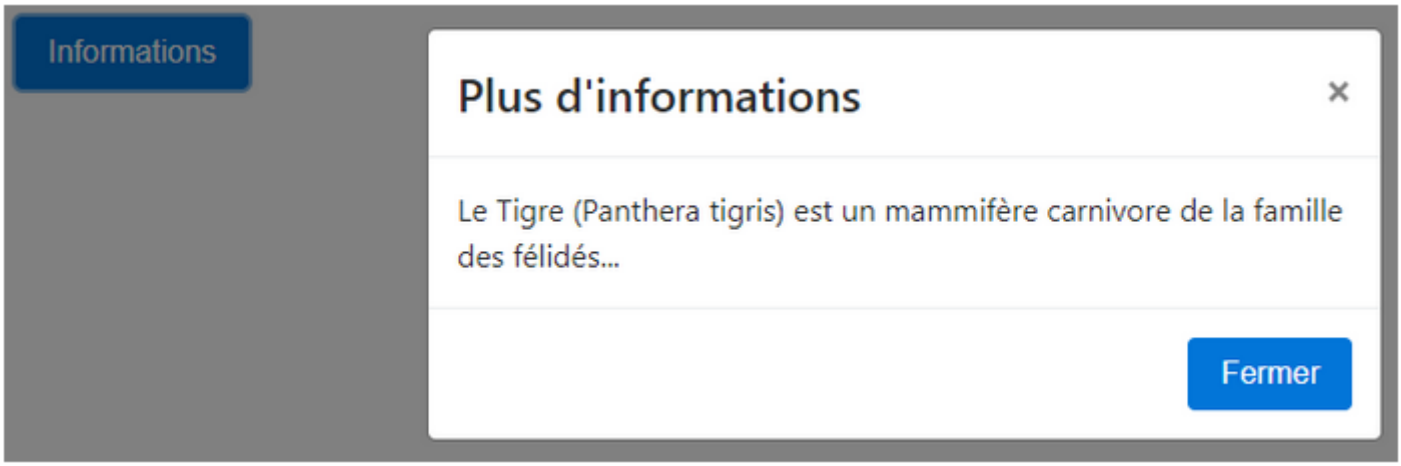
Dans les exemples vus ci-dessus pour fermer la fenêtre modale il faut cliquer n'importe où sur la page (en dehors de la fenêtre). Ce n'est pas vraiment très pratique. Il serait bien d'avoir un bouton dans la fenêtre pour assurer sa fermeture. Pour le réaliser on a l'attribut **data-dismiss= »modal** ». Classiquement on prévoit un petit bouton en forme de croix dans l'entête ou/et un bouton en pied de fenêtre.

Voici l'exemple ci-dessus complété avec des commandes de fermeture

:

```
<button type="button" data-toggle="modal" data-target="#infos"
class="btn btn-primary">Informations</button>
<div class="modal" id="infos">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h4 class="modal-title">Plus d'informations</h4>
        <button type="button" class="close" data-dismiss="modal">
          <span>&times;</span>
        </button>
      </div>
      <div class="modal-body">
         Le Tigre (Panthera tigris) est un mammifère carnivore de
la famille des félidés...
      </div>
      <div class="modal-footer">
          <button type="button" class="btn btn-primary" data-
dismiss="modal">Fermer</button>
      </div>
    </div>
  </div>
</div>
```

Tester en ligne



La classe **close** a pour but de styliser la petite croix et doit être prévue.

Dimension de la fenêtre

Il peut arriver de désirer une fenêtre plus grande ou plus étroite. On dispose de deux classes pour le réaliser :

- **modal-lg** : pour une fenêtre large
- **modal-sm** : pour une fenêtre étroite

Ces classes doivent aller avec la classe **modal-dialog**. Voici un exemple :

```
<button class="btn btn-primary" data-toggle="modal" data-target="#f1">Grande fenêtre</button>
<div class="modal" id="f1">
  <div class="modal-dialog modal-lg">
    <div class="modal-content">
      <div class="modal-body">
        Je suis une grande fenêtre !
        <button type="button" class="close" data-dismiss="modal">
          <span>&times;</span>
        </button>
      </div>
    </div>
  </div>
</div>
<button class="btn btn-primary" data-toggle="modal" data-target="#f2">Petite fenêtre</button>
<div class="modal" id="f2">
  <div class="modal-dialog modal-sm">
    <div class="modal-content">
      <div class="modal-body">
        Je suis une petite fenêtre !
        <button type="button" class="close" data-dismiss="modal">
          <span>&times;</span>
        </button>
      </div>
    </div>
  </div>
</div>
```

Grande fenêtre avec la classe **modal-lg** :

Petite fenêtre avec la classe **modal-sm** :

Animation fluide et fond d'écran

Animation fluide

Dans les exemples vus ci-dessus la fenêtre modale apparaît d'un coup à l'écran. Il est possible de la faire apparaître avec un mouvement fluide. Pour cela il suffit d'ajouter la classe **fade** à la classe **modal**.

Prenons un exemple précédent en ajoutant cette classe :

```
<button type="button" data-toggle="modal" data-target="#infos" class="btn btn-primary">Informations</button>
<div class="modal fade" id="infos">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal">
          <span>&times;</span>
        </button>
        <h4 class="modal-title">Plus d'informations</h4>
      </div>
      <div class="modal-body">
         Le Tigre (Panthera tigris) est un mammifère carnivore de la famille des félidés...
      </div>
      <div class="modal-footer">
          <button type="button" class="btn btn-primary" data-dismiss="modal">Fermer</button>
      </div>
```

```
    </div>
  </div>
</div>
```

Tester en ligne

Fond d’écran

Par défaut le fond de l’écran se grise lorsque la fenêtre modale apparaît et un clic dans cette zone grisée a pour effet de fermer la fenêtre. Si vous désirez conserver une apparence normale il faut utiliser l’attribut **data-backdrop= »false** ». Par exemple pour le cas vu ci-dessus il suffit de modifier cette partie du code du bouton déclencheur :

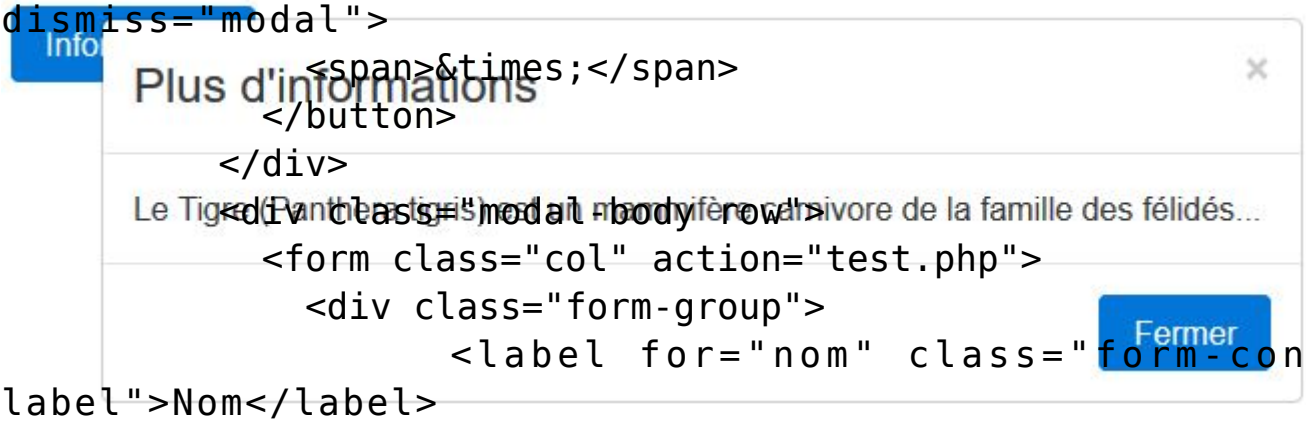
```
<button type="button" data-toggle="modal" data-backdrop="false" data-target="#infos" class="btn btn-primary">Informations</button>
```

Tester en ligne

Un formulaire dans la fenêtre modale

Il est tout à fait possible d'avoir des éléments interactifs au niveau de la fenêtre modale comme par exemple un formulaire. Voici un exemple de mise en œuvre :

```
<div class="container">

    <div id="html">
        <button data-toggle="modal" data-target="#formulaire" class="btn btn-primary">Informations</button>
    </div>
    <div class="modal fade" id="formulaire">
      <div class="modal-dialog">
        <div class="modal-content">
          <div class="modal-header">
            <h4 class="modal-title">Vos infos :</h4>
                <button type="button" class="close" data-dismiss="modal">
                <span>&times;</span>
            </button>
          </div>
          <div class="modal-body row">
            <form class="col" action="test.php">
              <div class="form-group">
                    <label for="nom" class="form-control-label">Nom</label>
                  <input type="text" class="form-control" name="nom" id="nom" placeholder="Votre nom">
              </div>
              <div class="form-group">
```



```
                    <label for="email" class="form-control-
label">Email</label>
                    <input type="email" class="form-control"
name="email" id="email" placeholder="Votre Email">
                </div>
                 <button type="submit" class="btn btn-primary pull-
right">Envoyer</button>
            </form>
          </div>
        </div>
      </div>
    </div>

</div>

<script
src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.0/umd/p
opper.min.js"></script>
<script src="js/bootstrap.min.js"></script>
<script>
  $(function(){
    $('form').submit(function(e) {
      e.preventDefault()
      var $form = $(this)
      $.post($form.attr('action'), $form.serialize())
      .done(function(data) {
        $('#html').html(data)
        $('#formulaire').modal('hide')
      })
      .fail(function() {
        alert('ça ne marche pas...')
      })
    })
    $('.modal').on('shown.bs.modal', function(){
      $('input:first').focus()
    })
  })
</script>
```

Le traitement du formulaire va se faire en Ajax, de cette façon il est facile de gérer la validation (ce que je ne fais pas dans cet exemple pour ne pas alourdir le code). Il nous faut un fichier PHP (test.php) pour gérer le traitement du formulaire :

```
<?php
echo("Le nom est " . $_POST['nom'] . " et l'Email est " . $_POST['email'] . ".");
```

Je me contente de mettre en forme les deux informations transmises et de renvoyer le tout. Ensuite j'affiche la phrase sur la page après avoir effacé la fenêtre modale (on va voir un peu plus loin dans ce chapitre comment on commande la fenêtre modale avec jQuery) :

Tester en ligne

Le nom est Dupont et l'Email est dupont@chezmoi.fr.

En fait vous pouvez mettre tout ce que vous voulez dans une fenêtre modale !

Accessibilité

Comme la fenêtre modale est une boîte de dialogue séparée du reste de la page et qui apparaît au-dessus il faut

prévoir **role= »dialog** ». Mais ce rôle n'est pas suffisant. Il faut aussi prévoir une étiquette de référence. Autrement dit répondre à la question « A quoi se rapporte cette boîte de dialogue ? ». L'idéal est d'utiliser un attribut **aria-labelledby** pour l'élément qui a le rôle **dialog**. On peut alors se référer par exemple à un titre dans la fenêtre.

Le contenu de la fenêtre lui-même se présente comme un document à consulter il faut prévoir **role= »document** ». Mais tout dépend évidemment ce que vous placez dans la fenêtre modale. Si vous prévoyez d'y mettre un formulaire alors le rôle change parce qu'il y a une interactivité.

Voici un exemple simple de réalisation :

```
<button type="button" data-toggle="modal" data-target="#infos" class="btn btn-secondary">Commande</button>
<div class="modal" id="infos" role="dialog" aria-labelledby="label" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
                <h4 class="modal-title" id="label">Plus d'informations</h4>
      </div>
      <div class="modal-body">
        Le Tigre (Panthera tigris) est un mammifère carnivore de la famille des félidés...
      </div>
    </div>
  </div>
</div>
```

Activation avec Javascript

Ouverture

On a vu qu'on peut pratiquement tout faire avec des propriétés. Ainsi la propriété **data-toggle= »modal** » placée dans l'élément déclencheur active automatiquement le plugin. Mais vous

avez aussi la possibilité d'utiliser Javascript pour utiliser ce plugin. Reprenons un exemple vu ci-dessus en n'utilisant plus cette propriété :

```
<button type="button" class="btn btn-primary">Informations</button>
<div class="modal" id="infos">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h4 class="modal-title">Plus d'informations</h4>
        <button type="button" class="close" data-dismiss="modal">
          <span>&times;</span>
        </button>
      </div>
      <div class="modal-body">
        Le Tigre (Panthera tigris) est un mammifère carnivore de la famille des félidés...
      </div>
      <div class="modal-footer">
          <button type="button" class="btn btn-primary" data-dismiss="modal">Fermer</button>
      </div>
    </div>
  </div>
</div>
```

Maintenant lorsqu'on clique sur le bouton, il ne se passe plus rien. Pour activer le plugin, il faut utiliser jQuery :

```
$('.btn').click(function() {
  $('.modal').modal('show')
})
```

Fermeture

De la même manière on a vu que la propriété **data-dismiss= »modal** » permet d'automatiser la fermeture. On peut utiliser jQuery pour accomplir cette action. Voilà le même exemple avec la fermeture effectuée par jQuery pour les deux boutons en plus de l'ouverture :

```
<div class="container">
```

```
    <button id="open" type="button" class="btn btn-
primary">Informations</button>
  <div class="modal" id="infos">
    <div class="modal-dialog">
      <div class="modal-content">
        <div class="modal-header">
          <h4 class="modal-title">Plus d'informations</h4>
          <button type="button" class="close closemodal">
            <span>&times;</span>
          </button>
        </div>
        <div class="modal-body">
          Le Tigre (Panthera tigris) est un mammifère carnivore de
la famille des félidés...
        </div>
        <div class="modal-footer">
             <button type="button" class="btn btn-primary
closemodal">Fermer</button>
        </div>
      </div>
    </div>
  </div>

</div>

<script
src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script src="js/bootstrap.js"></script>
<script>
  $(function(){
    $('#open').click(function() {
      $('.modal').modal('show')
    })
    $('.closemodal').click(function() {
      $('.modal').modal('hide')
    })
  })
</script>
```

Les événements

Vous disposez également de 4 événements pour ce plugin :

- **show.bs.modal** : se déclenche dès l'appel à la méthode show. Si c'est suite à un événement clic l'élément cliqué peut être connu avec la propriété **relatedTarget**
- **shown.bs.modal** : se déclenche lorsque la fenêtre devient visible. Si c'est suite à un événement clic l'élément cliqué peut être connu avec la propriété **relatedTarget**
- **hide.bs.modal** : se déclenche dès l'appel à la méthode **hide**.
- **hidden.bs.modal** : se déclenche lorsque la fenêtre est masquée

Vous avez par exemple un formulaire sur la fenêtre modale comme on l'a vu ci-dessus et vous désirez que le premier contrôle soit actif à l'affichage de la page, vous pouvez le réaliser facilement ainsi :

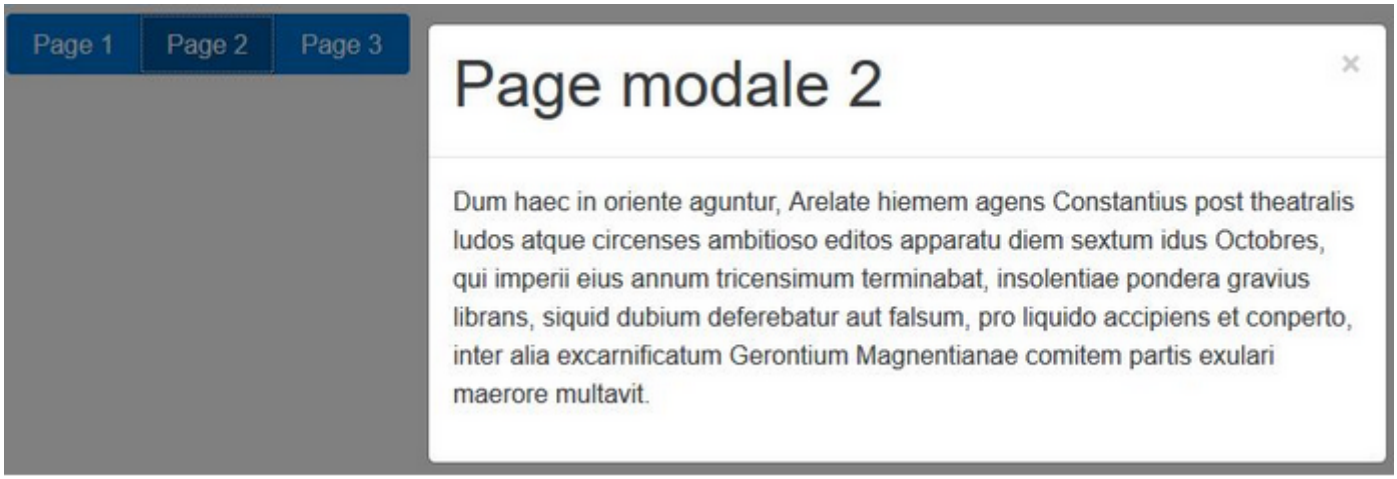
```
$('.modal').on('shown.bs.modal', function(){
  $('input:first').focus()
})
```

Une autre utilisation pratique de ces événements consiste à se servir de la propriété **relatedTarget**. Supposez que vous ayez 3 boutons pour ouvrir une fenêtre modale et que pour chaque bouton le contenu de la fenêtre doit être différent, par exemple on charge une page html distincte. On pourrait évidemment utiliser 3 fenêtres totalement séparées mais on peut faire quelque chose de plus élégant :

```
<div class="container">

  <div class="btn-group">
     <button id="page1" type="button" data-toggle="modal" data-target=".modal"  data-remote="1"  class="btn  btn-primary">Page 1</button>
     <button id="page2" type="button" data-toggle="modal" data-target=".modal"  data-remote="2"  class="btn  btn-primary">Page 2</button>
     <button id="page3" type="button" data-toggle="modal" data-target=".modal"  data-remote="3"  class="btn  btn-primary">Page 3</button>
  </div>
  <div class="modal fade" id="infos">
    <div class="modal-dialog">
```

```
      <div class="modal-content"></div>
    </div>
  </div>

</div>

<script
src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script src="js/bootstrap.js"></script>
<script>
  $(function(){
    $('.modal').on('show.bs.modal', function (e) {
      var button = $(e.relatedTarget)
      var index = button.data('remote')
        $(this).find('.modal-content').load('remote' + index +
'.html')
    })
  })
</script>
```

Tester en ligne



J’ai créé un attribut **data-remote** avec les valeurs 1, 2 et 3. Lorsque l’événement **show.bs.modal** est déclenché, autrement dit dès qu’on clique sur un bouton, on peut récupérer la valeur de la propriété **data-remote** du bouton cliqué et l’utiliser pour composer le nom de la page HTML à charger.

En résumé

- Les fenêtres modales permettent de faire apparaître des informations.
- Elles peuvent comporter une entête avec éventuellement un titre, un contenu et un bas de page.
- On peut adapter la largeur et la visibilité de la page en fond.
- On peut mettre n’importe quel contenu dans une fenêtre modale, par exemple un formulaire.
- jQuery permet de créer une meilleure interactivité par exemple en mettant le focus sur un contrôle de formulaire ou en chargeant une page HTML dans la fenêtre selon le bouton cliqué.