Programmer pour le client web : jQuery

jQuery

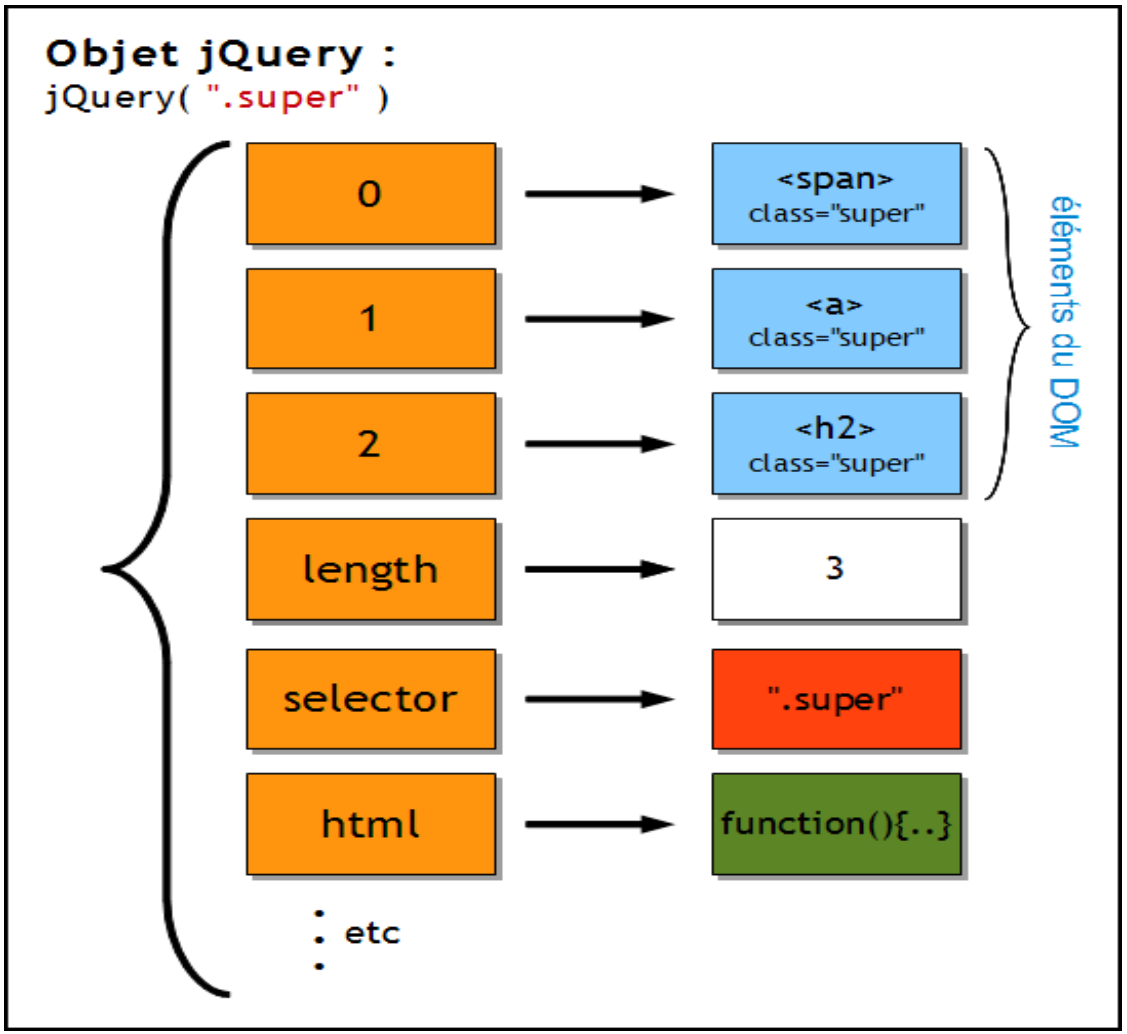
- bibliothèque javascript **cross-browser**
- nombreuses fonctionnalités pour manipuler le dom
- des plugins pour des effets tout prêts
  - carrousel photos, auto-complétion, calendriers ...
- autres outils du même type : Dojo, MooTools , YUI ...

principes de base

- la ***fonction*** jQuery (*aka* $) :
  - en tant qu'objet, fournit des méthodes et des propriétés utilitaires
  - jQuery.method() ; jQuery.property
  - $.method() ; $.property
  - **construit des objets jQuery, en sélectionnant des objets du Dom**
- les ***objets*** jQuery() construits par la fonction **$()**
  - contiennent des méthodes et propriétés pour manipuler le Dom
  - jQuery(*selector, [context]*).**method**() ; jQuery().**property**
  - $(*selector, [context]*).**method**() ; $().**property**

les Objets jQuery

- un objet jQuery encapsule 1 ou plusieurs objets du DOM et des propriétés et méthodes spécifiques
  - accès à un noeud DOM
    - $()[0], $()[1] …
  - $('.super').length
  -
- un objet jQuery N'EST PAS un objet DOM



source: http://www.siteduzero.com/informatique/tutoriels/jquery-ecrivez-moins-pour-faire-plus/jquery-et-le-document

$() : construction d'objets jQuery

- retourne un ou plusieurs objet(s) jQuery :
  - par sélection dans le DOM : **`$( selector )`**
  - par transformation d'un nœud du DOM
  - par création d'un sous arbre à partir d'une string html

```
// sélections dans le DOM :

let foo=  $('#foo')
let totos=$('.toto')

// transformation d'un objet du DOM en objet $
let n = document.getElementById("main")
                .parentNode ;
let n$ = $(n) ;          // transformation de n en 1 objet $

// création d'objets $ à partir d'une string
let p$ = $('<p>') ;
let di = $('<div class="main"><p>paragraphe 23</p></div>') ;
```

Sélection dans le DOM : les sélecteurs

- tous les sélecteurs css
  - `* , elem, .class, #id, elem[attr], elem[attr="val"], elem1>elem2, elem1 elem2, elem1+elem2`

- des sélecteurs spécifiques, peuvent être utilisés seuls ou accolés à un sélecteur css
  - `:hidden` : éléments non visibles : `$('div:hidden')`
  - `:parent` : éléments ayant des enfants : `$('p:parent')`
  - **spécial formulaire** : `:button, :checkbox, :checked, :input, :radio, :submit`

exemples

- **`$( 'div.foo' )`** : retourne un objet jQuery contenant tous les élements de classe "foo"
- **`$( '#bar' )`** : retourne un objet jQuery contenant l'élément portant l'id="bar"
- **`$( 'ul.menu > li' )`** : retourne un objet Jquery contenant tous les élements <li> enfants directs d'un <ul> de classe "menu"
- **`$('p.bar:parent')`** : retourne un objet Jquery contenant tous les élements <p> de classe "bar" ayant au moins un fils
- **`$('#form1 :checked)`** : retourne un objet Jquery contenant tous les élements "checked" (radio / checkbox)

parents, enfants et voisins d'un élément

- toutes les méthodes acceptent un sélecteur optionnel qui permet de filtrer le résultat
- **Parents :** .parent(), .parents(), .closest()
  - $("#myelem").**parent**() : le parent direct
  - $("#myelem").**parents('div')** : les parents filtrés par 'div'
  - $("#myelem").**closest('section')**: le + proche parent 'section'
- **Enfants :** .children(), .find()
  - $("#myelem").**children**() : les enfants directs
  - $("#myelem").**find('p.foo')** : les 'p.foo' dans le sous-arbre
- **Voisins :** .next(), .prev(), .siblings()
  - $("#myelem").**next**() : le voisin suivant
  - $("#myelem").**siblings**() : tous les voisins

les méthodes des objets jQuery

- agissent sur les éléments du DOM
- suivant les méthodes :
  - agissent sur tous les noeuds contenus dans l'objet
  - agissent sur le 1er noeud

accéder aux attributs

- **$().attr() : getter, setter –** reçoit un nom d'attribut et une valeur optionnelle
  - $('img').**attr('alt')** : retourne la valeur de "alt" de la 1ere image
  - $('img').**attr('alt', 'une belle image')**
  - peut recevoir une fn retournant la valeur :
    - $('img').**attr('alt', function(index, oldVal) { … })** ;

- **$().attr(jsonAttrs) : setter**
  - positionne plusieurs attributs du 1er noeud
  - $('img').attr({ alt : 'jolie photo' , title : 'photo gc',
    src : 'img0078.png'}) ;

Modifier l'apparence

- On change l'apparence d'un élément en changeant sa classe :
  - prévoir une classe pour chaque apparence d'un élément dans le code css
- utiliser les méthodes de manipulation de classes :
  - $().**addClass**('class') : ajouter 1 classe
  - $().**removeClass**('class') : retirer 1 classe
  - $().**toggleClass**('class') : ajouter/retirer une classe
  - $().**hasClass**('class') : test

quelques effets simples

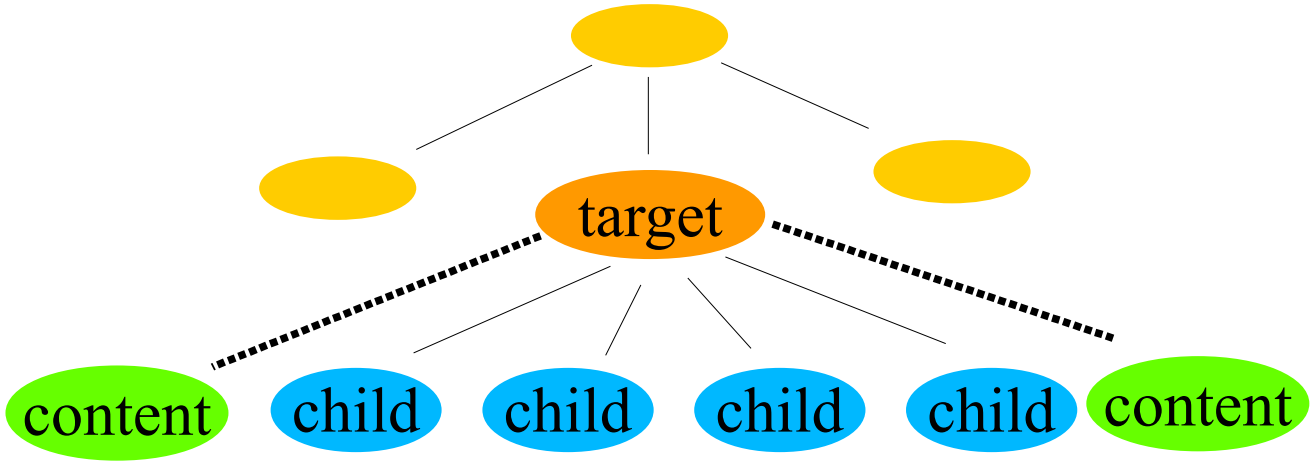
- méthodes s'appliquant à l'ensemble des éléments de l'objet $(), reçoivent en paramètre la durée de l'effet, en ms ou '*slow*' (600ms), '*fast*' (200ms)
- **Affichage/masquage**
  - $().**hide(d)** , $().**show(d)**, $().toggle(d)
  - $('a#bar').**toggle('slow')** : affiche/masque si el. caché/visible
- **fading : apparition progressive**
  - $().fadeIn(d), $().fadeOut(d), $().fadeToggle(d)
  - fadeTo(d,o) : ajuste l'opacité de l'elt. à la valeur fournie
- **slide : apparition glissée**
  - $().slideUp(d), $().slideDown(d), $().slideToggle(d)

manipuler l'arbre dom avec jQuery

- **créer** des éléments du dom :
  - **$("<*elem*>")** : retourne 1 objet jQuery contenant 1 nouveau noeud correspondant à *elem*
  - **$("*html")*** : crée un sous-arbre DOM correspondant au texte html transmis
- **accéder** au contenu html/text d'un noeud
  - $().**html**(*["html"]*) : getter, setter : retourne le contenu html du 1er élément sélectionné **ou** positionne le contenu html de tous les éléments sélectionnés si *["html"]* est présent
  - $().**text**(*["text"]*) : getter, setter : retourne le contenu TEXT du 1er élément sélectionné, ou positionne le contenu TEXT de tous les éléments sélectionnés si *["text"]* est présent

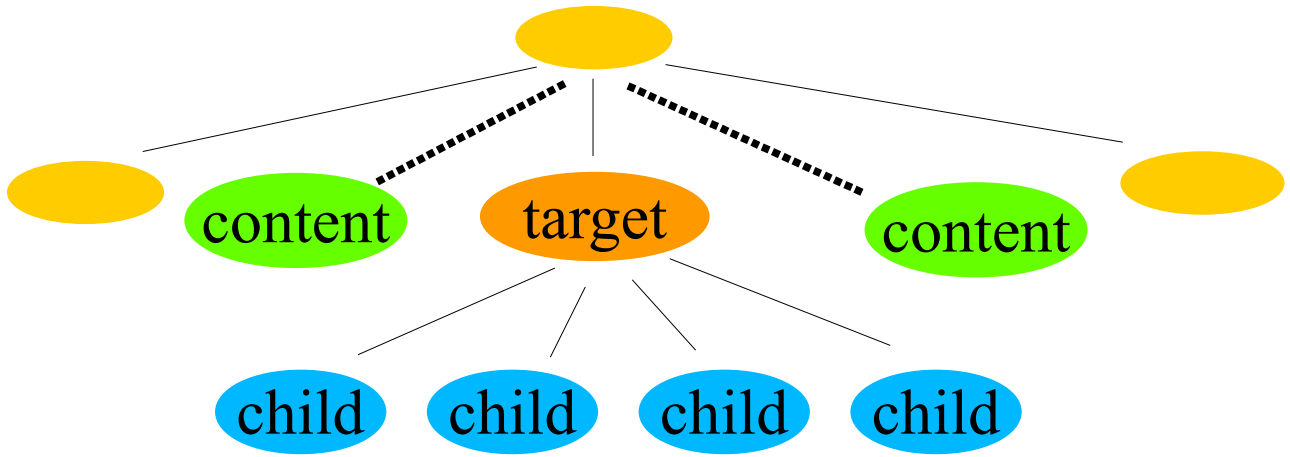
insérer dans le dom

- **jQuery** propose des méthodes d'insertion dans le DOM, permettant d'insérer un contenu par rapport à une cible



- **insertion *interne*** à l'intérieur de la cible
  - $(*content*).**appendTo**(***target***) : insère les éléments sélectionnés en fin de la cible en paramètre
  - $(***target***).**append**(*content*) : insère le contenu en paramètre en fin de tous les elts sélectionnés
  - $(*content*).**prependTo**(***target***) : insère les éléments sélectionnés en début de la cible en paramètre
  - $(***target***).**prepend**(*content*) : insère le contenu en paramètre en début de tous les elts sélectionnés
  - *content*, *target* : sélecteur, html, dom, jQuery

- **insertion *externe*** à l'extérieur de la cible:
  - $(*target*).**after**(*content*) : insère content après chaque élément sélectionné
  - $(*content*).**insertAfter**(*target*):insère les éléments sélectionnés après la cible
  - $(*target*).**before**(*content*) : insère content avant chaque élément sélectionné
  - $(*content*).**insertBefore**(*target*):insère les éléments sélectionnés avant la cible



- **suppressions :**
  - .**remove**() : supprime les éléments sélectionnés
  - .**empty**() : supprime les *fils* des éléments sélectionnés
  - .**unwrap**() : supprime les *parents* des éléments sélectionnés
- **remplacements**
  - .**replaceWith**() : remplace les éléments sélectionnés par le contenu fourni en paramètre
  - .**replaceAll**() : remplace chaque élément en paramètre par l'ensemble sélectionné
- **clonage**
  - .**clone**() : crée une copie de l'ensemble des éléments sélectionnés

parcourir une collection

- Pour parcourir tous les éléments d'un objet jQuery :
- $().**each**(*callback*), $().**map**(*callback*)
- **callback** est une fonction qui reçoit l'index de l'élt courant et l'élément lui-même.
- exemple : appliquer 1 classe à tous les voisins suivants de certains éléments

```
$('div > .foo).each((i,e) => {
                        $(e).next().addClass("bar") ;
                } ) ;
```

événements

- forme générale : **$().on('evt', data, handler(event))** :
  - lorsque l'événement ***'evt'*** est détecté sur 1 élément contenu dans $(), la fonction ***'handler'*** est appelée. Elle reçoit un objet <*event*> contenant notamment les données <*evtData*>
  - <*evt*> :`'click submit mouseover mousedown mouseup mouseenter mouseleave keypress keydown keyup  blur change focus select ..'`
  - <*data*> : données json, accessibles dans l'objet event : `event.data.`*`att`*
  - *handler(event)* : fonction appelée, reçoit en paramètre un objet event

exemples

```
$("p").on("click", (e) => {
                alert( e.target.text() ); } ) ;
```

```
$("p").on("click", {foo: "bar"}, (e) =>  {
                alert( e.data.foo ) ; } ) ;
```

```
$("form").on("submit", false)
```

raccourcis

- pour tous les événements js :
  - $().evt(handler) <=> $().on(evt, data, handler)
  - $('a.foo').**mouseover**(function() { … }) ;

- **$(document).ready(function() { … }**) : appelle la fonction lorsque le **DOM est complètement chargé** – valable uniquement sur $(document)
  - forme raccourcie : $(function() { … }) ;
  - moyen standard pour initialiser les gestionnaires d'evt.

exemple

```
$(document).ready( function() {

    // définir les composants de l'application
    // pattern module
     var jq1 = { modules : {}  } ;

     jq1.modules.menu = ( function(){
            return {
              init : function() {
                 ...
            } ;
      }) ();

     // faire des initialisations
      jq1.modules.menu.init() ;

     // enregistrer des handlers sur les evt
      $('#app').on('click',jq1.modules.menu.show ) ;

}) ;
```

utiliser jQuery

- download :
  - version compressée : en production
  - version non compressée : en développement

```
<script src="js/jquery-3.3.1.js"><script>
```

- CDN :
  - cdn jQuery : http://code.jquery.com
  - cdn google : https://ajax.googleapis.com/ajax/libs

```
<script src="https://code.jquery.com/jquery-3.3.1.js"></script>
```

- combiner les 2 :

```
<script src="https://code.jquery.com/jquery-3.3.1.js"></script>
<script>window.jQuery ||
document.write('<script src="js/jquery-3.3.1.js"><\/script>')
</script>
```