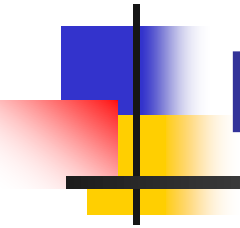
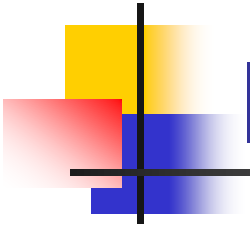


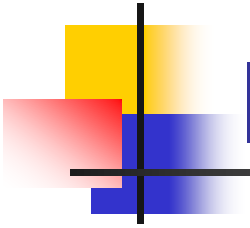
Langages côté Client

JavaScript - jQuery



Introduction

- JavaScript
  - Langage de script incorporé dans le HTML
  - Historiquement, premier langage de script pour le Web
  - Apporte des améliorations au HTML
    - HTML permet d'écrire
    - JavaScript permet de programmer, c'est-à-dire de gérer l'information
- Qualités :
  - Disponible sur les navigateurs actuels et gratuit
- Défauts :
  - Interprété et donc très lent, pas de débogueur



Introduction

- A quoi ressemble un script ?
  - C’est une portion de code qui vient s'insérer dans une page HTML
  - Le code du script n'est toutefois pas visible dans la fenêtre du navigateur car il est compris entre des balises (ou tags) spécifiques qui signalent au navigateur qu'il s'agit d'un script écrit en langage JavaScript
  - Balises annonçant le code Javascript :

```
<SCRIPT language="Javascript">
        Placez ici le code de votre script
</SCRIPT>
```

Variables

Déclaration

```
<script language="JavaScript">
    var date; // Déclaration sans affectation
    var compteur=0; // Déclaration avec affectation
    toto='coucou'; // Déclaration implicite par affectation
    var prem, second; // variables séparées par des virgules
    monNombre = new Number(); // Déclaration typée sans affectation
    e = new Number(2.71828); // Déclaration typée avec affectation
    var maChaine = new String(); //Déclaration de chaîne
    var toto = new Boolean(true); //Déclaration de booléen
</script>
```

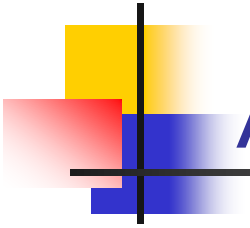
Tableau

- Array
  - Le type des éléments : nombres, chaînes, booléens, ...
  - La dimension 1, 2, ou 3, ... : tab(7) ; tab(x,y) ; tab(A,B,C) ; ...
  - Les indices : souvent des nombres entiers

```
<script language=JavaScript>
    // Tableau de chaînes, de dimension 1, indicé de 0 à 6 :
    Jour=new Array(7);
    Jour[0]='Dimanche' ;
    Jour[1]='Lundi' ;
    Jour[2]='Mardi' ; //...
    Jour[6]='Samedi' ;
    // En énumérant les éléments :
     jour=new Array('dimanche','lundi','mardi', ...
    ,'vendredi','samedi');
</script>
```

Exemple

```
temp=new Array(30);
    function relever() {
      for (i=1;i<=30;i++) {
         temp[i-1]=17+Math.floor(5*Math.random());  // Math.floor
         donne l'entier le plus proche
      };
     };
    function moyenner() {
      som=0;
      for (i=1;i<=30;i++) {
          som+=temp[i-1];
      };
      return Math.round(som/30);
    };
  relever();
  alert('la moyenne du tableau est ='+ moyenner());
```



Association avec un formulaire

- Utilisation dans un formulaire
  - Schéma d'utilisation
    - body :
      - Contient la définition du formulaire
      - Il fait appel aux variables et fonctions définies dans le head
    - head :
      - Contient les fonctions

Association avec un formulaire
Lire/Écrire

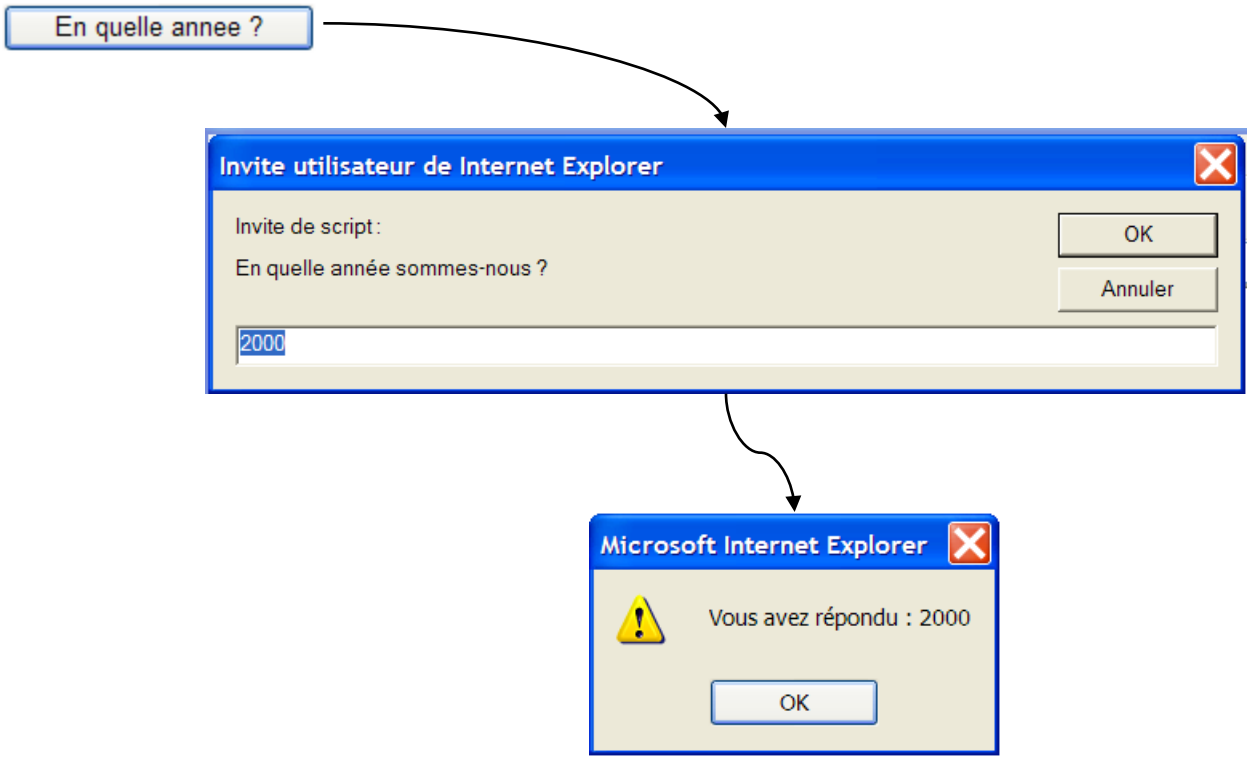
- Exemple 1 : appel à une fonction de lecture en cliquant sur un bouton du formulaire

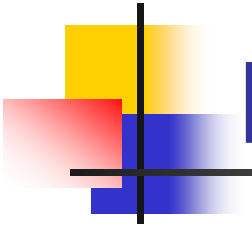
```
<html>
    <head>
     <title>Programme In2</title>
     <script language="JavaScript">
        function lireAnnee() {
           annee=prompt('En quelle année sommes-nous ? ', 2000);
           alert('Vous avez répondu : ' + annee) }
    </script>
</head>
<body>
    <form>
        <input type="button" value="En quelle annee ? "
                    onClick="lireAnnee()">
    </form>
</body>
</html>
```

Association avec un formulaire

Lire/Écrire : lire-ecrire-form0.html

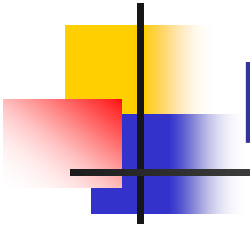
- Exemple 1 :





DOM

- Notion d’objet en Javascript
  - Le Javascript traite les éléments qui s'affichent dans votre navigateur comme des objets, c'est-à-dire des éléments :
    - classés selon une hiérarchie pour pouvoir les désigner précisément
    - auxquels sont associées des propriétés et des actions (méthodes)



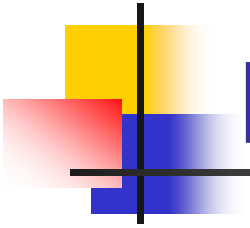
Les objets

- Comment JavaScript définit les objets ?
  - Javascript divise la page du navigateur en éléments (objets), afin de permettre d'accéder à n'importe lequel d'entre eux et de pouvoir les manipuler par l'intermédiaire de leurs propriétés
  - On commence généralement par l'objet le plus grand (celui contenant tous les autres) puis on descend dans la hiérarchie jusqu'à arriver à l'objet voulu
    - L'objet le plus grand est l'objet fenêtre : ***window***
    - Dans la fenêtre s'affiche une page, c'est l'objet ***document***
    - Cette page peut contenir plusieurs objets, comme
      - des formulaires,
      - des images, etc.

Les objets

- L'objet Navigator
  - a plusieurs propriétés concernant votre navigateur
    - *appName :*
      - *connaître le nom : Netscape, IE, Mozilla*
    - *appVersion :*
      - *connaître la version*
    - *language :*
      - *FR, AN*
    - *platform :*
      - *windows, Linux…*
  - *Pour le savoir : exécuter :*

```
<script language="Javascript">
    document.write(navigator.propriété); //où propriété =
    //platform…
</script>
```



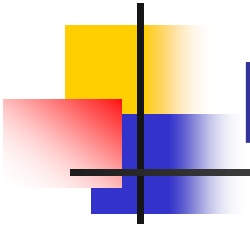
L’objet Navigator

- *Exemple d’utilisation de Navigator*

```
Nom = navigator.appName;
if (Nom == 'Netscape') {
    placer ici les instructions à exécuter s'il s'agit de
   Netscape Navigator 4 ou supérieur }
 if (Nom == 'Microsoft Internet Explorer') {
placer ici les instructions à exécuter s'il s'agit de
   Microsoft Internet Explorer 4 ou supérieur
}
```

Les objets

- L’objet Window
  - est l'objet par excellence dans Javascript, car il est le parent de chaque objet qui compose la page web
  - il contient donc :
    - l'objet document :
      - la page elle-même
    - l'objet location :
      - le lieu de stockage de la page
    - l'objet history :
      - les pages visitées précédemment
    - l'objet frames :
      - les cadres (division de la fenêtre en sous-fenêtres)



L’objet Window

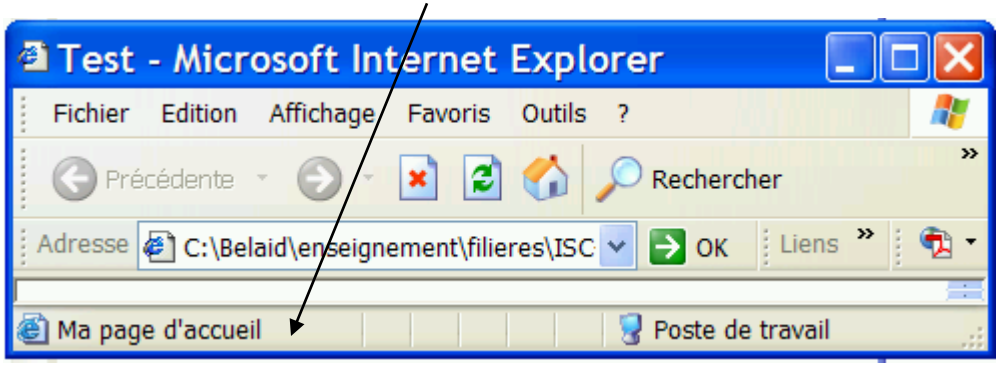
- L’objet Window
  - Propriétés :
    - Frames[] : tableau de cadres contenus
    - Length : nombre de cadres (nombre d'éléments du tableau *frames*
    - Name : nom de la fenêtre dans laquelle on se trouve
    - Parent : fenêtre qui englobe celle dans laquelle on se trouve
  - Méthodes :
    - alert(), confirm() et prompt() : font apparaître une boîte de dialogue
    - open(), et close() : permettent l'ouverture et la fermeture de fenêtres
  - Plusieurs exemples sous :

    http://fr.selfhtml.org/javascript/objets/window.htm

L’objet Window

- Propriété defaultStatus
  - affiche dans la barre d'état de la fenêtre d'affichage la valeur "Ma page d'accueil"
  - Exemple

```
<html>
<head>
<title>Test</title>
<script type="text/javascript">
   window.defaultStatus = "Ma page d'accueil";
</script>
</head>
<body>
</body>
</html>
```



http://fr.selfhtml.org/javascript/objets/window.htm#closed

L’objet Window

- Propriétés innerHeight, innerwidth
  - permettent de fixer au cours de l’exécution la hauteur et la largeur de la fenêtre

```
<html>
<head><title>Test</title>
<script type="text/javascript">
   window.innerHeight = 300;
</script>
</head>
<body>
</body>
</html>
```

L’objet Window

- La méthode open ()
  - Cette fonction ouvre une nouvelle fenêtre
    - window.open(" test.html","nom_de_la_fenetre","options_de_la_fenetre")
      - Ouvre « secondefenetre » et y affiche le fichier test.html. Secondefenetre peut être utilisé comme target pour l’affichage de l’extérieur
  - Exemple

```
<html>
<head>
<title>Test</title>
<script type="text/javascript">
    function nouvellefenetre() {
            mafenetre = window.open("window0.html",
            "secondefenetre", "width=300, height=200, scrollbars"); }
</script>
</head>
<body>
<a href="javascript:nouvellefenetre()"> nouvelle fenêtre </a><br>
</body>
</html>
```

  - Pour ouvrir une fenêtre à une position donnée : ajouter les attributs: top=px, et left=px

L’objet Window

- la méthode close () :

```
<a href="javascript:mafenetre.close()">fermer la
  fenêtre</a>
```

  - En cliquant sur ce lien, cela ferme la fenêtre précédemment ouverte avec le nom « mafenetre »

L’objet Window

- Vérification de l’état de la fenêtre
  - Exemple

```
<html><head><title>Test</title>
<script type="text/javascript">
<!--
    var InfoWin = window.open("fichier1.htm", "secondefenetre");
    function CheckOpen() {
    if(InfoWin.closed == true) alert("La fenêtre a été fermée"); else
    alert("La fenêtre est encore ouverte"); }
//-->
</script>
</head>
    <body>
            <a href="javascript:CheckOpen()">La fenêtre est-elle
            fermée?</a>
    </body>
</html>
```

L’objet Window

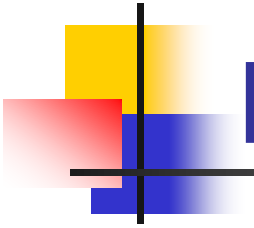
- Fermeture automatique d’une fenêtre, après 2’

```
<html>
    <head>
        <title>Test</title>
            <script type="text/javascript">
            var InfoWin = window.open("exercice1.html",
            "secondefenetre");
            InfoWin.setTimeout("top.close()",2000);
        </script>
</head>
   <body>
   </body>
</html>
```

Les objets

- L’objet document
  - élément majeur, permet de récupérer des informations d'un formulaire, créer des calques et les déplacer, écrire du texte...
  - Propriétés :
    - document.fgColor, permet de récupérer et de changer la couleur du texte de votre page HTML

      document.fgColor = "#993333";
    - document.bgColor, permet de récupérer et de changer la couleur de fond de votre page HTML
    - document.lastModified, permet de savoir quand la page html a été modifiée
      - document.lastModified;
        - Internet explorer renvoie : 11/07/2000 19:41:00 Netscape renvoie : Tuesday, November, 7 /2000 19:41:00



L’objet document

- document.linkColor
  - permet de récupérer et de changer la couleur des liens de votre page HTML
- document.location
  - permet de récupérer et changer l'url de votre page HTML, ce qui revient à charger une autre page HTML

    document.location = "URL/monDoc.HTML";
- document.write()
  - permet d'écrire dans votre page HTML
- document.images[ ]
  - permet de récupérer et changer les images de votre page HTML
- document.forms[ ]
  - permet de récupérer et changer les informations de votre formulaire

L’objet document

- Dernière date de modification
  - Donne la date de la dernière modification du document

```
<html>
<head><title>Test</title></head>
<body>
  <script type="text/javascript">
        document.write("dernière mise à jour: " +
        document.lastModified);
  </script>
</body>
</html>
```

L’objet document

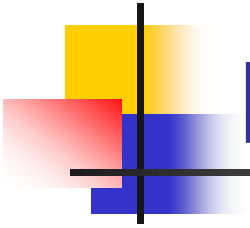
- Contenu d’une balise
  - Permet de récupérer le contenu de la balise <title>

```
<html>
   <head>
         <title>Test</title>
   </head>
   <body>
         <h1>
         <script type="text/javascript">
         <!--
                  document.write(document.title);
         //-->
         </script>
         </h1>
   </body>
</html>
```

L’objet document

- Récupération de l’URL où se trouve le document

```
<html>
  <head>
      <title>Test</title>
  </head>
  <body>
      <script type="text/javascript">
      <!--
      document.write("Ce fichier: " + document.URL);
      //-->
      </script>
  </body>
</html>
```



Le formulaire

- L’objet : forms
  - Avec l'objet forms, qui se trouve sous l'objet **document** dans la hiérarchie JavaScript, vous avez accès aux formulaires définis dans un fichier HTML
  - Syntaxe :
    - document.forms["nom_formulaire"].propriété
    - document.forms["nom_formulaire"].méthode
  - Plusieurs exemples sous :
    - http://fr.selfhtml.org/javascript/objets/forms.htm

Le formulaire

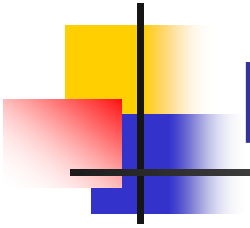
- Exemple
  - Il s'agit d'accéder à la case à cocher pour modifier le contenu de la zone du texte en inscrivant : case cochée ou case non cochée
  - La modification se fera par la fonction ModifChamp();
  - On déclare la case à cocher et la zone texte comme suit :

```
<form name="form1"> <br>
<input type="checkbox" name="checkbox"
   onClick="ModifChamp();"> <br>
<input type='TEXT' name='champ_text' value='Essai du
   javascript' size='24'>
</form>
```

Le formulaire

- Ensuite, on se réfère à la case à cocher et à la zone de texte à travers forms :

```
<script language="Javascript">
  function ModifChamp() {
   if (document.forms["form1"].checkbox.checked) {
     document.forms["form1"].champ_text.value='Bouton
   coché' }
   else {
   document.forms["form1"].champ_text.value='bouton
   non coché' } }
</script>
```



Le formulaire

- Le champ form a plusieurs propriétés :
  - **Action ()**
    - Définit l'URL où le formulaire sera envoyé
  - **Elements**
    - Tableau représentant les éléments du formulaire
  - **Length**
    - Nombre d'éléments à l'intérieur du formulaire
  - **Method**
    - Définit le type d'envoi du formulaire (get ou post)
  - **Name**
    - Définit le nom du formulaire
  - **Target**
    - Définit la page (fenêtre ou frame) de réponse
  - **Parent**
    - Indique une fenêtre d'un cadre (frame)

Le formulaire

- La propriété **action ()**
  - Pour réaliser l'action qui accompagne le formulaire

```
<html>
  <head><title>Test</title>
        <script type="text/javascript">
              function confirmation() {
              window.confirm("Ce formulaire est envoyé à
              " + document.formulaire_test.action); }
        </script>
  </head>
  <body>
        <form name="formulaire_test" action="mailto:toi-
        meme@cheztoi.com"  onSubmit="confirmation()">
        <input type="text" size="40" name="saisie">
        <input type="submit" value="Envoyer">
        </form>
  </body>
</html>
```

Le formulaire

- La propriété **length**
  - donne le nombre de formulaires définis

```
<body>
<form name="formulaire_test" action="mailto:toi-meme@cheztoi.com">
    <input type="hidden" value="Daniel">
    <input type="submit" value="Daniel">
</form>
<form name="formulaire_test" action="mailto:toi-meme@cheztoi.com">
    <input type="hidden" value="Antoine">
    <input type="submit" value="Antoine">
</form>
<script type="text/javascript">
    document.write(document.forms.length + " formulaires");
</script>
</body>
```

Le formulaire

- La propriété **method :**
  - Sauvegarde la valeur qui figure lors de la définition du formulaire dans l'attribut method, en principe égale à "get" ou "post"
  - Si l'utilisateur envoie le formulaire en cliquant sur le bouton d'envoi la fonction Methode() est appelée

```
<body>
    function Methode() {
            if(document.formulaire_test.action.indexOf("@") > 0)
              document.formulaire_test.method = "post";
            else
              document.formulaire_test.method = "get";
              return true;
    }
    …
    <form name="formulaire_test" action="mailto:toi-
            meme@cheztoi.com" onSubmit="return Methode()">
            <input type="text" size="40" name="saisie">
            <input type="submit" value="Envoyer">
    </form>
```

Le formulaire

- La propriété **name :**
  - Sauvegarde le nom d'un formulaire

```
<html>
   <head><title>Test</title>
   </head>
<body>
   <form name="formulaire_test" action="mailto:toi-meme@cheztoi.com">
            <input type="text" size="40" name="saisie">
            <input type="submit" value="Envoyer">
   </form>
   <script type="text/javascript">
            <!--
            document.formulaire_test.saisie.value =
                     document.formulaire_test.name;
            //-->
   </script>
</body>
</html>
```

Le formulaire

- L’envoi de mail :
  - On ne peut pas envoyer un formulaire tel qu’il est par mail (il faut utiliser php)
  - Cependant, on peut utiliser la formule suivante pour composer totalement un mail :
    - window.open("MAILTO:" + sDestinataire + " ?subject= " + sObjet + " &body=" + document.forms[0].elements["ta_commentaires"]) ;
      - mailto : pour l’adresse
      - ?subject : pour le sujet
      - &body : pour le texte du mail
      - ta_commentaires : est une chaîne de caractères qui rassemble l’information à mettre dans le corps du mèl

Le formulaire

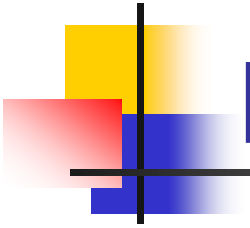
- La propriété **target :**
  - Précise la cible (cadre) dans laquelle l'affichage sera fait :

```
<html>
<head><title>Test</title>
        <script type="text/javascript">
            function cible() {
                    document.formulaire_test.target = "bas";
                    return true; }
        </script>
</head>
<body>
        <form name="formulaire_test" action="fichier.htm"
         onSubmit="return cible()">
        <input type="text" size="40" name="saisie">
        <input type="submit" value="Envoyer">
        </form>
</body>
</html>
```

Le formulaire

- L'action **submit() :**
  - Permet l'envoi du formulaire : JavaScript lance un compte à rebours avec la méthode **setTimeout()**. Après 60000 millièmes de secondes, donc après une minute, la fonction on_y_va() est appelée. Celle-ci envoie le formulaire avec submit()

```
<html>
<head><title>Test</title></head>
<body>
    <form name="formulaire_test" action="/cgi-bin/estime.pl"
        method="get">
            <input type="text" size="40" name="champ1"><br>
            <input type="text" size="40" name="champ2"><br>
        </form>
        <script type="text/javascript">
            function on_y_va() {
              document.formulaire_test.submit(); }
              window.setTimeout("on_y_va()",60000);
    </script>
</body></html>
```



Les elements

- Le champ **elements : sous-objet de forms**
  - Propriétés :
    - checked (coché)
    - defaultChecked (coché par défaut)
    - defaultValue (valeur entrée par défaut)
    - form (nom du formulaire)
    - name (nom de l'élément)
    - type (type de l'élément)
    - value (valeur/contenu de l'élément)
  - Méthodes :
    - blur() (quitter l'élément)
    - click() (cliquer sur l'élément)
    - focus() (positionner sur l'élément)
    - handleEvent() ((traiter l'événement)
    - select() (sélectionner du texte)

Le formulaire

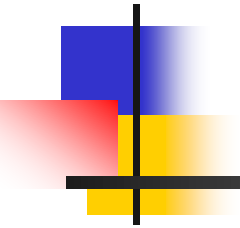
- Checked : exemple: Sauvegarde si oui ou non une case à cocher ou une case d'option est activée. Les valeurs possibles sont true ou 1 ou false ou 0.

```
<script type="text/javascript">
    <!-- function Ensuite() {
      if(document.formulaire_test.mode[0].checked == true)
      window.location.href="fichierfrm.htm"; else
      if(document.formulaire_test.mode[1].checked == true)
      window.location.href="fichier.htm";
      else alert("Veuillez faire un choix"); }
    //-->
</script>
</head><body>
<form name="formulaire_test" action="">
    <input type="radio" name="mode" value="avec"> avec cadres
    <input type="radio" name="mode" value="sans"> sans cadres <br>
    <input type="button" value="Lancer" onClick="Ensuite()">
</form>
</body>
</html>
```

Le formulaire

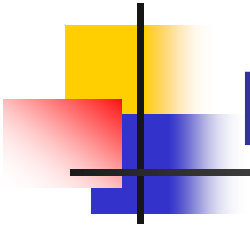
- **defaultValue :** Sauvegarde le texte par défaut d'un champ de saisie

```
<html><head><title>Test</title> </head>
<body>
    <form name="formulaire_test" action=""> uri: <input size="40"  name="uri"
    value="http://www.xy.fr/">
      <input type="button" value="Vas-y"
      onClick="window.location.href=document.formulaire_test.uri.value">
    </form>
    <script type="text/javascript">
    <!--
      if(navigator.userAgent.indexOf("en") > 0) {
      document.formulaire_test.url.defaultValue = "http://www.xy.com/";
      document.formulaire_test.url.value =
      document.formulaire_test.url.defaultValue; }
      //-->
    </script>
    </body>
</html>
```



Javascript

Gestion des événements



Les évènements

- Présentation
  - Les événements sont l'intérêt du JS en matière de programmation Web
  - Ils donnent une interactivité à la page que vous consultez, ce qui n'existe pas avec le HTML, si on excepte bien entendu le lien hypertexte
  - Le JS permet de réagir à certaines actions de l'utilisateur
- Pour cela, on précise :
  - L'événement (Event)
    - Clic de souris, survol de zones, chargement de la page…
  - Le gestionnaire de l'événement (OnEvent)
    - OnClick, onMouseOver… : appel de fonction pour agir en conséquence

Les évènements

- Syntaxe du gestionnaire d'événement :

on*Event*="Action_Javascript_ou_Fonction();"

- Lorsqu'il est utilisé dans un lien hypertexte, par exemple, la syntaxe sera la suivante :

```
<a href="URL"
    "on Evenement='Action_Javascript_ou_Fonction();'">
    Lien
</a>
```

Les évènements

- Le clic de souris
  - **Gestionnaire d'événement**
    - onClick
  - **Exemple :**
    - <input type="button" onClick="alert('vous avez cliqué sur le bouton') ;">
  - **Balises supportées :**
    - <input type="button">
    - <input type="checkbox">
    - <input type="radio">
    - <input type="reset">
    - <input type="submit">
    - <a href..>

Les évènements

- Le chargement
  - **Gestionnaire d'événement**
    - onLoad
  - **Exemple :**
    - <body onLoad="alert('la page est chargée !') ;">
  - **Balises supportées :**
    - <body>
    - <frameset>
  - **Effets :**
    - Au chargement, réaliser tel événement

Les évènements

- Error
  - **Gestionnaire d'événement**
    - onError
  - **Exemple :**
    - <img src="pix.gif" onError="alert('Erreur de chargement !') ;">
  - **Balises supportées :**
    - <body>
    - <frameset>
    - <img>
  - **Effets :**
    - Prévient l'erreur au chargement

Les évènements

- Abort
  - **Gestionnaire d'événement**
    - onAbort
  - **Exemple :**
    - <img src="pix.gif" onAbort="alert('Vous avez interrompu le chargement de l'image !') ;">
  - **Balises supportées :**
    - <img>
  - **Effets :**
    - Prévient l'erreur au chargement

Les évènements

- Le passage de la souris
  - **Gestionnaire d'événement**
    - onMouseOver
  - **Exemples :** onmouseover.html

```
<div style="width:50; height:50; background:lightsteelblue;"
    onMouseOver="alert('Le curseur entre dans la zone
    bleue');"></div>
<div> <P onMouseOver="this.style.color='red'"
    onmouseout="this.style.color='black'"> Move the mouse pointer
    over this text, then move it elsewhere in the document.</div>
<a href="http://www.google.fr" onMouseOver="alert('Pour aller sur
    google.fr, cliquer ici');">http://www.google.fr</a><a
    href="http://www.google.fr" onMouseOver="alert('Pour aller sur
    google.fr, cliquer ici') ;">http://www.google.fr</a>
```

  - **Balises supportées :**
    - Presque toutes
  - **Effets :**
    - Prévient quand on survole la cible : zone, texte, lien

Les évènements

- Le passage de la souris
  - **Gestionnaire d'événement**
    - onMouseOut
  - **Exemples :**
    - http://www.google.fr
  - **Balises supportées :**
    - Presque toutes
  - **Effets :**
    - Prévient quand on s'éloigne de la cible

Les évènements

- Le focus
  - **Gestionnaire d'événement**
    - onFocus
    - se déclenche lorsque l'élément reçoit le focus (devient actif) soit par action de l'outil de pointage (souris), soit par la navigation tabulée (touches du clavier)
      L'événement onfocus est l'opposé de l'événement onblur
  - **Exemple :**

```
<input type="text" size="40" maxlength="50" name="pharma_name"
 onFocus='javascript:this.style.backgroundColor="yellow"'
 onMouseOver='javascript:this.style.background="yellow"'
onMouseOut='javascript:this.style.background="white";this.style.bord
    er="1"'  onBlur='javascript:this.style.backgroundColor="white"'>
```

  - **Balises supportées :**
    - Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, window
  - **Effets : tester**

Les évènements

- Le blur :
  - Gestionnaire d'événement
    - onBlur
  - Exemple :
    - <input type="text" onblur="this.size = 20" onfocus="this.size = 50">
  - Balises supportées :
    - <input type="text">
    - <select>
    - <textarea>
    - <input type="password">
  - Effets :
    - onBlur : permet de savoir lorsque le champ perd le focus
    - onFocus : permet de savoir lorsque le champ a le focus

Les évènements

- Les changements
  - **Gestionnaire d'événement**
    - onChange
  - **Exemples :**

    <input type="text" value="votre nom" name="nom"
    onChange="alert('vous avez changé votre nom')">
  - **Balises supportées :**
    - <input type="text">
    - <select>
    - <textarea>
    - <input type="password">
  - **Effets :**
    - Avertit du changement par rapport à ce qu'il y avait d'écrit dans la zone d'écriture

Les évènements

- La sélection
  - **Gestionnaire d'événement**
    - onSelect
  - **Exemples :**

```
<form>
    Select text: <input type="text" value="Hello world!"
    onselect="alert('You have selected some of the text.')">
    <br /><br />
    Select text: <textarea cols="20" rows="5"
    onselect="alert('You have selected some of the text.')">
    Hello world!</textarea>
</form>
```

  - **Balises supportées :**
    - <input type="text">
    - <textarea>
  - **Effets :**
    - Avertit de la sélection d'un champ

Les évènements

- L’envoi
  - **Gestionnaire d’événement**
    - onSubmit
  - **Exemple :**

```
<form name="testform" action=“function()" onsubmit="alert('Hello ' +
    testform.fname.value +'!')"> What is your name?<br />
<input type="text" name="fname" />
<input type="submit" value="Submit" />
</form>
```

  - **Balises supportées :**
    - <input type="submit">
  - **Effets :**
    - L’événement se produit quand le bouton de soumission est actionné

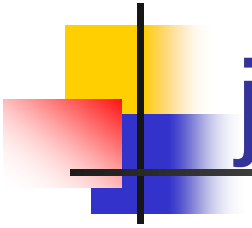
Les évènements

- Le reset
  - **Gestionnaire d'événement**
    - onReset
  - **Exemple :**
    - <input type="reset" value="Effacer" name="effacer" onSubmit="alert('On efface tout !') ;">
  - **Balises supportées :**
    - <input type="reset">
  - **Effets :**
    - En appuyant sur le bouton effacer, il remet dans la zone de texte : votre nom

jQuery

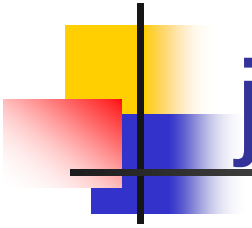
Éléments du langage

http://www.siteduzero.com/tutoriel-3-160885-manipuler-le-contenu.html



jQuery

- Introduction
  - Bibliothèque JavaScript libre
  - Permet de traverser et de manipuler facilement l'arbre DOM des pages web à l'aide d'une syntaxe fortement similaire à celle de Xpath
  - Permet par exemple de changer/ajouter une classe CSS, créer des animations, modifier des attributs, etc.
  - Gère les événements JavaScript
  - Fait des requêtes AJAX simplement



jQuery

- Installation
  - jQuery est tout simplement un fichier JavaScript
  - Il vous suffit donc de le télécharger sur le site officiel

    http://jquery.com
  - Intégration dans la page Web

    ```
    <script type="text/javascript" src="jquery.js"></script>
    ```
  - Ou directement sur Google code

    ```
    <script type="text/javascript"
      src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js">
    </script>
    ```

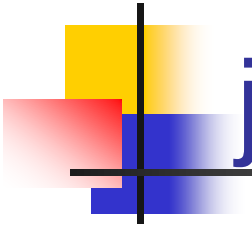
Squelette du fichier HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
 <head>
  <title>Titre</title>
  <meta http-equiv="Content-Type" content="text/html;
  charset=iso-8859-1" />
 </head>
 <body>
   <!-- le contenu -->
  <script type="text/javascript" src="jquery.js"></script>
  <script type="text/javascript">
   // c'est ici que l'on va tester jQuery
  </script>
 </body>
</html>
```

- Exemple : helloworld.html

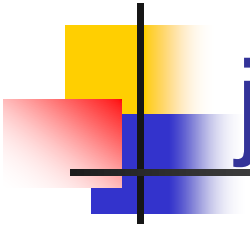
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
 <head>
  <title>Hello World avec jQuery</title>
  <meta http-equiv="Content-Type" content="text/html;
   charset=iso-8859-1" />
 </head>
 <body>
     Salut tout le monde !
   <script type="text/javascript" src="jquery-1.3.2.js"></script>
  <script type="text/javascript">
    $('body').html('Hello World');
  </script>
 </body>
</html>
```

- Commentaires
  - Si « Hello World » s'affiche, tout va bien
  - Si « Salut tout le monde ! » s'affiche, JavaScript est peut être désactivé, jquery.js n'est pas dans le même dossier que le code HTML, ou vous avez fait une erreur en recopiant



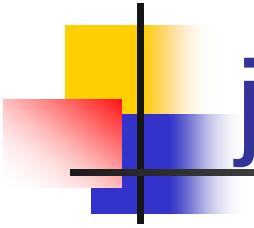
jQuery

- Fonction principale
  - jQuery repose sur une seule fonction :
    jQuery() (abrégée $())
  - C'est une fonction JavaScript
  - Elle accepte des paramètres
  - Elle retourne un objet
- C'est une fonction clé
  - Elle focalise le traitement sur l'objet en paramètre :
    - permet de le trouver
    - d'appliquer dessus des propriétés et des méthodes



jQuery

- $('tout-objet')
  - $ accepte un sélecteur CSS en argument
  - Le sélecteur peut être un identifiant :
    - $('#nomID') // retourne un élément ≅ document.getElementById
  - Le sélecteur peut être une classe sélecteur
    - $('.nomClasse') //retourne tous les éléments qui correspondent à cette classe
  - $ accepte plusieurs classes
    - $('.article,.nouvelles,.edito') //retourne tous les éléments qui correspondent à ces classes

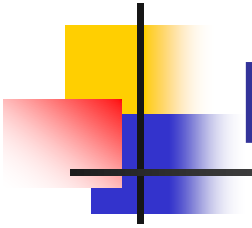


jQuery

- $ accepte des sélecteurs spécifiques
  - $(':radio'), $(':header'), $(':first-child')
- des sélecteurs en forme de filtres
  - $(':checked'), $(':odd'), $(':visible')
- Plus fort :
  - $(':contains(du texte)')
- des attributs
  - $('a[href]'), $('a[href^=http://'), $('img[src$=.png]'

Exemple

```
<html>
<script type="text/javascript" src="http://
  ajax.googleapis.com/ajax/libs/jquery/1/jqu
  ery.min.js">
</script>
<body>
<div id="monDiv">Bonjour</div>
<a href="#"
  onClick="jQuery('#monDiv').hide();">
disparition</a>
</body>
</html>
```



Manipulation du contenu

- Quelques méthodes sur l'objet sélectionné
  - html()
    - permet d'accéder au contenu de l'objet si on l'appelle sans aucun argument,
    - et permet de modifier le contenu de l'objet si on l'appelle avec comme premier argument une chaîne de caractères (représentant le contenu)

- Exemple
  - Pour ce code :
    - J'aime les frites. :
  - La méthode suivante :
    - $('#titre'); // Sélectionne notre balise mais ne fait rien
    - alert($('#titre').html()); // Affiche le contenu "J'aime les frites."
    - $('#titre').html('Je mange une pomme'); // Remplace le contenu ("J'aime les frites.") par "Je mange une pomme"
    - $('#titre').html($('title').html()); // Remplace le contenu par le titre de la page (contenu dans la balise <title>)

- text() :
  - soit le code HTML suivant

```
<p id="premier">
  <span class="texte">
    Salut tout le monde
  </span>
  <img src="photo_1.jpg" alt="Premiere Photo !" />
</p>
```

  - $('#premier').text()
    - renvoie "Salut tout le monde" (avec tous les retours à la ligne et les espaces)
  - $('#premier').text('les pommes')
    - change le contenu du paragraphe en :

```
&lt;strong&gt;les pommes&lt;/strong&gt;
```

- replaceWith()
  - permet de remplacer **la balise et son contenu**
  - On peut lui passer du code html ou encore un objet jQuery
  - Ainsi

    ```
    $('div').replaceWith('<span>Salut!</span>')
    ```
  - transformera

    ** **Bonsoir.. :)** **
  - en

    ****Salut!****.

- replaceWith()
  - Plus complet !

```
// Remplace les liens <a>...</a> par <em>censuré</em>
$('a').replaceWith('<em>censuré</em>');
$('h1').replaceWith($('.titre:first'));
$('#titre').replaceWith('<h1>'+$('#titre').html()+'</h1>');
$('.recherche').replaceWith('<a
   href="http://google.com">Google</a>');
```

- replaceAll

```
// Tous les <h1> vont être remplacés
$('#titre').replaceAll($('h1'));
// Revient à faire :
$('#titre').replaceAll('h1');
```

- Insertion à l'intérieur et à l'extérieur : prepend() et append()
  - $('span').prepend('balise span » ') transformera

    ****Hello World!****
  - en

    ****balise span » Hello World !****
  - $('span').append(' « balise span') : transformera

    ****Hello World !****
  - en

    ****Hello World ! « balise span****.

- append (contenu)
  - Ajoute du contenu à l'intérieur des éléments spécifiés
  - Exemple 1
    - Ajoute un objet jQuery à la suite du contenu de chaque paragraphe

      $("p").append($("b"));
    - Code de test:

      I would like to say: Hello
    - Résultat:

      I would like to say: Hello
  - Exemple 2
    - Ajoute du contenu HTML à la suite du contenu de chaque paragraphe

      $("p").append("Hello");
    - Code de test:

      I would like to say:
    - Résultat:

      I would like to say: Hello

- After (contenu)
  - Insère du contenu après chaque élément de la sélection
  - Exemple 1
    - Insère un élement à la suite des paragraphes.

      ```
      $("p").after( $("#foo")[0] );
      ```
    - Code de test:

      ```
      <b id="foo">Hello</b><p>I would like to say: </p>
      ```
    - Résultat:

      ```
      <p>I would like to say: </p><b id="foo">Hello</b>
      ```
  - Exemple 2
    - Insère un objet jQuery a la suite des paragraphes

      ```
      $("p").after( $("b") );
      ```
    - Code de test:

      ```
      <b>Hello</b><p>I would like to say: </p>
      ```
    - Résultat:

      ```
      <p>I would like to say: </p><b>Hello</b>
      ```

- Before /after
  - Insère du contenu avant les éléments retournés par la recherche
  - Exemple
    - Insère un objet jQuery (similaire à un tableau d'élements DOM) avant tous les paragraphes

      ```
      $("p").before( $("b") );
      ```
    - Code de test:

      ```
      <p>I would like to say: </p><b>Hello</b>
      ```
    - Résultat:

      ```
      <b>Hello</b><p>I would like to say: </p>
      ```

- insertAfter()/insertBefore()
  - Insère tous les éléments définis par A après ceux définis par B dans l'expression $(A).insertAfter(B). Equivaut à $(B).after(A)
  - Exemple

```
$("p").insertAfter("#foo");
```

    - Code de test:

```
<p>I would like to say: </p> <div id="foo">
  Hello</div>
```

    - Résultat:

```
<div id="foo">Hello</div><p>I would like to say: </p>
```

- remove
  - Supprime tous les élements de la DOM répondant aux critères de sélection. Mais ne supprime PAS les élements de l'objet jQuery, ce qui permet une utilisation de ces élements même si ceux ci ne figurent plus dans le document
  - Exemple

```
$("p").remove();
```

    - Code de test:

```
<p>Hello</p> how are <p>you?</p>
```

    - Résultat:

```
how are
```

- Exemple : contenu-jquery-correction.html
  - Soit le document HTML : contenu-jquery.html
  - Proposez du code jQuery pour faire ces manipulations
    - enlever les liens ;
    - enlever le texte en gras ;
    - enlever le texte en italique ;
    - enlever le texte décoré ;
    - vider les boutons ;
    - voir le code ;
    - transformer les liens en boutons ;
    - dupliquer le texte ;
    - regrouper les liens ;
    - mettre des titres ;
    - regrouper les titres ;
    - colorer le texte ;
    - organiser sémantiquement le texte ;

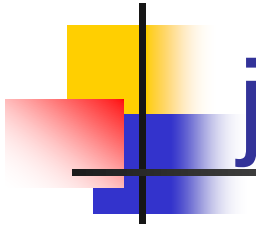
- Exemple : contenu-jquery-correction.html
  - <button onclick="semantique()">Organiser sémantiquement le texte</button>
  - <button onclick="colorer()">Colorer le texte</button>
  - <button onclick="mettreTitres()">Mettre des titres</button>
  - <button onclick="liensEnBoutons()">Transformer les liens en boutons</button>
  - <button onclick="dupliquerTexte()">Dupliquer le texte</button>
  - <button onclick="regrouperTitres()">Regrouper les titres</button>
  - <button onclick="regrouperLiens()">Regrouper les liens</button>
  - <button onclick="viderBoutons()">Vider les boutons</button>
  - <button onclick="enleverLiens()">Enlever les liens</button>
  - <button onclick="enleverGras()">Enlever le texte en gras</button>
  - <button onclick="enleverItalique()">Enlever le texte en italique</button>
  - <button onclick="enleverDecor()">Enlever le texte décoré</button>
  - <button onclick="voirCode()">Voir le code</button>

- Exemple : contenu-jquery-correction.html

```
function enleverLiens() {
  $('#contenu a').remove();
}

function enleverGras() {
  $('#contenu strong,#contenu b').remove();
}

function enleverItalique() {
  $('#contenu em,#contenu i').remove();
}

function enleverDecor() {
  $('#contenu *:not(html):not(body):not(p):not(button)').remove();
}

function viderBoutons() {
  $('#contenu button').empty();
}
```

```
function voirCode() {
  $('#contenu p').text($('p').html());
}

function liensEnBoutons() {
  $('#contenu a').wrap('<button></button>');
}

function dupliquerTexte() {
  $('#contenu p').clone().appendTo('#contenu');
}

function regrouperLiens() {
  $('#contenu a').wrapAll('<div></div>');
}

function mettreTitres() {
  $('#contenu .titre1').wrap('<h1></h1>');
  $('#contenu .titre2').wrap('<h2></h2>');
}
```

```
function regrouperTitres() {
  $('h1').wrapAll('<div></div>');
  $('h2').wrapAll('<div></div>');
}

function colorer() {
  $('#contenu .rouge').wrap('<span style="color:red"></span>');
  $('#contenu .vert').wrap('<span style="color:green"></span>');
  $('#contenu .orange').wrap('<span style="color:orange"></span>');
  $('#contenu .bleu').wrap('<span style="color:blue"></span>');
}

function semantique() {
  $('#contenu .italique').wrap('<i></i>');
  $('#contenu .gras').wrap('<b></b>');
  $('#contenu .souligne').wrap('<u></u>');
  $('#contenu .barre').wrap('<del></del>');
}
```



jQuery et le DOM

- Objet jQuery
  - La fonction principale ($()) renvoie un **objet jQuery** à partir duquel on peut naviguer dans l'arborescence
- Élément du DOM
  - $('body')[0] **désigne le premier élément** de l'objet jQuery résultant de l'appel à la fonction principale
  - Exemple

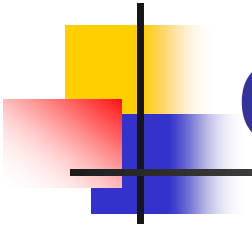
```
// Renvoie 'body'
$('body')[0].tagName.toLowerCase();
```

Exemple complet

```
// Déclaration d'un objet
var objet = {};
// Déclaration d'attributs
objet.variable1 = 'Hello';
objet['variable2'] = 'World';
// Affiche 'Hello World'.
alert(objet.variable1+' '+objet.variable2);
// On ne peut pas utiliser le point
// car une variable ne commence
// jamais par un chiffre
objet[0] = 'premier élément';
// [0] ou ['0'] reviennent au même
objet['1'] =
    document.getElementById('titre');
// Affiche 'premier élément'.
alert(objet[0]);
// Affiche 'titre' si la balise existe
alert(objet[1].id);
```

```
// Ce sont des variables comme les autres
// donc je mets ce que je veux !
objet.length = 'Moi aimer frites !';
objet.selector = 89;
```

Les éléments ([0], [1] etc.) contenus dans un objet jQuery sont donc des attributs de cet objet



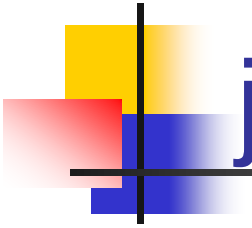
Quelques méthodes

- Parents et enfants
  - parents() renvoie l'ensemble des **éléments parents**
  - parent() **l'élément parent direct**
  - children() renvoie **les éléments enfants directs**

- Exemple

```
<html>
 <head>
  <script type="text/javascript"
   src="http://ajax.googleapis.com/aj
   ax/libs/jquery/1.3.2/jquery.min.js">
   </script>
  <style type="text/css" media="all">
  </style>
 </head>
 <body>
  <div id="contenu">
  </div>
 </body>
</html>
```

```
// Affiche 'HTML'.
alert($('body').parent()[0].tagName);
// Affiche 'HEAD'.
alert($('title').parents()[0].tagName);
```



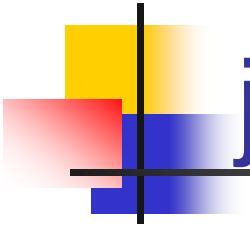
jQuery et le DOM

- Créer des éléments
  - Au lieu de
    - document.createElement('balise') ;
  - On utilise
    - $('<balise/>')
  - Par exemple
    - $('<h1 />') crée un titre de niveau 1

Exemple

```
<html>
  <head>
    <script type="text/javascript"
     src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js"></script>
    <style type="text/css" media="all">
    </style>
  </head>
  <body>
    <div id="contenu">
<button onclick="essai1();">Essai 1</button>
<button onclick="essai2();">Essai 2</button>
<button onclick="essai3();">Essai 3</button>
<button onclick="essai4();">Essai 4</button>
    </div>
  </body>
</html>
```

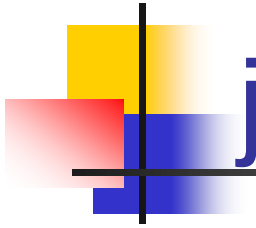
```
$(function(){
document._createElement =
    document.createElement;
document.createElement = function(balise){
  alert(balise+' créée !');
  document._createElement(balise);
};
});
function essai1(){
  $('<span/>');
}
function essai2(){
  $('<span />');
}
function essai3(){
  $('<span></span>');
}
function essai4(){
  $('<span>Super contenu :p </span>');
}
```



jQuery et le DOM

- Les attributs

```
// Renvoie l'attribut 'title'
// de l'élément ayant comme id 'titre'
$('#titre').attr('title');
// Écrit après #titre son attribut 'title'
$('#titre').after($('#titre').attr('title'));
//donne la valeur 'le site…' à l'attribut 'title'
$('div.header_gauche img').attr('title','le Site du Zér0');
```



jQuery et le DOM

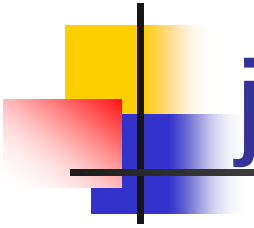
- Les formulaires

```
<input type="text" id="texte" value="Salut!" />
<textarea id="zonetexte">Ceci est une
zone de texte !</textarea>
<input type="radio" name="choixradio" value="Radio 1" />Radio 1
<input type="radio" name="choixradio" value="Radio 2" />Radio 2
<input type="radio" name="choixradio" value="Radio 3" />Radio 3
<select id="choixselect">
  <option>Select 1</option>
  <option>Select 2</option>
  <option>Select 3</option>
</select>
```

jQuery et le DOM

- Les formulaires

```
// Renvoie 'Salut!'.
$("#texte").val();

// Renvoie 'Ceci est une\nzone de texte !'.
$("#zonetexte").val();

// Renvoie l'attribut value de la balise sélectionnée.
// Par exemple 'Radio 1'.
$('input[name="choixradio"]:checked').val();

// Renvoie le contenu de la balise option sélectionnée.
// Par exemple 'Select 1'.
$('#choixselect').val();
```



jQuery et le DOM

- **Boucler sur des éléments**
  - each() sur un objet jQuery
  - Exemple

Exemple

```
<html>
 <head>
  <script type="text/javascript"
   src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js"></script>
  <style type="text/css" media="all">
  </style>
 </head>
 <body>
  <div id="contenu">
<a href="http://www.google.fr">Google France</a>
<a href="http://www.siteduzero.fr">Site du Zéro</a>
<p>Paragraphe.<br />Comment allez-vous ?
 <q>petite citation</q>
</p>
<a href="http://jquery.com">Faites un tour sur jQuery.com !</a>
  </div>
 </body>
</html>
```

```
//fonction anonyme
$(function(){
$('#contenu a').each(function(i){
    $(this)
     .prepend('<small>Lien n°'+(i+1)+' »</small> ')
    .append(' <small>« '+$(this).attr('href')+'</small>');
});
});
```

Exemple

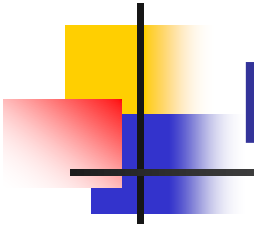
```
<html>
 <head>
  <script type="text/javascript"
   src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js"></script>
  <style type="text/css" media="all">



  </style>
 </head>
 <body>
  <div id="contenu">



  </div>
 </body>
</html>
```

```
//each() avec des données
$.each([
  0,1,1,2,3,5,8,13,21
],function(n){
  $('#contenu').append(n+'ème nombre
     de Fibonacci : '+this+'<br />');
});
//Résultats
     0ème nombre de Fibonacci : 0
     1ème nombre de Fibonacci : 1
     2ème nombre de Fibonacci : 1
     3ème nombre de Fibonacci : 2
     4ème nombre de Fibonacci : 3
     5ème nombre de Fibonacci : 5
     6ème nombre de Fibonacci : 8
     7ème nombre de Fibonacci : 13
     8ème nombre de Fibonacci : 21
```

Exemple

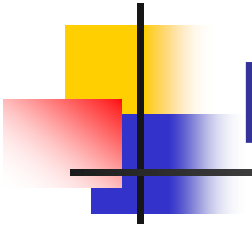
```
<html>
 <head>
  <script type="text/javascript"
   src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js"></script>
  <style type="text/css" media="all">



  </style>
 </head>
 <body>
  <div id="contenu">

<q>Un petit pas pour
   l'<strong>homme</strong> !</q>
<q>Et un grand pas pour
   l'<strong>humanité</strong> !</q>
  </div>
 </body>
</html>
```

```
/each() avec un tableau
var tableau = [
 'Bonjour, comment vas-tu ?',
 "Aujourd'hui, il fait beau !",
 $('q:first').text()
]
,chaine = '';
$.each(tableau,function(n,texte){
 chaine += 'Texte n°'+(n+1)+' :
   '+texte+'\n';
});
$(function(){
$('<pre></pre>')
 .html(chaine)
 .appendTo('#contenu');
});
//Résultat
Un petit pas pour l'homme ! Et un
   grand pas pour l'humanité !Texte
   n°1 : Bonjour, comment vas-tu ?
   Texte n°2 : Aujourd'hui, il fait beau !
   Texte n°3 : Un petit pas pour
   l'homme !
```



Les événements

- Fonctionnement
  - jQuery dispose de méthodes simples pour **attacher des événements à des fonctions** (ou « écouter un événement »)



Les événements

- Ce qu'on utilisait avant :

```
element.addEventListener('evenement',function(){
  // Action
});
// ou
element.onevenement = function(){
  // Action
};
```

- En jQuery
  - Les événements seront créés grâce à des **méthodes** ayant pour nom le type de l'événement, l'argument étant la**fonction de retour**

```
// Écoute d'un événement
elements_jQuery.evenement(function(){
  // Action
});
```

Les événements

➔ Toutes les fonctions que l'on va voir sont sans argument

- Utilisation dans un formulaire
  - Sélection
    - select est déclenché lorsque du texte est sélectionné dans un <input type="text" /> ainsi que dans un <textarea></textarea>
  - Exemple

```
$(':text,textarea').select(function(){
  alert($(this).val());
});
```

- Changement
  - Change est déclenché lorsque un champ <input …> est changé
- Exemple

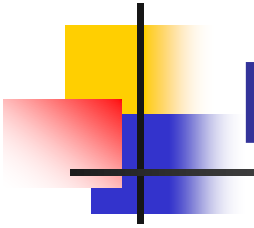
```
$(':input').change(function(){
  alert($(this).val());
});
```

- ***Soumission du formulaire***
  - submit est déclenché lorsqu'un formulaire est soumis par, par ex. <input type="submit" />
- Exemple

```
$('form[name="inscription"]').submit(function(){
  if($('form[name="inscription"] :password:first').val().length < 6){
    alert('Veuillez rentrer au moins 6 caractères dans votre mot de passe');
    return false;
    }
});
```

- *Focalisation*
  - focus est déclenché lorsqu'un élément d'un formulaire est « focalisé » par l'utilisateur, soit en cliquant dessus, soit grâce aux raccourcis du navigateur qui permettent de parcourir la page (les tabulations par exemple)
  - Syntaxe

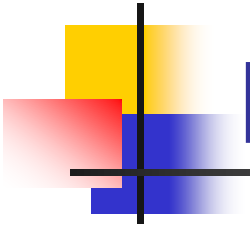
```
$(':input').focus(function(){
  $(this).css('background-color','#00f');
});
$(':input').blur(function(){
  $(this).css('background-color','#f00');
});
```



Événements

- Touches
  - L'appui sur une touche se décompose en trois étapes successives :
    - keydown : **enfoncement** de la touche
    - keypress : **maintien** de la touche enfoncée
    - keyup : **relâchement** de la touche
  - Le premier argument de la fonction de retour passée en paramètre est un objet contenant des informations sur la touche appuyée
  - Un attribut which ou keyCode (dépend du navigateur) désigne le numéro de la touche appuyée
  - Syntaxe

```
$(document).keypress(function(evenement){
  // Si evenement.which existe, codeTouche vaut celui-ci.
  // Sinon codeTouche vaut evenement.keyCode.
  var codeTouche = evenement.which || evenement.keyCode;
  alert(codeTouche);
});
```



Événements

- Souris
  - ***Clics de souris***
    - Un clic de souris se décompose en trois étapes successives :
      - mousedown : **enfoncement** de la souris
      - mouseup : **relâchement** de la souris ;
      - click : **clic** de la souris
    - Le premier argument de la fonction de retour passée en paramètre est un objet contenant des informations sur la touche appuyée, pageX pour la position X et pageY pour la position Y

- *Exemple*
  - *Récupérer la position du clic*
    - jQuery

```
$(function(){
$(document).mousedown(function(clic){
  $('#posX').text('Position X : '+clic.pageX);
  $('#posY').text('Position Y : '+clic.pageY);
});

});
```

    - HTML

```
<span id="posX"></span>
<span id="posY"></span>
```

- Autre exemple

```
<html>
 <head>
  <script type="text/javascript"
   src="http://ajax.googleapis.com/a
   jax/libs/jquery/1.3.2/jquery.min.js"
   ></script>
  <style type="text/css"
   media="all">
  </style>
 </head>
 <body>
  <div id="contenu">

<span id="posX"></span>
<span id="posY"></span>
  </div>
 </body>
</html>
```

```
$(function(){

$(document).mousedown(function(c
    lic){
 $('#posX').text('Position X :
   '+clic.pageX);
 $('#posY').text('Position Y :
   '+clic.pageY);
});

});
```

Résultat

Position X : 238 Position Y : 157

Autre exemple

```
<html>
 <head>
  <script type="text/javascript"
   src="http://ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js"></script>
  <style type="text/css" media="all">
body{cursor: pointer;}
#contenu{padding: 24px;
 cursor: pointer;}
#contenu span{
 background-color: #fff;}
  </style>
 </head>
 <body>
  <div id="contenu">
  </div>
 </body>
</html>
```

```
$(function(){
$('*')
 .mousedown(function(){
  $('#contenu')
   .css('background-color','#f00')
   .append('<span style="color: #f00">
   Down!</span>'); })
 .mouseup(function(){
  $('#contenu')
   .css('background-color','#00f')
   .append('<span style="color: #00f">
   Up!</span>');})
 .click(function(){
  $('#contenu')
   .css('background-color','#f0f')
   .append('<span style="color: #f0f">
   Clique!</span>');})
 .dblclick(function(){
  $('#contenu')
   .css('background-color','#0ff')
   .append('<span style="color: #0ff">
   Double Clique!</span>');
 });
```

Résultat


Down! Down! Down! Down! Down! Down! Up! Up! Up! Up! Up! Up! Clique! Clique! Clique! Clique! Clique! Clique!

Mouvements de souris

- La souris peut **entrer au dessus** d'un élément, **bouger** sur cet élément et enfin **partir** de cet élément.
- On distingue donc :
  - mouseenter : la souris entre au-dessus de l'élément ;
  - mouseleave : la souris quitte l'élément
  - mouseover : la souris entre au-dessus de l'élément ou un de ses enfants
  - mouseout : la souris quitte l'élément ou un de ses enfants
  - mousemove : la souris bouge sur l'élément
- Le premier argument de la fonction de retour passée en paramètre est un objet contenant des informations sur la touche appuyée, pageX pour la position X et pageY pour la position Y

- ***Exemple***

```
<html>
  <head>
    <script type="text/javascript"
     src="http://ajax.googleapis.co
     m/ajax/libs/jquery/1.3.2/jquery.
     min.js"></script>
    <style type="text/css"
     media="all">
    </style>
  </head>
  <body>
    <div id="contenu">

<span id="posX"></span>
<span id="posY"></span>
    </div>
  </body>
</html>
```

- ***jQuery***

```
$(function(){
$(document).mousemove(functio
    n(clic){
 $('#posX').text('Position X :
    '+clic.pageX);
 $('#posY').text('Position Y :
    '+clic.pageY);
});
});
```

Gérer des événements

- Fenêtre
  - *Défilement*
    - scroll est déclenché lorsque l'utilisateur utilise un ascenseur horizontal ou vertical

```
$(document).scroll(function(){
  alert('Arrête de me défiler !');
});
```

  - *Redimensionnement*
    - resize est déclenché lorsque l'utilisateur redimensionne la fenêtre en utilisant les poignées par exemple, mais aussi en réduisant sa fenêtre
    - Pour l'utiliser sur la fenêtre principale du navigateur il faut appliquer la méthode sur $(window) et non $(document)

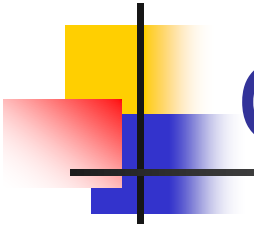
```
$(window).resize(function(){
  alert('Arrête de me redimensionner !');
});
```

- *D'autres événements utiles*
  - *Quand le document est prêt*
    - *Code*

```
$(document).ready(function(){
  alert('Le document est prêt !');
});
```

  - *Quand on quitte la page*
    - unload est déclenché lorsque l'utilisateur quitte la page. Vous ne pouvez pas faire grand chose à ce stade là au niveau de l'utilisateur à part envoyer une alerte
    - Code

```
$(document).unload(function(){
  alert('Au revoir et à bientôt !');
});
```



Créer des animations

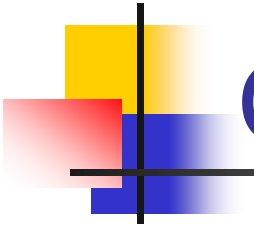
- animate(style, [duration], [easing], [complete])
  - Permet d'animer le style CSS de vos éléments
  - Paramètres
    - style (*objet contenant des couples attribut/valeur CSS*) :
      - le style de l'élément à la fin de l'animation
    - duration (*entier* ou *chaîne de caractères*) (durée) :
      - un entier positif qui est le nombre de millisecondes représentant la durée de l'animation
    - easing (*chaîne de caractères*) (évolution) :
      - "swing" , "linear"
    - complete (*fonction*) (fonction de retour) :
      - une fonction appelée quand l'animation d'un élément est finie
  - Exemple : animate.html, animate-jq.html

- jQuery

```
$('#anim1').click(function(){
    $('#contenu p')
    .css('width','400px')
    .animate({width : '500px'});
});

$('#anim2').click(function(){
  $('#contenu p span').animate({
     padding : '50px', opacity : 0.4},
    'slow','linear');
});

$('#anim3').click(function(){
    $('#contenu p strong')
    .animate({fontSize : '2em',
      paddingLeft : '50px' },
      2000,function(){
                alert('fini ! ');
      });
});
```

- HTML

```
<html>
 <head>
  <script type..."></script>
  <style type="text/css" media="all">
    #contenu p{
     width: 400px;font-size: 20px;color: #d22;
      line-height: 30px;border: 1px dashed #00f;}
    #contenu p span{ padding: 2px;
      background-color: #aaa;color: #000;}
      #contenu p strong {text-decoration:
   underline;
      padding: 0;border-left: 2px solid #00f;}
  </style>
 </head>
 <body>
  <div id="contenu">
<p>jQuery est une <span>bibliothèque
    <strong>JavaScript</strong>...
</p>
  <button id="anim1">Animation 1</button>
  <button id="anim2">Animation 2</button>
  <button id="anim3">Animation 3</button>
  </div>
 </body>
</html>
```



Créer des animations

- Agir sur le style CSS
  - On peut agir sur les paramètres CSS pour les modifier dynamiquement

Agir sur le style CSS

On peut agir sur les paramètres CSS pour les modifier dynamiquement

- jQuery

```
$('#plus').click(function(){
  $('#rectangle').animate({
    width : '+=32px'
  },1000,'linear');
});

$('#moins').click(function(){
  $('#rectangle').animate({
    width : '-=32px'
  },1000,'linear');
});
```

- HTML

```
#rectangle
{
  position: relative; display: block;
  width: 512px;height: 48px;
  padding: 8px;color: #fff;
  background-color: rgb(144,96,128);
}
    </style>
  </head>
  <body>
    <div id="contenu">
     <div id="rectangle">Salut
      !</div><br />
     <button id="plus">Plus !</button>
      <button id="moins">Moins
      !</button>
    </div>
```

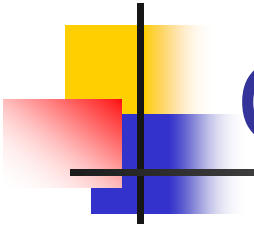
- ***Animer des ascenseurs***
  - ➔ Utiliser attributscrollTop et / ou scrollLeft afin de **faire défiler le contenu de l'élément**
  - ➔ Ceci ne marche qu'avec des éléments qui ont un ascenseur et qui peuvent défiler

- jQuery

```
$('body')
    .animate({ scrollTop : '800px'},
    5000);
```

- HTML

```
<html>
 <head>
  <script ..."></script>
  <style type="text/css" media="all"></style>
 </head>
 <body>
  <div id="contenu">
Salut !<br />Salut !<br />Salut !<br />Salut !<br
   />Salut !<br />Salut !<br />Salut !<br />Salut
   !<br />Salut !<br />Salut !<br />Salut !<br />
Salut !<br />Salut !<br />Salut !<br />Salut !<br
   />Salut !<br />Salut !<br />Salut !<br />Salut
   !<br />Salut !<br />Salut !<br />Salut !<br />
Salut !<br />Salut !<br />Salut !<br />Salut !<br
   />Salut !<br />Salut !<br />Salut !<br />Salut
   !<br />Salut !<br />Salut !<br />Salut !<br />
Salut !<br />Salut !<br />Salut !<br />Salut !<br
   />Salut !<br />Salut !<br />Salut !<br />Salut
   !<br />Salut !<br />Salut !<br />Salut !<br />
Salut !<br />Salut !<br />Salut !<br />Salut !<br
   />Salut !<br />Salut !<br />Salut !<br />Salut
   !<br />Salut !<br />Salut !<br />Salut !<br />
...
  </div>
 </body>
</html>
```



Créer des animations

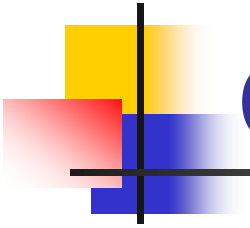
- Agir sur l'évolution d'une animation
  - On agit sur **les attributs de styles pour les faire évoluer au cours du temps**
  - linear :
    - les attributs évoluent **proportionnellement par rapport au temps**
  - swing :
    - **les attributs démarrent de façon moins brusque** que linear pendant la première moitié de l'animation **puis arrivent de façon moins brusque** vers leurs valeurs finales

- jQuery

```
$('#go').click(function(){
    $('#linear')
      .css('width','0px')
      .animate({ width : '90%' },
                4000,'linear'
      );
    $('#swing')
      .css('width','0px')
      .animate({ width : '90%' },
                4000,'swing'
      );
});
```

- HTML

```
<html>
 <head>
  <script …"></script>
  <style type="text/css" media="all">
   #conteneur{position: relative;display: block;
   margin: 24px;}
   #conteneur div{position: relative; display: block;
    width: 100%;margin: 0;color: #fff;width: 90%;
   border: 1px solid #000;}
   #linear{background-color: rgb(144,96,128);
            padding: 16px;}
   #swing{background-color: rgb(96,128,144);
            padding: 16px;}
  </style>
 </head>
 <body>
   <div id="contenu">
            <div id="conteneur">
                 <div id="linear">Linear</div>
                 <div id="swing">Swing</div>
            </div>
            <button id="go">Lancer les animations
            </button>
   </div>
 </body>
</html>
```



Créer des animations

- Agir sur la durée
  - En jQuery, **la durée de l'animation est toujours définie** (sinon c'est "normal", soit 400 millisecondes)
  - Pour modifier cette durée, il faut agir sur l'attribut correspondant
  - On appelle cela : évolution par l'attribut
  - Une des nouveautés de la version 1.4 est de pouvoir **attribuer une évolution à chaque attribut CSS**

Créer des animations

- Agir sur la durée
  - Exemple

```
$('p')
  .css('width','400px')
  .animate({
      width   : [ '800px' , 'linear' ], opacity : 0.5
  },
'linear');

$('p span')
    .animate({padding  : '50px', fontSize : '28px'},
      { duration : 'slow', easing : 'swing',
        specialEasing : {fontSize : 'linear'}
});
```

Gérer les animations

- Utiliser les effets
  - Tous les effets utilisent l'évolution swing
  - Visibilité
    - show()
      - permet d'**afficher** les éléments en question
    - hide()
      - permet de **cacher** les éléments en question
    - toggle()
      - permet de jongler entre la présence ou l'absence de l'élément (**si l'élément est caché, l'afficher, sinon le cacher**)
    - Exemple

```
// Utilisé pour afficher / cacher une balise secret.
$('blockquote.secret').toggle('normal');
```

- Ces trois méthodes peuvent
  - ne prendre aucun argument
    - ➔pas d'animation et tout se fait **de façon brute**
  - ou les arguments classiques cités plus haut :
    - ➔la hauteur, la largeur et l'opacité changent **de manière progressive**
- On peut utiliser toggle() avec les arguments classiques ou avec un seul argument, un booléen qui, à *true* affiche l'élément avec show(), mais qui à *false* cache l'élément avec hide()
- Exemple
  - $('a').toggle(true); // affiche les liens
  - $('textarea').toggle(false); // cache les zones de texte
  - // affiche ou cache les paragraphes selon que la checkbox soit cochée ou non
  - $('p').toggle($('#afficherParagraphes').attr('checked'));

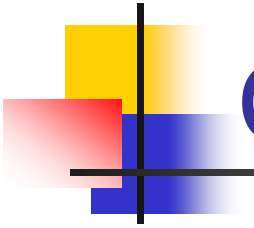
- Glissement
  - slideDown() permet de **dérouler** verticalement les éléments en question
  - slideUp() permet d'**enrouler** verticalement les éléments en question
  - slideToggle() permet de jongler entre la présence ou l'absence de l'élément (**si l'élément est caché, le dérouler, sinon l'enrouler**)
- Disparition
  - fadeIn() : fait **apparaître** les éléments en modifiant l'opacité de manière progressive
  - fadeOut() : fait **disparaître** les éléments en modifiant l'opacité de manière progressive
  - fadeTo() : modifie l'opacité des éléments sélectionnés et permet de lancer une fonction à la fin de l'animation

- Exemple sur les paramètres CSS
- jQuery

```
$('#boite').animate({
    // width ? => width 0
    width : 'toggle'
    // height ? => height 200
    , height : '200px'
},4000);

$('#boite').animate({
    // width 0 => width 800
    width : 'toggle'
    // height 0 => height 200
    , height : 'toggle'
},2000);
```

- HTML

```
<html>
 <head>
  <script …"></script>
  <style type="text/css" media="all">
   #boite{width: 800px;
          background-color: #000;
          color: #fff;
   }
  </style>
 </head>
 <body>
  <div id="contenu">
          <div id="boite">Une boite
          !</div>
  </div>
 </body>
</html>
```



Contrôler les animations

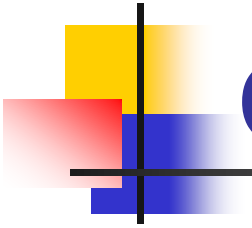
- Sélecteur d'animations
  - Le sélecteur :animated permet de filtrer les éléments qui sont **animés au moment où la fonction s'exécute**

- jQuery

```
function ajouterEtEnleverClasseAnime(){
  // ajouter la classe 'anime' aux éléments
     animés
  $('#contenu
     div:animated').addClass('anime');
  $('#contenu
     div:not(:animated)').removeClass('ani
     me');
}
function animerBoite( n ){
  var
    x = Math.floor( Math.random() * 500 )
     // nombres aléatoires entre 0 et 499,
     y = Math.floor( Math.random() * 500);
  $( '#boite' + ~~n ).animate({
     top  : y, left : x
  } , 2000 ); // l'animation dure 2 secondes
}
// met à jour tous les 500 millisecondes
setInterval(
     ajouterEtEnleverClasseAnime , 500 );
```

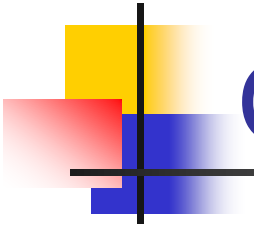
- HTML

```
#contenu div{
  background-color: #fff; position: absolute;
  width: auto;height: 64px; top: 200px;left: 200px;
  border: 1px solid #000;z-index: 50;}
#contenu div button {width: 100%;  height: 40px;
     margin: 12px 0;  color: #000;  padding: 0 12px;
     cursor: pointer;}
#contenu div.anime{ background-color: #00f;
  border-color: #fff;}
#contenu div.anime button{ background-color: #00f;
  border-color: #00f;  color: #fff;}
   </style>
  </head>
  <body>
    <div id="contenu">
<button
     onclick="ajouterEtEnleverClasseAnime();">Mettre à
     jour</button>
<div id="boite1"><button onclick="animerBoite( 1
     );">Animer!</button></div>
<div id="boite2"><button onclick="animerBoite( 2
     );">Animer!</button></div>
<div id="boite3"><button onclick="animerBoite( 3
     );">Animer!</button></div>
    </div>
```



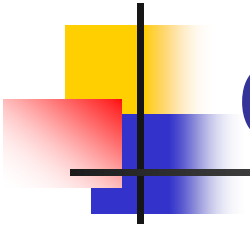
Gérer les animations

- Stopper les animations
  - stop([viderLaQueue, allerALaFin])
    - viderLaQueue (*booléen*) :
      - Si à *true* : stoppe l'animation en cours et celles à venir (qui **sont présentes dans la queue**)
      - Si à *false* : (par défaut), stoppe seulement la **première animation de la queue**. Les autres animations dans la queue **commenceront tout de suite**
    - allerALaFin (*booléen*) :
      - Si à *true* : ne stoppe pas l'animation, mais la fait **aller à sa fin** : l'élément concerné aura donc son style final et la fonction de retour sera appelée
      - Si à *false* (par défaut) : stoppe l'animation et ne fait rien de spécial
  - Exemple : stpper-anim.html



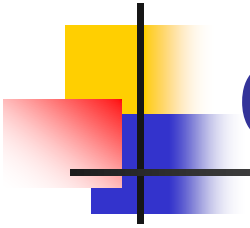
Gérer les animations

- Gestion de files d'attente
  - On peut gérer une ou plusieurs files d'attentes par lancement, mise en attente de plusieurs animations
  - Explorer cela par vous-mêmes



Gérer les événements

- Vocabulaire des événements
  - Quand un événement est déclenché sur un élément, et qu'il faut **associer une fonction à appeler lors de ce déclenchement**, on dit qu'on « écoute » un événement
  - On lie, ou on attache (« to bind ») une fonction à un type d'événement bien déterminé sur un élément. L'élément en question est alors « écouté » sur ce type d'événement
  - On supprime, on délie ou on détache (« to unbind ») une fonction à un type d'événement sur un élément lorsqu'on enlève l'appel à la fonction lors du déclenchement de l'événement sur cet élément
  - La fonction de retour est appelée un « écouteur » d'événement (addEventListener() signifie « ajouter un écouteur d'événement » !).



Gérer les événements

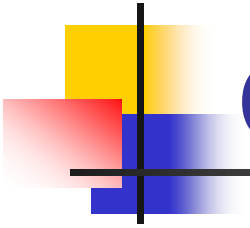
- Écouter
  - Les méthodes vues, comme click() ou mouseenter(), permettent de **lier un événement à une fonction** :
    - on appelle ça un écouteur d'événement.
    - bind() permet de généraliser cela en lui passant en premier argument une chaîne de caractères représentant tous les événements (séparés par des espaces) **à lier avec la fonction passée en paramètre**

- jQuery

```
$(function(){
$('#contenu p, #contenu textarea')
  .bind('mouseenter focus',function(){
    $(this).css('border-color','#222');
  })
  .bind('mouseleave blur',function(){
    $(this).css('border-color','#bbb');
  });
$('#contenu q').bind('dblclick',function(){
  if($(this).attr('auteur')){
    alert("L'auteur de cette citation est
     "+$(this).attr('auteur')+' !');
  }
});
});
```

- HTML

```
<html>
 <head>
  <script …></script>
  <style …>
   #contenu p, #contenu textarea{
   border: 1px solid #bbb;}
  </style>
 </head>
 <body>
  <div id="contenu">
   <p>Yipii!</p>
   <q auteur="Martin Luther King">I
     have a dream</q>
   <br /><br />
   Appuyer sur
  <code>[TAB]</code><br />
   <textarea>
  </div>
 </body>
</html>
```



Gérer les événements

- Supprimer
  - unbind() permet **d'enlever l'écouteur d'événement** créé préalablement avec bind()
  - nom de l'événement (unbind('click')) : supprime tous les écouteurs **du type de l'événement**
  - nom de l'événement et fonction (unbind('click',fonction)): supprime juste **un écouteur bien précis** pour ce type d'événement

- jQuery

```
$(function(){
$('#contenu a:not(.interne)').click(function(){
  alert('adresse du lien : '+$(this).attr('href'));
  $(this).unbind('click');
  return false;});
$('#contenu textarea')
  .focus(zoneTexte)
  .focus(function(){$(this).after('<br />Clic !'); })
  .blur(function(){$(this).css('background-
     color','#ff8');})
  .one('dblclick',function(){$(this).after('<br
     />Double Clic !');
    $(this)
      .unbind('focus',zoneTexte)
      .unbind('blur');});});

function zoneTexte(){
  $(this).css('background-color','#f8f');
}

function unbindZoneTexte(){
  $('#contenu textarea').unbind();
}
```

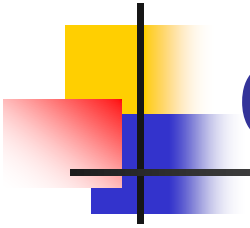
- HTML

```
<html>
 <head>
  <script type="text/javascript"
   src="http://ajax.googleapis.com/ajax/l
   ibs/jquery/1.3.2/jquery.min.js"></scrip
   t>
  <style type="text/css" media="all">
  </style>
 </head>
 <body>
  <div id="contenu">
<button
   onclick="unbindZoneTexte()">$('#con
   tenu textarea').unbind() !</button><br
   />
<a
   href="http://www.siteduzero.com/">Si
   te du Zér0</a><br />
<textarea>
  </div>
 </body>
</html>
```

Gérer des événements

- Attacher plusieurs événements en même temps
  - Une des nouveautés de la version 1.4 est de pouvoir donner un objet en argument à bind(), contenant des **couples nom de l'événement / fonction de retour attachée**
  - **Exemple**

```
$('h1')
  .bind({ mouseenter: function(){$(this).css('color','#000');
      }
    , mouseleave: function(){$(this).css('color','#00f');
      }
    , click: function(){ $(this).css('color','#ff0');
      }
  });
```



Gérer des événements

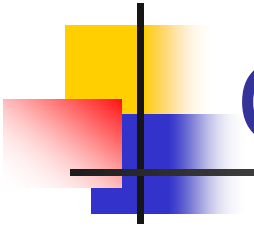
- Déclencher
  - On peut **déclencher virtuellement des événements sur des éléments sans avoir pour autant l'aval de l'utilisateur**
  - trigger()
    - trigger() permet de **déclencher un événement**, produisant l'action par défaut du navigateur ainsi que **celle que vous avez définie**
    - Les méthodes utilisées plus haut qui pouvaient être appelées sans argument sont donc **le raccourci d'un appel à trigger()**

- jQuery

```
$(function(){

$('#contenu :text')
  .focus(function(){
    $(this).trigger('click');
  })
  .click(function(){
    $(this).after('<br />clic !');
  });

});
```

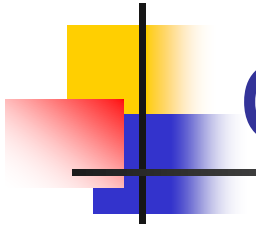
- HTML

```
<html>
  <head>
    <script type="text/javascript"
     src="http://ajax.googleapis.com/ajax/l
     ibs/jquery/1.3.2/jquery.min.js"></scrip
     t>
    <style type="text/css" media="all">



    </style>
  </head>
  <body>
    <div id="contenu">

Appuyer sur <code>[TAB]</code><br />
<input type="text" value="essai !" />
    </div>
  </body>
</html>
```



Gérer des événements

- triggerHandler()
  - triggerHandler() agit presque de la même manière que trigger() sauf qu'elle **n'actionne pas le mécanisme par défaut du navigateur** (par exemple quand un champ gagne le « focus », le texte du champ est sélectionné et on peut écrire), mais appelle seulement la fonction qu'on a définie



Gérer des événements

- Nos propres événements
  - On peut **créer nos propres événements**

- jQuery

```
$(function(){

$('#contenu a')
  .bind('lien',function(){
    alert('Vous aller vers '+$(this).attr('href')+' !');
  })
  .click(function(){
    if(!$(this).hasClass('interne')){
      $(this).trigger('lien');
    }
  });

});
```

- HTML

```
<html>
  <head>
    <script …"></script>
    <style type="text/css" media="all">
    </style>
  </head>
  <body>
    <div id="contenu">
<a
    href="http://www.siteduzero.com">Sit
    e du Zér0</a><br />
<a href="http://www.jquery.com">Site
    officiel de jQuery</a>
<a href="http://www.google.fr">Google.fr
    !</a> <a class="interne"
    href="http://moi.free.fr">Mon
    site!!!</a>
    </div>
  </body>
</html>
```