

LATITUDE
FORMATION
RÉFÉRENTIEL GÉOGRAPHIQUE
CARTOGRAPHIE
ANALYSE SPATIALE
LONGITUDE
COORDONNÉES GÉOGRAPHIQUES
POLOGIQUE
FORMATION GÉOGRAPHIQUE
RASTER
ION DE PROJET BDD
SIG
ATLAS
GPS
VECTEUR
LOGICIELS SIG

*Lea*fflet 



2D3D.GIS Organisme de formation
1 bis Perspective de l'Océan 17000 La Rochelle France

1

**PRÉSENTATION DE LEAFLET
INTRODUCTION AU LANGAGE
JAVASCRIPT**



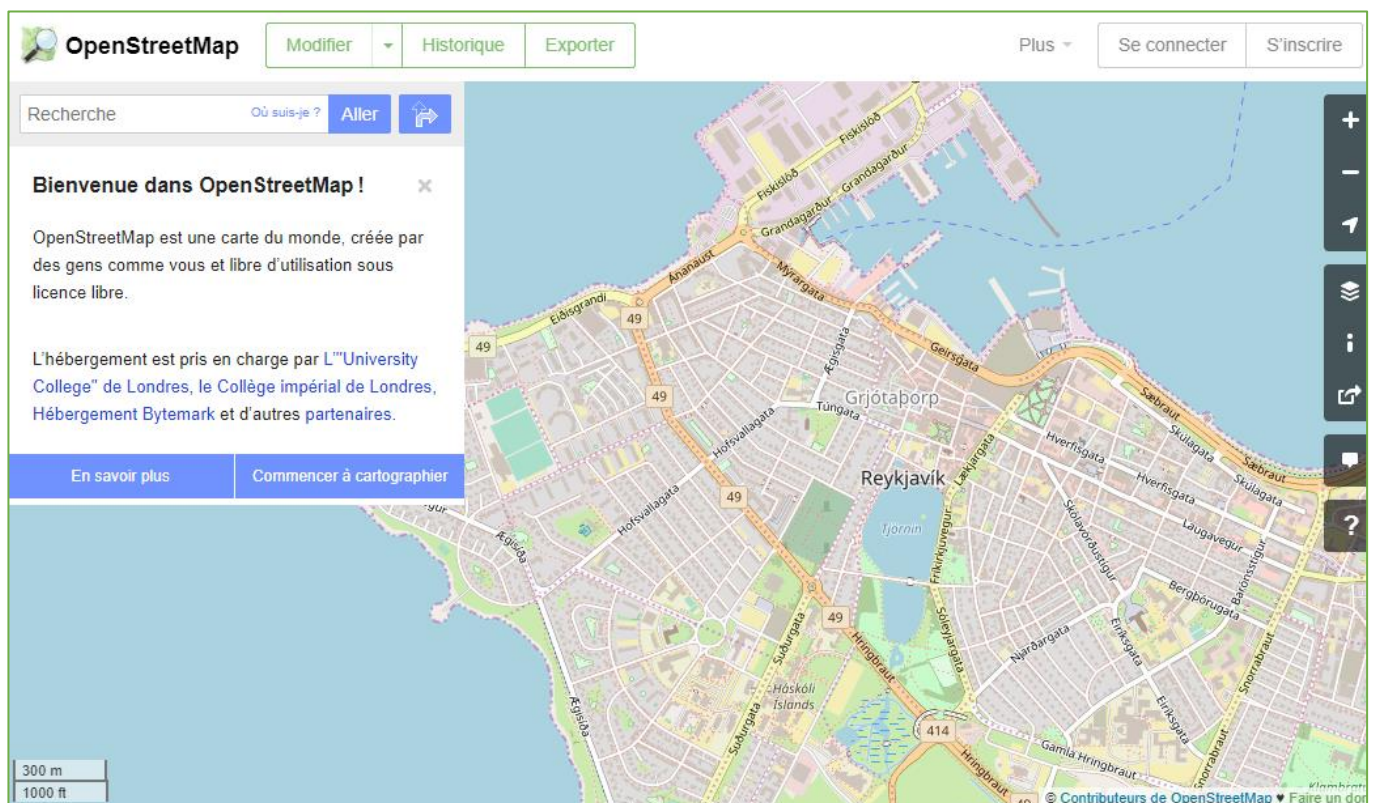
- Découvrir la notion d'API de développement
- L'architecture de développement web client
- Introduction au langage javascript

COURS

PRÉSENTATION DE LEAFLET

Leaflet est une bibliothèque (API) de cartographie en ligne gratuite, **open source** publiée sous licence BSD-2. Elle permet de déployer des composants cartographiques sur n'importe quel type de page web pour **tous les navigateurs** possibles (Internet Explorer, Edge, Safari, Chrome, Firefox, Opera, etc.) grâce à ses fonctions embarquées développées en langage **javascript**.

Une API est une interface de programmation permettant de développer « plus facilement » des fonctionnalités à l'aide de classes, fonctions et composants déjà programmés et documentés.



Le site www.openstreetmap.org est basé sur la bibliothèque leaflet (historiquement, il était sur OpenLayers). De nombreux autres sites célèbres utilisent Leaflet comme composant cartographique comme **Facebook**, **Wikipedia**, Flickr, Pinterest, Foursquare, etc.

Leaflet est concurrent des API cartographiques comme Google Maps API, OpenLayers, Bing Maps API, API Géoportail, etc.

LE LANGAGE JAVASCRIPT

PRÉSENTATION DE JAVASCRIPT

Leaflet est un ensemble de classes et fonctions écrites en langage javascript.

Le javascript est un langage de **programmation de scripts** principalement utilisé dans les pages web interactives. Environ 90% des sites web actuels utilisent le javascript.

Ce langage est incorporé dans un document HTML. Historiquement il s'agit même du premier langage de script pour le web. Il permet d'apporter des améliorations au langage HTML en offrant la possibilité d'exécuter des commandes du côté client, c'est-à-dire au niveau du navigateur et non du serveur web, notamment pour gérer toutes les interactions des utilisateurs (événements).

Ainsi le langage javascript est fortement dépendant du navigateur appelant la page web dans laquelle le script est incorporé, mais en contrepartie il ne nécessite pas de compilateur, contrairement au langage java (avec lequel il a longtemps été confondu), ou de plugin avec le langage flash.

FRAMEWORKS/LIBRAIRIES DE DÉVELOPPEMENT

De nos jours, la majeure partie des sites internet utilisent des frameworks ou librairies de développement javascript comme **jQuery** ou Bootstrap. Ces bibliothèques permettent de simplifier le développement et de le rendre compatible sous tous les navigateurs web.

Nous utiliserons par la suite jQuery.

EXEMPLE DE SCRIPT

Un script 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 spécifiques qui signalent au navigateur qu'il s'agit d'un script écrit en langage javascript.

Les balises annonçant un code javascript sont les suivantes :

```
<script type="text/javascript" language="javascript">  
  // Ici le code javascript  
</script>
```

EXERCICE

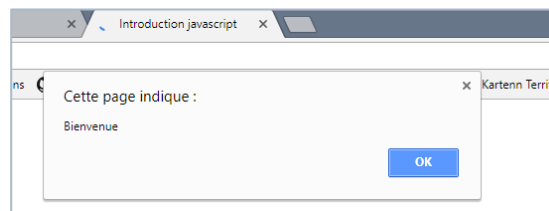
CRÉER UN PREMIER SCRIPT JAVASCRIPT

- Créez un dossier nommé **leaflet_intro** à l'emplacement de votre choix
- Créez un nouveau fichier HTML nommé **index.html**
- Éditez le fichier index.html avec votre éditeur de texte préféré (Sublime Text, Atom, Brackets, Notepad++, etc.)

- Saisissez le code HTML suivant :

```
<html>
<head>
  <title>Introduction javascript</title>
  <script type="text/javascript" language="javascript">
    alert("Bienvenue");
  </script>
</head>
<body>
</body>
</html>
```

- Enregistrez le fichier
- Double-cliquez dessus pour l'exécuter dans un navigateur web (**Google Chrome est conseillé** pour tous les exercices suivront, ce navigateur offrant la meilleure console de débogage)
- Vous devriez avoir le résultat suivant, à l'ouverture de la page HTML :



Vous venez d'exécuter votre premier script javascript.

FONCTIONS ET ÉVÉNEMENTS

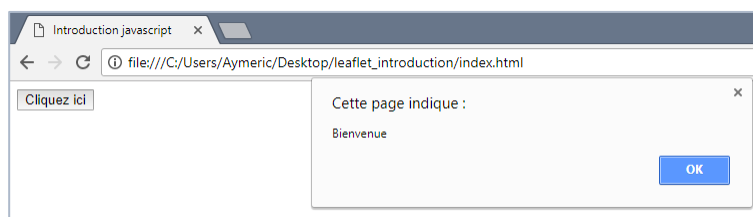
- Modifiez le contenu entre les **balises script** pour y créer une fonction nommée **afficheMessage** sans paramètre (avec des parenthèses ouvrante et fermante) et qui sera exécutée sur une action de l'utilisateur, et non au chargement de la page.

```
<script type="text/javascript" language="javascript">
  function afficheMessage() {
    alert("Bienvenue");
  }
</script>
```

- Enregistrez le fichier **index.html** et **rafraîchissez** la page dans le navigateur web pour constater le résultat : **Rien ne s'exécute**
- Pour que la fonction s'exécute, il va falloir créer un **appel de fonction** sur une interaction avec l'utilisateur, par exemple, le **clic sur un bouton**. Ajoutez un bouton nommé **Cliquez ici** dans le corps de la page HTML.

```
<body>
  <button onclick="afficheMessage();">Cliquez ici</button>
</body>
```

- Enregistrez à nouveau et rafraîchissez la page, puis cliquez sur le bouton pour lancer l'appel de fonction :



La gestion des événements en HTML/javascript permet de rendre dynamique les pages en fonction des actions de l'utilisateur sur celle-ci. Dans l'exemple ci-dessus, nous avons utilisé l'événement **onclick**, qui déclenche la fonction **afficheMessage** lorsque l'utilisateur clique sur le bouton.

Les événements peuvent se placer sur n'importe quel type d'élément HTML, ici un **button**, mais il est possible de les utiliser sur des `body`, `div`, `p`, `span`, `img`, `input`, etc.

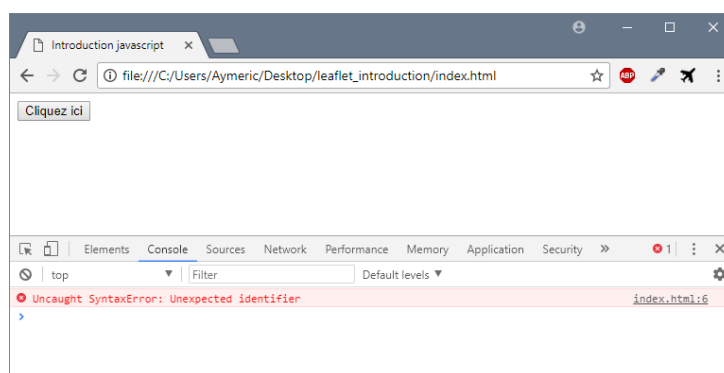
Les autres types d'événements que l'on peut placer dans les balises HTML sont :

- `onload` : chargement de la page
- `onclick` : clic avec la souris
- `onmouseover` : survol de la souris
- `onmouseout` : la souris quitte le survol
- etc.
- `onfocus` : focus actif (souris ou clavier)
- `onblur` : focus perdu (souris ou clavier)
- `onmousedown` : bouton de la souris est enfoncé
- `onmouseup` : bouton de la souris relâché après clic

CONSOLE DE DEBUGGAGE

Dans le cas où le message **Bienvenue** n'apparaît pas, il se peut qu'il y ait une erreur javascript. Pour vous aider à déboguer, il est possible d'utiliser la console embarquée dans le navigateur web.

- Lancez la console en cliquant sur la touche F12 du clavier (fonctionne avec tous les navigateurs du marché)
- Allez dans l'onglet **Console**



L'**erreur** apparaît en rouge avec le **numéro de ligne**. Vous vous rendez compte qu'il manque un caractère ou que vous avez fait une faute de frappe.

NB : S'il n'y a pas d'erreur javascript et que votre script ne fonctionne pas, vérifiez que votre navigateur **autorise le javascript** et que vous n'avez pas désactivé les **fenêtres d'alerte javascript**. Le cas échéant, il s'agit d'une erreur dans le **HTML**.

VARIABLES ET TYPES DE VARIABLES

Le message actuel est écrit directement en paramètre de la fonction javascript embarquée **alert**, nous allons maintenant le faire passer par une variable.

- Modifier le contenu de la fonction **afficheMessage** pour y ajouter la variable **message** :

```
function afficheMessage() {
  message = "Bienvenue";
  alert(message);
}
```

A l'exécution, vous ne devriez **pas voir de changement** car le fonctionnement n'a pas été modifié.

Il est à noter qu'en javascript les **variables n'ont pas obligation d'être typées**. C'est leur affectation qui déterminera le type de la variable, dans l'exemple ci-dessus, une **chaîne de caractère**.

FONCTIONS PARAMÉTRÉES

La fonction **afficheMessage** ne fait qu'afficher un message dont le contenu est **statique**, nous allons modifier la fonction pour qu'elle puisse être utilisée pour afficher plusieurs messages en fonction de l'appel de fonction.

- Modifier le contenu de la fonction **afficheMessage** pour y ajouter un paramètre **txt** qui sera passé à la variable **message** et ajouter un bouton avec un événement **onmouseover** qui affichera un autre message **Welcome** :

```
function afficheMessage(txt) {
  message = txt;
  alert(message);
}
```

```
<button onclick="afficheMessage('Bienvenue');">Cliquez ici</button>
<button onmouseover="afficheMessage('Welcome');">Survolez ici</button>
```

- Testez l'exécution, vous devriez avoir **2 messages** différents sur **2 événements** différents avec **une seule fonction**.

Le nom du paramètre de la fonction est libre. Il aurait été possible de passer directement le paramètre txt à la fonction alert().

SI CELA NE FONCTIONNE PAS : Pensez à regarder les erreurs dans la console (F12)



MODIFICATION DU CONTENU HTML EN JAVASCRIPT

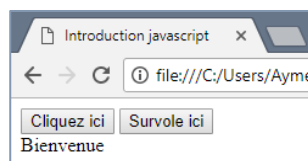
Le langage javascript permet de modifier toute partie de la page HTML, ce que l'on appelle le **DOM** (Document Object Model). Pour cela il existe des fonctions javascript embarquées.

- Modifiez la page HTML pour remplacer la fonction de popup **alert** par l'écriture du message dans la page HTML en elle-même, dans une div portant l'identifiant **information**.

```
function afficheMessage(txt) {
  message = txt;
  document.getElementById("information").innerHTML = txt;
}
```

```
<button onclick="afficheMessage('Bienvenue');">Cliquez ici</button>
<button onmouseover="afficheMessage('Welcome');">Survolez ici</button>
<div id="information"></div>
```

- Le résultat du clic ou du survol des boutons affiche désormais le message dans la page HTML :



ARCHITECTURE DE FICHIERS

Pour des raisons d'optimisation, d'organisation et de séparation du HTML et du javascript, il est recommandé de séparer les deux langages dans des fichiers distincts. Le but étant de créer des **fichiers javascript *.js** pour les appeler dans le HTML via des balises script avec une source :

```
<script type="text/javascript" language="javascript" src="js/application.js"></script>
```

- Créez un dossier nommé **js** dans le dossier **leaflet_intro** et y créer un nouveau fichier nommé **application.js**
- Coupez/collez le contenu qui se trouve **entre les balises script** de votre fichier html dans ce nouveau fichier js (NB : Ne mettez pas les balises script dans le fichier js)

```
1 function afficheMessage(txt) {
2   message = txt;
3   document.getElementById("information").innerHTML = txt;
4 }
5
```

- Ajoutez le chemin relatif dans la balise script pour faire pointer la source vers le fichier **application.js**

```
<html>
<head>
  <title>Introduction javascript</title>
  <script type="text/javascript" language="javascript" src="js/application.js"></script>
</head>
<body>
  <button onclick="afficheMessage('Bienvenue');">Cliquez ici</button>
  <button onmouseover="afficheMessage('Welcome');">Survolez ici</button>
  <div id="information"></div>
</body>
</html>
```

- Vérifiez que l'exécution fonctionne toujours. Si ce n'est pas le cas, vérifiez que le fichier application.js est bien chargé en regardant dans la console.

LIBRAIRIE JQUERY

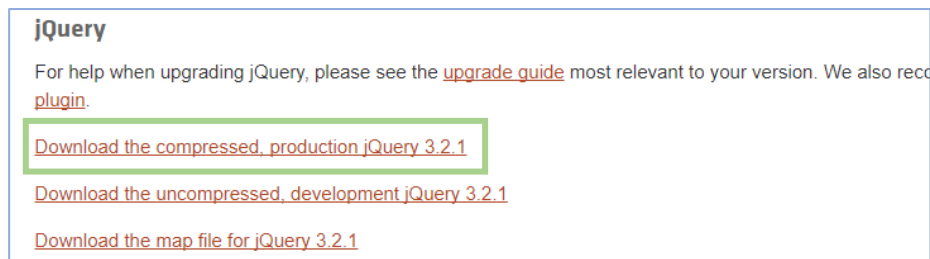
TÉLÉCHARGER JQUERY

De nos jours, il est devenu indispensable d'utiliser un framework ou une librairie de développement javascript qui permet de gagner un temps considérable pour le développement de fonctionnalités simples. Et bien plus ! Grâce à **jQuery**, une instruction qui prendrait 5 lignes de code, peut se faire en 1 seule.

Son slogan **Write less, do more** est on ne peut plus vrai.

- Télécharger la librairie jQuery (1 seul fichier) en allant sur le site www.jquery.com

- Cliquez sur le bouton marron **Download**
- Sur la page qui apparaît choisissez **Download the compressed, production...**



- Votre navigateur va télécharger automatiquement le fichier, copiez/collez le dans votre dossier **leaflet_intro/js** et laissez le nommé comme par défaut.

Si le fichier n'est pas téléchargé automatiquement, votre navigateur l'ouvrira directement et vous affichera le code source compressé du fichier.



```

/*! jQuery v3.2.1 | (c) JS Foundation and other contributors | jquery.org/license */
!function(a,b){"use strict";"object"==typeof module&&"object"==typeof module.exports?n
window:this,function(a,b){"use strict";var c=[],d=a.document,e=Object.getPrototypeOf,f
c=b.createElement("script");c.text=a,b.head.appendChild(c).parentNode.removeChild(c)}\
{jquery:c,constructor:r,length:0,toArray:function(){return f.call(this)},get:function(
r.each(this,a)),map:function(a){return this.pushStack(r.map(this,function(b,c){return
b=this.length.c+=a+(a<0?b:0):return this.pushStack(c)=0&&c<b?f(this[c]:[])}).end:functi

```

- Si tel est le cas, faites un **ctrl+S** pour enregistrer le fichier et mettez le dans votre dossier **leaflet_intro/js**.

Votre dossier js doit maintenant contenir deux fichiers.

 application.js	18/10/2017 16:55	Fichier de JavaScript	1 Ko
 jquery-3.2.1.min.js	18/10/2017 17:28	Fichier de JavaScript	85 Ko

UTILISER JQUERY

Pour utiliser jQuery dans vos scripts, il faut dans un premier temps inclure le fichier téléchargé précédemment dans la page HTML.

- Ajoutez la ligne suivante, avant l'appel à votre script perso **application.js**

```

<script type="text/javascript" language="javascript" src="js/jquery-3.2.1.min.js"></script>
<script type="text/javascript" language="javascript" src="js/application.js"></script>

```

- Pour tester le bon fonctionnement de jQuery, modifier la fonction `afficheMessage` et notamment la ligne contenant le **document.getElementById** par la ligne qui suit (n'oubliez pas le **#** devant **information**, signifiant l'id information) :

```

function afficheMessage(txt) {
    message = txt;
    $("#information").html(txt);
}

```

- Rafraîchissez la page pour tester le bon fonctionnement, votre message devrait toujours s'afficher sur la page. **Si vous obtenez une erreur de type \$ is not defined**, c'est que jQuery n'a pas pu être chargé, vérifiez le chemin.

Vous constatez rapidement que l'écriture du javascript avec jQuery est plus compréhensible et simplifiée par rapport au javascript natif notamment grâce au `$("#...")` qui remplace `document.getElementById("#...")` et également des fonctions pratiques comme ici la fonction `html()`.

⇒ Nous allons utiliser jQuery pour toute la suite des exercices.

CODE SOURCE DES FICHIERS A CE NIVEAU DU COURS

A ce niveau des exercices, les deux fichiers devraient être composés du code source qui suit :

```
<html>
<head>
  <title>Introduction javascript</title>
  <script type="text/javascript" language="javascript" src="js/jquery-3.2.1.min.js"></script>
  <script type="text/javascript" language="javascript" src="js/application.js"></script>
</head>
<body>
  <button onclick="afficheMessage('Bienvenue');">Cliquez ici</button>
  <button onmouseover="afficheMessage('Welcome');">Survolez ici</button>
  <div id="information"></div>
</body>
</html>
```

```
function afficheMessage(txt) {
  message = txt;
  $("#information").html(txt);
}
```

Nous allons pouvoir améliorer ce code grâce à jQuery en séparant totalement le HTML du javascript et notamment les appels de fonction sur les événements **onclick** et **onmouseover** qui polluent le code HTML.

Cette volonté de séparer le code permettant de gérer la partie dynamique de la partie visuelle est importante.

SÉPARATION DU HTML ET DU JAVASCRIPT

jQuery va permettre de séparer le code HTML des appels et fonction javascript. Pour cela nous allons utiliser des **event listeners**.

- Dans le fichier javascript, ajoutez les **deux event listeners** click et mouseover suivants :

```
$("#btnBienvenue").click(function () {
  afficheMessage('Bienvenue');
});
```

```
$("#btnWelcome").mouseover(function () {
  afficheMessage('Welcome');
});
```

- Dans le fichier HTML, **retirer les événements** sur les deux boutons et **ajoutez un id** sur chacun d'eux :

```
<button id="btnBienvenue">Cliquez ici</button>
<button id="btnWelcome">Survolez ici</button>
```

NB : Si vous tentez d'exécuter, **vous ne devriez pas avoir d'erreur** (si vous avez une erreur, corrigez là). Au clic ou au survol des boutons, rien ne se passe. Ceci est dû à l'exécution séquentielle de notre fichier HTML. Il commence par charger le javascript et crée 2 event listeners sur les boutons btnBienvenue et btnWelcome mais ces deux boutons n'existent pas encore dans la page car l'exécution séquentielle n'est pas encore arrivée au bout de la page.

Il ne peut donc pas rattacher les événements aux boutons. Pour faire en sorte que cela fonctionne, il va falloir ajouter un autre event listener, celui de la **fin du chargement de la page (document)**.

- Ajoutez l'**event listener ready** sur le document (la page HTML) et mettez y les 2 autres event listeners :

```
$(document).ready(function() {
    $("#btnBienvenue").click(function () {
        afficheMessage('Bienvenue');
    });

    $("#btnWelcome").mouseover(function () {
        afficheMessage('Welcome');
    });
});
```

- Retestez l'exécution du script et vous devriez récupérer les fonctionnalités sur les boutons.

Nous venons de totalement séparer le javascript du HTML, le code s'en trouve bien mieux organiser et plus propre. Voici le code **HTML à gauche** et **javascript à droite**, à ce stade de l'exercice :

```
<html>
<head>
  <title>Introduction javascript</title>
  <script type="text/javascript" language="javascript" src="js/jquery-3.2.1.min.js"></script>
  <script type="text/javascript" language="javascript" src="js/application.js"></script>
</head>
<body>
  <button id="btnBienvenue">Cliquez ici</button>
  <button id="btnWelcome">Survolez ici</button>
  <div id="information"></div>
</body>
</html>
```

```
function afficheMessage(txt) {
    message = txt;
    $("#information").html(txt);
}

$(document).ready(function() {
    $("#btnBienvenue").click(function () {
        afficheMessage('Bienvenue');
    });

    $("#btnWelcome").mouseover(function () {
        afficheMessage('Welcome');
    });
});
```

COMMENTAIRES JAVASCRIPT ET HTML

Il est important de rédiger des commentaires dans son code pour organiser le contenu en sections et aussi pour aider à la compréhension des commandes et actions réalisées.

Pour écrire un **commentaire dans un fichier HTML**, il faut utiliser les balises suivantes, sur une ou plusieurs lignes :

```
<body>
  <!-- Boutons d'action -->
  <button id="btnBienvenue">Cliquez ici</button>
  <button id="btnWelcome">Survolez ici</button>
```

```
<!-- Le résultat
      s'affichera ici -->
<div id="information"></div>
```

La même chose en **javascript** avec des commentaires sur une ou plusieurs lignes :

```
// Au chargement de la page
$(document).ready(function() {
    $("#btnBienvenue").click(function () {
        afficheMessage('Bienvenue');
```

```
/* Fonction d'affiche
   d'un message texte dans la page html */
function afficheMessage(txt) {
    message = txt;
    $("#information").html(txt);
```

Vous pouvez utiliser les commentaires pour désactiver temporairement une ligne de l'exécution pour des tests.

UTILISER LA CONSOLE POUR LOGGER DES VARIABLES

Nous allons voir ici comment utiliser la **console pour tester son code javascript**. La console va nous permettre de logger toute variable javascript et la parcourir qu'il s'agisse d'une variable simple ou d'un tableau.

La console permet aussi d'**exécuter du code javascript** à tout moment pour, par exemple, forcer l'appel d'une fonction, afficher

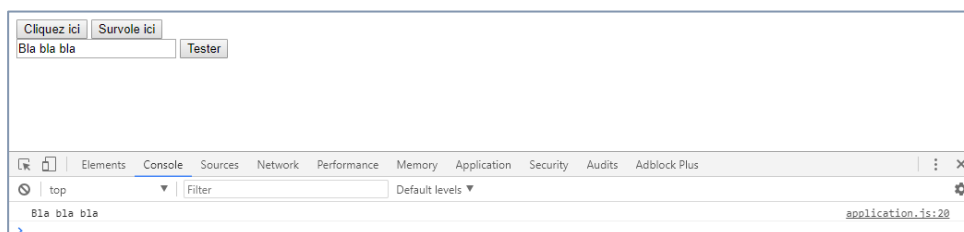
- Ajoutez un **input de type text** dans la page HTML, avec l'identifiant **txtSaisie** en dessous de la div information, ajoutez également un bouton qui permettra de lancer la fonction de test en mettant un id **btnSaisie** :

```
<div id="information"></div>
<div>
  <input type="text" id="txtSaisie" />
  <button id="btnSaisie">Tester</button>
</div>
```

- Côté javascript, ajoutez l'**event listener** du bouton **dans le document ready** et lancer les commandes qui seront chargées de récupérer la valeur saisie dans le champs texte grâce à la fonction **.val()**.

```
$("#btnSaisie").click(function() {
  valeur = $("#txtSaisie").val();
  console.log(valeur);
})
```

- Le résultat de l'exécution devrait vous permettre de récupérer ce que vous avez saisi dans le champs texte pour l'afficher dans la console (F12).



CONDITIONS ET BOUCLES JAVASCRIPT

Comme dans tout langage de programmation, javascript dispose de **structures conditionnelles** et de **boucles**. Pour tester

Nous allons utiliser le bouton et champ de saisie pour la tester la valeur et ainsi faire un affichage différent en fonction de celle-ci.

Dans l'event listener du click sur le bouton, nous allons ajouter un test permettant d'afficher le **type de valeur** qui a été saisi, du **texte** ou un **numérique**.

- Ajoutez les lignes suivantes après l'appel de la console :

```
if ($.isNumeric(valeur)) {
  afficheMessage('Vous avez saisi un nombre');
}
else {
  afficheMessage('Vous avez saisi un texte');
}
```

Le résultat de l'exécution doit vous permettre de visualiser si votre saisie est un texte ou un nombre :

Vous avez saisi un nombre
123

Vous avez saisi un texte
abc

La fonction **isNumeric** est fournie par la librairie **jQuery**, sans cette fonction, le test pour savoir si une valeur saisie est numérique ou non est un peu plus complexe :

```
if (!isNaN(parseFloat(valeur)) && isFinite(valeur)) {
```

On remarque donc encore une fois la grande utilité de **jQuery**.

EXERCICE EN AUTONOMIE

A partir des exercices précédents, vous allez créer une **calculatrice** qui réalise une opération sur 2 valeurs saisies dans des champs texte et qui affichera le résultat « **Le résultat est x** » comme ceci :

A 12 + B 8 =
Le résultat est 20

NB : La concaténation et l'addition utilisent tous les deux le même opérateur en javascript « **+** », le type des données des variables définira l'opération à effectuer. Le texte récupéré à partir des champs de saisie est toujours du **type texte** par défaut.

Si jamais on souhaite récupérer les convertir en nombres, il faut utiliser la fonction de casting **parseInt(variable)** ou **parseFloat(variable)** qui renverra un nombre entier ou décimal.

EXERCICE : JEU

Le javascript permet d'interagir également avec le style de la page. Dans cet exercice nous allons créer un mini-jeu dans lequel il faut **cliquer le plus rapidement possible sur un bouton**.

Un timer se déclenche lorsque la souris passe sur le bouton **GO** et s'arrête lorsque l'on clique dessus. Le timer s'arrête avec votre temps réalisé.

- Créez l'arborescence nécessaire au fonctionnement d'une page HTML + Script :
 - Créez un nouveau fichier HTML nommé **jeu.html**
 - Créez un fichier javascript **jeu.js** dans le dossier JS et appelez le dans le HTML

Code HTML

```
<html>
<head>
  <title>Introduction javascript</title>
  <script type="text/javascript" language="javascript" src="js/jquery-3.2.1.min.js"></script>
  <script type="text/javascript" language="javascript" src="js/jeu.js"></script>
</head>
<body>
  Chrono : <span id="timer">0</span> secondes
  <div><button id="btn">GO</button></div>
</body>
</html>
```

Code Javascript

```
tps = 0;
go = false;
stop = false;

$(document).ready(function() {
  $("#btn").mouseover(function () {

    if (!go) {
      window.setInterval(function(){
        go = true;
        var start = Date.now();
        return function() {
          tps = (Date.now()-start)/1000;
          if (!stop) $('#timer').html(tps);
        };
      }(), 1);
    }
    $(this).html("STOP !");
    largeurJeu = 1000;
    if (!stop) $(this).css("margin-left", Math.floor(Math.random()*largeurJeu));
  });

  $("#btn").click(function () {
    stop = true;
    $(this).html("WIN !");
  });
});
```