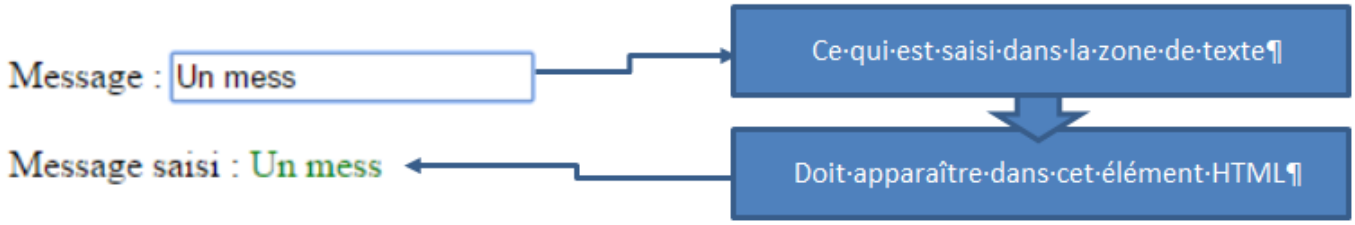
TD n°0

Exemple de sensibilisation...

Pas toujours évident de voir l'intérêt d'utiliser un framework ; Surtout si on a jamais eu, auparavant, l'occasion de travailler sur les mêmes sujets, mais sans framework...

Voici un exemple minimaliste permettant de se rendre compte de l'intérêt d'AngularJS :

Il s'agit de synchroniser un texte contenu dans la page avec une saisie effectuée dans une zone de texte : tout simple !



-- Javascript sans framework

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>javascript</title>
</head>
<body>
    <h1>Javascript</h1>
    <div>
        Message : <input id="txtMessage" type="text"><br>
        <br> Message : <span id="message"></span>
    </div>
<script type="text/javascript">
document.addEventListener("DOMContentLoaded", function(event) {
    var setTextContent=function(element, text) {
        while (element.firstChild!==null)
            element.removeChild(element.firstChild); // remove all existing content
        element.appendChild(document.createTextNode(text));
    };
    var changeFunction=function(event){
        var target = event.target || event.srcElement;
        setTextContent(document.getElementById("message"),target.value);
    };
    var addEvent=function(event,element,handler){
        if (document.addEventListener) {// For all major browsers, except IE 8 and
earlier
            element.addEventListener(event, handler);
        } else if (document.attachEvent) {// For IE 8 and earlier versions
            element.attachEvent(event, handler);
```

```
        }
    };
    addEvent("keyup",document.getElementById("txtMessage"),changeFunction);
    addEvent("paste",document.getElementById("txtMessage"),function(event){
        setTimeout(function(){changeFunction(event);}, 20);
        });
});
</script>
</body>
</html>
```

Il nous faut :

1. Intercepter le chargement terminé du DOM de la page (c'est à l'événement **DOMContentLoaded** de nous le dire) : sans cette précaution (pas vraiment indispensable sur cette exemple), il se pourrait que l'on agisse sur des éléments non encore chargés...
2. une fonction pour modifier le texte contenu dans un élément HTML : **setTextContent**
3. une fonction pour ajouter un listener sur un événement d'un élément DOM : **addEvent**
4. une fonction pour synchroniser la zone de texte et l'élément HTML : **changeFunction**
5. Ajouter l'écoute des événements **keyup** et **paste** qui impliquent une modification de la zone de texte, et ajouter un temps d'attente sur le paste, pour qu'il se soit terminé avant l'appel de changeFunction

Certaines de ces fonctions pourraient être réutilisées ailleurs (et donc délocalisées), mais l'ensemble est tout de même laborieux, et technique.

-- JQuery

```
<!DOCTYPE html>
<html data-ng-app>
<head>
<meta charset="UTF-8">
<title>JQuery</title>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
</head>
<body>
    <h1>JQuery</h1>
    <div>
        Message : <input id="txtMessage" type="text"><br>
        <br> Message : <span id="message"></span>
    </div>
<script type="text/javascript">
$( document ).ready(function() {
    $("#txtMessage").on("keyup",function(){
        $("#message").text($(this).val());
    });
    $.fn.pasteEvents = function( delay ) {
        if (delay == undefined) delay = 20;
        return $(this).each(function() {
            var $el = $(this);
            $el.on("paste", function() {
                $el.trigger("prepaste");
                setTimeout(function() { $el.trigger("postpaste"); }, delay);
```

```
            });
        });
    };
    $("#txtMessage").on("postpaste", function() {
        $("#message").text($(this).val());
    }).pasteEvents();
});
</script>
</body>
</html>
```

Il faut :

1. Agir lorsque le dom de la page est chargé : **$(document).ready**
2. Créer un événement **postpast**, pour agir après que le “coller” d'un texte se soit produit (c'est ce qu'on appelle un trick : bricolage ou astuce...)
3. Associer du code aux événements **keyup** et **postpaste**, pour synchroniser le contenu de la zone de texte et de l'élément HTML

Le code est propre, mais la technicité masque l'approche métier : difficile d'identifier au premier coup d’œil le rôle de ces lignes

-- AngularJS

```
<!DOCTYPE html>
<html data-ng-app>
<head>
<meta charset="UTF-8">
<title>AngularJS</title>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.8/angular.min.js"></script>
</head>
<body>
    <h1>AngularJS</h1>
    <div>
        Message : <input type="text" data-ng-model="message"><br>
        <br> Message : {{message}}
    </div>
</body>
</html>
```

Il faut :

1. Définir la portée de notre application : **<html data-ng-app>**
2. Déclarer que la zone de texte est synchronisée sur le model message : **data-ng-model=“message”**
3. Invoquer message avec une expression dans l'élément HTML : **{{message}}**

Une approche très propre, orientée métier...
Impressionnant non ?

Premiers pas

AngularJS est un framework côté client atypique, permettant de construire des applications client riche avec une véritable séparation entre les couches (métier/interface/contrôle).

Il introduit de nouveaux concepts (directives, services, modules…) avec lesquels il faut se familiariser dans un premier temps.

Pour vous permettre une prise en main plus rapide du Framework, nous avons choisi d'utiliser le cours réalisé par Code school mis à disposition du public.

- Créer un compte gratuit sur Code school
- Suivre en ligne le cours Shaping up with Angular.js