IUT - département Informatique **Janvier 2018**
1ère année Formation initiale **J-M. Ilié, E. Keita, J. Zanazaka**
Cours : CDIN

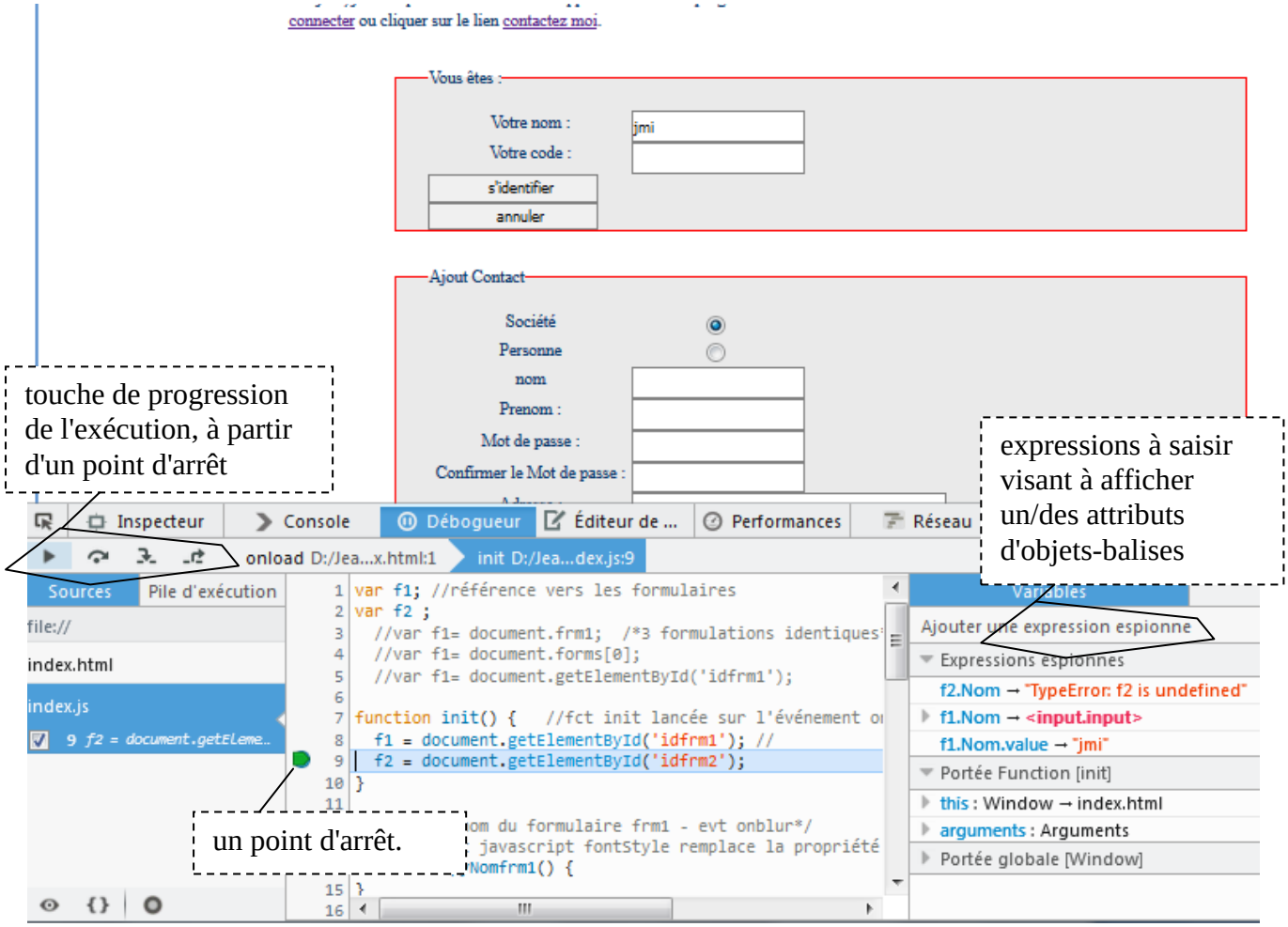
TP
«Les formulaires HTML et javascript»

BUT du TP

Reprendre le formulaire du site ePortFolio et l'améliorer
Spécification d'événements pour des balises html et de méthodes de réactions à ces événements.

Matériel:

- Considérer le site HTML fourni, notamment sa page `index.html` et 2 sous répertoires.
- Le sous répertoire `scriptJS` contient le fichier `index.js` destiné à l'écriture de script javascript
  Le second `styleCSS` contient notamment le fichier `form.css` pour les styles de présentation du formulaire.
- le navigateur donne accès à un débogueur de code javascript et une console d'affichage de message
  Pour les faire apparaître, lancer `inspecter element` en cliquant dans la fenêtre du navigateur (bouton droit de la souris).



Spécification :

On s'intéressera au second formulaire de la page `index.html`, concernant les points suivants :

-1- Vérification d'égalité entre mot de passe et confirmation de mot de passe
-2- Afin d'éviter une double saisie , recopie du nom saisi dans le 1er formulaire pour le proposer dans le 2ème formulaire.
-3- Gestion de la sélection d'un champ <select>.

Réalisation :

0- Rattacher le fichier javascript `index.js` au fichier `index.html`.

Dans l'entête de la page HTML (`head`), créer la balise `<script>` `<script type="text/javascript" src="…"></scripit>` en précisant l'attribut `src`.

Note sur la déclaration `<body onload="init()">` dans `index.html`,
- l'attribut `onload` de la balise `body` assure ici que la fonction `init()` sera lancée en réaction au chargement des objets-balises en mémoire. Cette fonction implémentée dans `index.js` est utilisée pour affecter des variables globales référençant en mémoire les objets formulaires.
- la déclaration d'attribut `onload="init()"` correspond en pratique à définir le corps de la fonction `window.onload` par un appel à la fonction init().

- Pour un affichage visant à déboguer votre code, les 2 possibilités suivantes s'offrent à vous :
  Utiliser l'instruction `alert('string')` qui affiche la chaîne passée en paramètre dans une fenêtre type pop-up.
  Ou bien utiliser l'instruction `console.log('string')` qui affiche la chaîne sous la forme d'un `log` dans la `console` du debogueur du navigateur (`firefox`). Faire `examiner element` dans la page web -> selection onglet `console` -> selection onglet `journal` , puis sélection `log`).
- Pour une exploitation du débogueur du navigateur (`firefox`), faire : `examiner element` dans la page web -> selection onglet `debogueur`, puis sélectionner le fichier `index.js`, mettre des points d'arrêts dans les fonctions an cliquant sur des numéros de lignes `ciblés`, puis lancer un des événements qui provoque l'exécution d'une fonction contrainte par un point d'arrêt.

1- Comparaison du mot de passe et sa confirmation.

Dans le fichier `index.html`, pour le champ de contrôle `cpasse` , spécifier l'événement `onchange` de façon à réagir quand l'utilisateur quitte le champ de confirmation du mot de passe.
Sur cet événement, lancer la méthode de réaction `cmpPasse(this)`, sachant que le paramètre `this` correspond à une référence sur le champ de saisie courant au moment de l'événement, ici `cpasse`.
Puis, développer cette méthode par la fonction du même nom dans le fichier `index.js`, de façon à comparer l'attribut value des champs de contrôle `passe` et `cpasse`
(syntaxe de la forme : `if (refObjet.value==…)`).

- Afficher l'état de la comparaison, via une fenêtre d'alert par exemple.

2- copie d'un champ de contrôle

- Pour le champ de contrôle `<input name="nom">` dans le second formulaire,

spécifier l'événement `onfocus` de façon à lancer exécuter la fonction `cpyNomfrm1()` au moment où le champ de contrôle gagne le `focus` (i.e. la saisie devient possible dans ce champs).
Cette fonction est à développer dans le fichier `index.js`. Commencer par faire référence aux 2 champs de contrôle de saisie des noms : `Nom` pour le 1er formulaire et `nom` pour le 2ème, de façon à recopier leur attribut `value` de l'un vers l'autre.

Note : il y a plusieurs façons de faire référence à un champ de contrôle,
soit via une référence à l'objet formulaire qui englobe le champ, en utilisant en sus l'attribut `name` comme référence relative au champ de contrôle ciblé, par exemple,
`document.getElementById('ifFrm1').Nom`
ou bien directement par exemple en exploitant une collection de références sur des balises ayant une même valeur d'attribut `name`, par exemple,
`document.getElementByName('nom')[i]` référence le i+1ème objet-balise de la page ayant `name="nom"`. Avec l'indice 0, il s'agit du 1er indice que 0 quand

- Faire apparaître la valeur copiée en lui appliquant en sus un style italique. Utiliser pour cela l'objet référencé par `style` dans l'objet-balise `nom`. Les propriétés de style sont définies dans cet objet comme autant de propriétés de cet objet, en respectant toutefois la règle d'écriture javascript dite camel-case, tel que `font-style` en css devient `fontStyle` en javascript. Parmi les valeurs possibles de `fontStyle` il y a `italic` et `normal`.
- Faire que la valeur copiée soit d'emblée sélectionnée, afin que toute saisie de l'utilisateur remplace cette valeur. Utiliser pour cela la fonction `select()` définie comme un attribut des objets-balises `input` (`refObjetnom.select();`)
- Spécifier les événements `onchange` dans la balise html de saisie du `nom`, de façon à pouvoir lancer à chaque fois la méthode de réaction `normalStyle()`. Puis, développer le code de cette fonction dans `index.js` pour obtenir un style sans italique (soit, `refObjetnom.style.fontSize= "normal"`).

3- Gestion dynamique du genre de la personne

Au niveau du source de `index.html`, on remarquera la présence d'un `<select name='genre'>`, permettant de sélectionner une valeur de genre à associer au nom d'une personne. Par défaut, ce select n'est pas affiché puisque ce formulaire cible par défaut une entreprise, d'où une règle spécifique à étudier dans le fichier `form.css`.

- Modifier la spécification des événements des 2 boutons radios, pour faire apparaître ce select quand l'utilisateur déclare une personne et le cacher quand l'utilisateur déclare une société. Spécifier pour cela l'événement `onclick` comme attribut des 2 boutons radios, afin de lancer l'exécution de la fonction `affSelect(btn)` à implémenter. On passera en paramètre la valeur `this`, correspondant à la référence sur l'objet balise courant au moment de l'événement, à savoir celle du bouton cliqué.

- Compléter les réactions à ces clics par une copie du nom du premier formulaire vers le second, via un appel à la fonction `cpyNomfrm1()` déjà implémentée.

- Afin de débogage, afficher la valeur de l'option sélectionnée, ce chaque fois que l'utilisateur change d'option. Spécifier l'événement `onchange` dans la balise `<select>` pour lancer l'exécution de la fonction `affSelection()` à implémenter.

Dans cette fonction, créer une référence sur l'objet-balise correspondant au select, pour accéder à 2 attributs: l'indice `selectedIndex` indiquant l'indice de l'option sélectionnée et le tableau `options` des références sur les objets-balises correspondant aux différentes options du select. On notera aussi que tout objet-balise correspondant à une option a un attribut `text` qui mémorise la valeur visible de l'option. C'est cette valeur qu'il faudra afficher pour l'option sélectionnée par l'utilisateur.

4- Vérification globale sur l'événement `onsubmit`

On s'intéressera ici à vérifier le format syntaxique de quelques champs de contrôle.

- Dans le fichier `index.html`, pour le formulaire `frm2` , spécifier l'événement `onsubmit`, de façon à lancer le code `return verifFrm2Format()`, à valeur booléenne. La fonction `verifFrm2Format` est déjà définie dans le fichier `index.js`, faisant appel modulairement à différentes fonctions de vérification, concernant le nom le téléphone et le mot de passe.
- En pratique, il s'agit de retourner la valeur `true` si la vérification est correcte induisant que le formulaire peut effectivement être soumis, ou `false` en cas d'erreurs détectées pour que le processus de soumission soit stoppé.
  Pour cela, spécifier `onsubmit="return verifFrm2Format()"` dans la balise du formulaire `frm2`.
- Développer ces dernières en spécifiant des `expressions régulières` servant de modèle pour les chaînes saisies. Se référer au cours pour cela, sachant en particulier que les chaînes de caractères sont en fait des objets javascript possédant différentes méthodes dont la fonction booléenne `match()` permettant la comparaison d'un modèle de chaîne avec une chaîne saisie.
  `chaine.match(model)` rend vrai si `chaine` correspond à `model`.
- Respecter les points suivants :
  - mot de passe : chaîne de caractères quelconque entre 6 et 8 caractères
  - nom : chaîne de caractères alphanumérique d'au moins 2 caractères
  - téléphone : 10 chiffres en local ou 11 dont le préfixe du pays.
- Pour des raisons de débogage,
  utiliser la fonction javascript `console.log(string)` permettant d'effectuer des affichages dans la console du navigateur. Les messages émis sont visibles sous l'onglet `journal`. Note : on peut utiliser d'autres classifications d'affichage, dont `console.error(string)` pour les cas d'erreurs.
  Autre possibilité : écrire dans la div d'identifiant msg ( ), spécifier pour être initialement invisible et rendu visible pour l'affichage d'un message d'erreur. L'écriture se fait en affectant la propriété `innerHTML` de l'objet-balise `div` cité et la propriété de style concernant la visibilité est `visibility` (valeur `hidden` ou `visible`).