Université Grenoble Alpes - IUT2 - Département informatique
Module 4103C

TP 2 : JavaScript, Evénements & DOM - Partie 1 (2H)

Objectifs du TP :

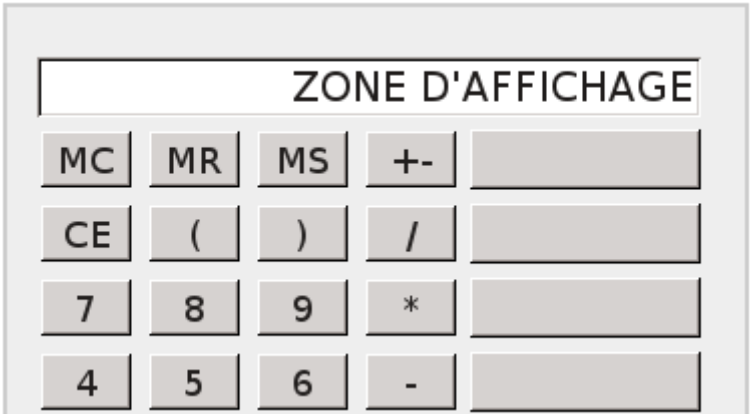
- Développer sa connaissance des événements pouvant potentiellement survenir sur des éléments HTML
- Savoir associer des fonctions JavaScript aux différents événements
- Savoir utiliser l'API DOM pour modifier les pages web dynamiquement

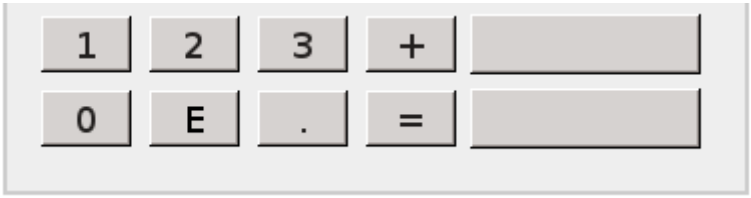
Ce document est consultable sur l'intranet du département.

Les sites suivants : http://www.w3schools.com/js/js_events.asp et http://www.w3schools.com/jsref/dom_obj_event.asp peuvent vous être utiles dans cet apprentissage

0. Récupération du HTML et CSS à animer

Afin d'illustrer différents aspects de la gestion des événements et de l'utilisation de l'API DOM nous allons développer une calculatrice paramétrable. Récuperez ici les fichiers **`calculatrice.html`**, **`calculatrice.css`** et **`calculatrice.js`** utiles pour la réalisation de ce TP. Si vous ouvrez le document HTML dans votre navigateur vous devriez visualiser la calculatrice ci-contre . En l'état, aucune gestion d'événement n'est réalisée et par conséquent aucun calcul ne peut être réalisé avec la calculatrice. Le clic sur les boutons de la calculatrice ne déclenche aucune action. Il est juste possible de saisir des nombres (en fait ce





que l'on veut) dans la zone d'affichage mais là pas de JavaScript, il s'agit du comportement par défaut d'un input de type text en html. Nous allons réaliser l'animation de la calculatrice en JavaScript

1. Implémentation des fonctions de base de la calculatrice

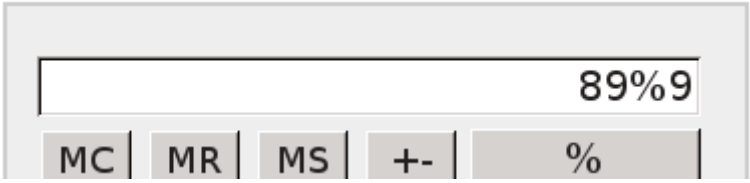
1. Modifiez `calculatrice.html` pour que le clic sur le bouton `CE` appelle une fonction nommée `rab`.
2. Dans `calculatrice.js`, écrivez la fonction `rab` qui remet la zone d'affichage à blanc. Indications: D'abord, on récupèrera l'élément html correspondant à la zone d'affichage par la fonction `getElementById` du DOM. Ensuite on affectera une chaine vide ("") à l'attribut `value` de l'élément récupéré **directement** sans passer par `setAttribute`. Testez en saisissant dans la zone d'affichage à l'aide du clavier une chaîne de caractères quelconque et en cliquant sur le bouton `CE`
3. Modifiez `calculatrice.html` pour que le clic sur le bouton `=` appelle une fonction `calcul`.
4. Dans `calculatrice.js`, écrivez la fonction `calcul` qui remplace le contenu de la zone d'affichage par l'évaluation de l'expression saisie dans la zone d'affichage. Afin de réaliser cette évaluation, on utilisera la fonction JavaScript prédéfinie `eval` qui permet l'évaluation des expressions arithmétiques. Testez en saisissant dans la zone d'affichage une expression arithmétique, par exemple `((5.9+56)-4)*89` et en cliquant sur le bouton `=` (sur l'exemple le résultat affiché doit être `5153.099999999999`). En vous inspirant de `http://www.w3schools.com/js/js_errors.asp` ajoutez la gestion des erreurs produites par `eval` (typiquement des erreurs de syntaxe) en affichant le message dans une boîte de dialogue (alert)
5. Afin de gérer l'affichage des symboles correspondant aux boutons cliqués, écrivez une fonction `affiche` qui prend en paramètre le bouton cliqué et qui concatène la valeur de l'attribut `value` du bouton cliqué, au contenu de la zone d'affichage
6. Afin d'associer cette fonction à l'événement `click` pour les boutons `(, ),0, 1, 2, 3, 4, 5, 6, 7, 8, 9, /, *, -, +, .` et les grands boutons sans symbole, nous n'allons pas procéder comme dans les 2 exercices précédents (il serait fastidieux de modifier 23 lignes de

`calculatrice.html`). Nous allons associer la fonction `affiche` aux différents boutons par programmation. Pour cela, dans `calculatrice.html` associez une fonction `init` à l'événement `onload` de l'élément `body` (vous pouvez vous inspirez du code suivant : http://www.w3schools.com/jsref/event_onload.asp). Dans `calculatrice.js`, écrivez la fonction `init` qui récupère les éléments ayant la classe `bouton_simple` avec la fonction `getElementsByClassName` et qui boucle sur ces éléments afin de leur ajouter l'attribut `onclick` avec la valeur `'affiche(this)'` (au moyen de la fonction `setAttribute`). Vérifiez la modification du DOM par l'inspecteur du web developper tool. On aurait pu associer l'événement à la fonction sans passer par la page web (voir cours). A ce stade, vous devez pouvoir effectuer des calculs par simples clics sur les boutons de votre calculatrice.

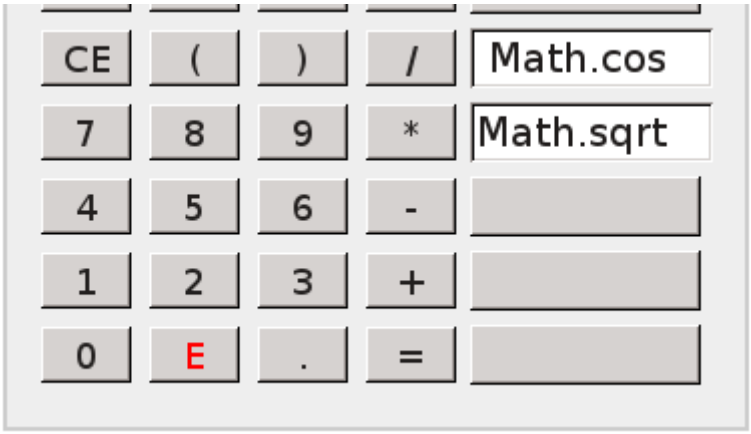
7. Le bouton `+-` est un peu différent, il ne suffit pas de recopier le symbole dans la zone d'affichage. Ecrivez la fonction `plusmoins` qui ajoute un "-" devant le contenu de la zone d'affichage s'il n'y en avait pas et qui l'enlève s'il y était (utilisez les fonctions de l'objet String). Modifiez `calculatrice.html` pour que le clic sur le bouton `+-` appelle la fonction `plusmoins`.
8. Implémentons maintenant les touches "mémoires" de la calculatrice. Faites en sorte que le clic sur le bouton `MS` appelle la fonction `range_memory` qui range le contenu de la zone d'affichage dans une variable globale `memory`
9. Faites en sorte que le clic sur le bouton `MR` appelle la fonction `affiche_memory` qui concatène le contenu de la variable globale `memory` au contenu de la zone d'affichage si la valeur de la variable est définie (sinon ne rien faire) (voir http://www.w3schools.com/jsref/jsref_undefined.asp)
10. Faites en sorte que le clic sur le bouton `MC` appelle la fonction `raz_memory()` qui affecte la valeur `undefined` à la variable globale `memory` (remise à blanc de la mémoire de la calculatrice). Voilà, votre calculatrice est maitenant complètement fonctionnelle. Vous pouvez effectuer des calculs, sauvegarder un résultat intermédiaire (clic sur `MS`) et le réutiliser (clic sur `MR`) dans un nouveau calcul

2. Boutons éditables

On veut rendre certains boutons de la calculatrice éditables (voir image ci-contre). Il s'agit des grands boutons situés à droite qui ne contiennent initialement aucun



symbole. On veut pouvoir y associer n'importe quel opérateur arithmétique ou fonction mathématique de l'objet `Math`. Par exemple : `%` ou `>>` (voir



http://www.w3schools.com/js/js_operators.asp et http://www.w3schools.com/js/js_comparisons.asp) ou `Math.round` ou `Math.sqrt` (voir http://www.w3schools.com/js/js_math.asp).

1. Afin de pouvoir éditer certains boutons de la calculatrice, le passage dans un mode "édition" est requis. Nous allons gérer le mode courant par une variable globale booléenne `edition`. Lorsque cette variable vaudra `true`, on sera on mode "édition" et lorsque que cette variable vaudra `false`, on sera en mode "calcul". Déclarez cette variable et initialisez la à `false` (au départ, il n'est pas possible d'éditer les boutons)
2. Faites en sorte que le clic sur le bouton `E` appelle une fonction `mode_edition` qui affecte la valeur `true` à la variable `edition` et mette le `E` du bouton en rouge (il suffit de modifier le style de l'élément, inspirez vous du code suivant : http://www.w3schools.com/js/js_htmldom_css.asp)
3. Compléter la fonction `mode_edition` pour qu'un nouveau clic sur le bouton `E` appelle cette fois-ci une fonction `mode_calcul`. Indications: Il faut remplacer la valeur de l'attribut `onclick` du bouton `E` par `mode_calcul()` avec `setAttribute`
4. Ecrivez la fonction `mode_calcul` qui permet de revenir à l'état initial. Cette fonction permet de repasser en mode calcul, remet le E du bouton en noir et associe de nouveau l'appel de la fonction `mode_edition` au clic sur le bouton `E`
5. A ce stade, lorsque l'on clique sur le bouton `E`, il passe en rouge et lorsque l'on reclique dessus il repasse en noir. Mais tout ceci ne nous permet pas toujours d'éditer les boutons. Pour permettre l'édition, dans la fonction `mode_edition`, supprimez l'attribut `onclick` (avec `RemoveAttribute`) et ajoutez l'attribut `ondblclick` (avec `setAttribute`) sur tous les boutons de la colonne de droite (ils ont la classe `bouton_libre`) pour qu'un double-clic sur ces boutons appelle la fonction `edit` avec pour argument le bouton double-cliqué (Pensez à

faire une boucle sur les éléments de la classe `bouton_libre` récupérés avec la fonction `getElementsByClassName`)

6. Compléter la fonction `mode_calcul` pour défaire les actions de la question précédente (clic-->affiche et pas de double-clic)
7. Ecrivez la fonction `edit` qui change de `button` à `text` le type de l'élément passé en argument (avec `setAttribute`)
8. Compléter la fonction `edit` pour qu'un double-clic appelle la fonction `fix` qui le retransforme en bouton
9. Compléter la fonction `fix` pour qu'un double-clic appelle la fonction `edit` qui le retransforme en champ texte
10. Compléter la fonction `mode_calcul` pour que tous les éventuels champs texte de la colonne de droite redeviennent des boutons lorsque l'on repasse en mode calcul. Testez l'édition de bouton, en essayant par exemple, `%` sur un bouton et `Math.round` sur un autre

3. Changement de place de la zone d'affichage

Faites en sorte qu'un double-clic sur la zone d'affichage la déplace en bas de la calculatrice comme dans l'image ci-contre (il suffira de placer l'élément comme dernier enfant de l'élément `<div id="calc">`). Un double clic sur la zone d'affichage devra aussi la replacer en haut. Aidez-vous des exemples et explications données dans http://www.w3schools.com/jsref/dom_obj_all.asp