

TP JS 1 - Premiers pas en JS

Décompressez l'archive déposée sur Moodle pour ce TP. Le dossier résultant contient différents fichiers à réutiliser ou à compléter. Pensez à consulter le site [MDN](#), et pour visualiser ce qui est attendu, ce [démonstrateur](#).

Exercice 1. Console, alertes, boîtes d'invite

Lancez Firefox, chargez le fichier **console.html**, ouvrez la console du navigateur et mettez-vous en mode d'édition multi-lignes. Exécutez ensuite dans la console les instructions qui vous sont demandées pour chaque question (en les conservant toutes dans l'éditeur).

1. Créez une constante **x** stockant la chaîne de caractères "15" et l'affichez dans la console.
2. Re-exécutez vos instructions. Qu'observez-vous ? Rechargez la page. Re-exécutez vos instructions. Qu'observez-vous ?
3. Affectez la valeur **true** à **x**. Qu'observez-vous ?
4. Commentez cette dernière instruction.
5. Transformez votre constante **x** en une variable.
6. Affichez l'invite "Saisir une valeur" dont le champ de saisie affiche la valeur de **x** par défaut. Stockez la réponse du visiteur dans une nouvelle variable **y** dont vous afficherez et le type et la valeur dans la console. Testez les 3 cas possibles : **Annuler**, **OK** en conservant la valeur par défaut, **OK** en changeant la valeur par défaut. L'affichage attendu est illustré en Figure 1.

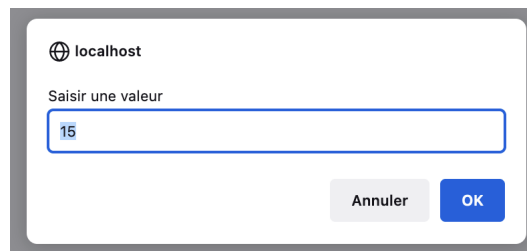


FIGURE 1 – Affichage de l'invite

Exercice 2. Débogueur

1. Lancez Visual Studio. Copiez-collez les instructions de l'exercice précédent dans le fichier fourni **./debugger/debugger.js**. L'importez dans **debugger/debugger.html** et chargez ce dernier dans le navigateur.
2. Faites en sorte que le paragraphe encodé dans le fichier HTML s'affiche après que le visiteur ait renseigné sa réponse dans l'invite, et non pas avant.
3. Placez-vous dans le débogueur de Firefox et y affichez le fichier **debugger.js**. Placez un point d'arrêt sur la première ligne. Rechargez la page puis exécutez le script pas à pas en cliquant sur la deuxième flèche (**Passez la fonction**) du panneau droit du débogueur. Observez les valeurs que prennent vos variables **x** et **y** à chaque pas (en les survolant à la souris, ou bien en consultant la rubrique **Portées**, objet **Window**, qui s'affiche automatiquement dans le panneau droit).
4. Supprimez le point d'arrêt sur la première ligne et en placez-un sur l'instruction affichant l'alerte contenant **x**. Rechargez la page. Au point d'arrêt, modifiez la valeur de **x** dans la console et poursuivez l'exécution en pas à pas. Qu'observez-vous ?
5. Dans la console, tapez **window.** puis cherchez **x** parmi les propriétés suggérées, complétez en choisissant **x** et exécutez. Qu'en déduisez-vous ? Modifiez la valeur de **y** via l'objet **window** dans la console.

Exercice 3. Conversion chaîne-entier

prompt.html importe **prompt.js** et affiche le résultat d'un appel à la fonction **prompt_positive_number()**.

1. Implémentez cette fonction qui demande un nombre au visiteur et convertit la chaîne saisie en nombre entier en utilisant **parseInt**. La fonction doit vérifier qu'il s'agit bien d'un nombre et qu'il est positif avant de le renvoyer. Elle réitère la demande dans le cas contraire.

2. Ajoutez un paramètre d'entrée à la fonction dénotant la base de représentation des entiers utilisable par le visiteur (2, 8, 10, etc.). Modifiez le corps de la fonction pour que la conversion puisse fonctionner selon la base passée en argument (p. ex. le visiteur devra saisir un nombre octal si la base vaut 8). Étendez **prompt.html** pour afficher le résultat d'un appel en base 2 dans un titre de deuxième niveau. Testez. Étendez **prompt.html** pour afficher le résultat d'un appel en base 8 dans un titre de troisième niveau. Testez. L'affichage attendu est illustré en Figure 2.

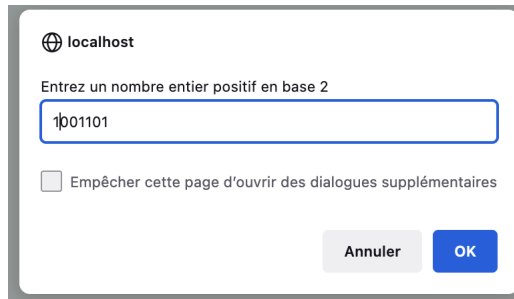


FIGURE 2 – Affichage de l'invite

Exercice 4. Chaînes

toNumber.html importe **toNumber.js** et affiche le résultat de différents appels à la fonction **toNumber()**. Cette dernière attend en entrée une chaîne de caractères et renvoie le nombre entier (positif ou négatif) correspondant. Elle doit donc vérifier que la chaîne communiquée correspond à la représentation décimale d'un entier. Dans le cas contraire, elle renvoie la valeur **NaN**.

1. Implémentez cette fonction sans utiliser **parseInt** et en vous aidant de l'opérateur **typeof** et des méthodes **String.prototype.charAt** et **String.prototype.charCodeAt**. L'affichage attendu est illustré en Figure 3.

toNumber

```
toNumber('234') 234 (number)
toNumber('-11') -11 (number)
toNumber('-3.1415') NaN (number)
toNumber('ab') NaN (number)
parseInt('123ab') 123 (number)
parseInt('[0]') NaN (number)
parseInt(/^123/) NaN (number)
```

FIGURE 3 – Conversion de différentes chaînes