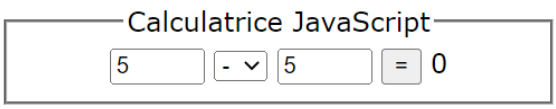
Travaux pratiques

Exercice 1

1. Créer la page index.html qui correspond à l'affichage suivant :



2. Créer un fichier nommé calculette.js qui contient une fonction nommée calculer permettant de faire l'addition, la soustraction, la multiplication et la division des deux entiers introduits par l'utilisateur.

Exercice 2

1. Créer la page index.html qui correspond à l'affichage suivant :

2. Ecrire une procédure JavaScript calculerEnF() qui permet de convertir une température en degré Celsius introduite par l'utilisateur en degrés Fahrenheit.

La formule de conversion : 9 / 5 * tempC + 32

3. Ecrire une procédure JavaScript calculerEnC() qui permet de convertir une température en degré Fahrenheit introduite par l'utilisateur en degrés Celsius.

La formule de conversion : (5 / 9) * (tempF - 32)

Exercice 3

Pour ce jeu Taquin :

- Créer un répertoire nommé **Taquin**
- Dans ce répertoire, créer la page **index.html** qui correspond à l'affichage suivant :

Chaque case est un bouton HTML.

- Créer un répertoire nommé css dans le répertoire Taquin.
- Créer un fichier nommé **style.css** à ces boutons dans le répertoire **css**.
- Créer la fonction move qui prend comme paramètres la ligne et la colonne de la case à échanger avec la case vide dans un fichier nommé **script.js** dans le répertoire **js** .

Question Bonus : Modifier le code de sorte de compter et afficher le nombre de déplacements effectués.

Exercice 4

Jeu de nombre aléatoire :

1. Créer la page **index.html** qui correspond à l'affichage suivant :

2. Créer un **script.js** qui permet de générer un nombre aléatoire et vérifier le nombre introduite par le joueur à l'aide de la fonction verification() qui ne prend rien en paramètre.
3. Le joueur doit taper un nombre dans la zone de saisi où le focus doit être placé. Pour chaque essai, un message est affiché qui indique si le nombre à trouver est plus grand ou plus petit. Si le joueur réussit à trouver le nombre, un message est affiché indiquant le nombre de coups d'essai.

Exercice 5

L'objectif principal est de vérifier l'intégrité des données envoyées par le formulaire vers le serveur. Il serait inutile de surcharger le serveur avec l'envoi de données manquantes, incomplètes ou incorrectes. Cette vérification peut se faire côté client par le langage Javascript. Lorsqu'on clique sur le bouton Valider, la fonction verifierFormulaire() est appelée pour réaliser le contrôle de :

- saisie sur un champ texte
- numéricité d'une saisie dans un champs texte
- caractères alphabétiques d'une saisie dans un champs texte
- caractères alphabétiques et numériques d'une saisie dans un champs texte
- longueur d'une saisie dans un champs texte
- saisie sur une adresse e-mail
- un choix par bouton radio

Une fois les données du formulaire contrôlées (toutes les informations sont présentes et cohérentes), l'envoi peut être effectué.

Exercice 6

1. Définir un objet nommé **membre**, chaque membre ayant un identifiant (id), un nom, un prénom et un grade. On doit pouvoir créer un membre à l'aide d'une fonction constructrice avec ses différentes propriétés. L'objet membre a également une méthode toString personnalisée qui permet d'afficher ses propriétés.
2. Définir un objet nommé **equipe** qui contient des membres.
3. L'objet équipe possède une méthode nommée **ajouter** qui permet d'ajouter un membre à l'**equipe**
4. Afficher les membres de **l'équipe** en utilisant la fonction toString de **membre**

```
ID: 1, Nom: Alain, Prenom: Casali, Grade: MCF
ID: 2, Nom: Alexandra, Prenom: SALOU, Grade: ATER
ID: 3, Nom: Frédéric, Prenom: Flouvat, Grade: MCF
ID: 4, Nom: Lotfi, Prenom: Lakhal, Grade: Pr
ID: 5, Nom: Safa, Prenom: Yahi, Grade: MCF
ID: 6, Nom: Rim, Prenom: Jouini, Grade: ATER
ID: 7, Nom: Vincent, Prenom: Risch, Grade: MCF
```

Exercice 7

1. Réaliser une classe nommée **Commune**, chaque commune ayant un code INSEE, un code postal, un nom (département), une région, une superficie et un code département. On doit pouvoir créer une commune avec ses différentes propriétés. Il faut également avoir des accesseurs.
2. Créer une méthode nommée **afficheCommune** pour afficher les différentes propriétés de l'objet est sollicitée
3. La classe **CommuneDetails** hérite de la classe parente **Commune** tous les attributs et dispose d'autres attributs supplémentaires code commune, code canton, code arrondissement, code région et d'une méthode d'affichage spécifique (**afficheCommuneDetails**) intégrant en plus ces nouveaux attributs.
4. Exploiter ces deux classes en instanciant un objet de la classe **Commune** et un objet de la classe **CommuneDetails** .
5. Ajouter les méthodes nécessaires qui permettent d'accéder et de modifier les valeurs des attributs instanciés.
6. Adapter la classe **Commune**, de manière à compter le nombre d'instanciations réalisées.