Le Jeu des Additions

Auteur : Jérôme CRECY.
Date: 30 Mars 2020.

Durée Estimée : 8 heures.

1. Les Notions nécessaires à la réalistaion de cet Exercice.

Certaines notions utilisées dans cet exercices ont déjà été abrodées d'autres sont nouvelle.

Je vous invite à , peut-être, passer un peu plus de temps sur les nouvelles notions, mais aussi à ne pas hésiter à consulter la documentation des notions déjà présentées. Elle peut vous apporter des indications et des éléments de compréhension supplémentaires.

1.1 Construction d'objet

En Javascript tout est Objet. Le modèle Objet de Javascript ne repose pas, comme beaucoup d'autres langages, sur des **classes** mais sur des **prototypes**.

Il est à noter que les classes ont, récement (2015), été introduites en javascript, leur utilisation est pour l'heure quasi-nulle. Beaucoup de développeurs ne le savent même pas. Je n'en parlerai pas dans ce document mais vous pouvez trouver des infos sur la MDN(https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Classes).

> Vous trouverez plus d'information sur la mdn https://developer.mozilla.org/fr/docs/Learn/JavaScript/Objects/Basics ou sur le le site de Pierre-Giraud https://www.pierre-giraud.com/javascript-apprendre-coder-cours/introduction-programmation-orientee-objet/.

Voici l'essentiel dont nous aurons besoin.

Il existe 2 façons (en fait plus mais pour nous ca suffira) de créer un objet en Javascript :

- Créer un objet javascript littéral.
- Créer un objet à partir d'un constructeur.

1.1.1 Objet littéral:

On peut créer directement un objet javascrit entre "{}" comme dans l'exemple suivant et l'affecter à une variable. On parle ici d'**objet javascript littéral**:

```
var contact={
    nom: "Dupont",
    prenom: "Pierre",

    nomComplet: function(){
        return this.nom+" "+this.prenom;
    },
```

```
8
9       direBonjour: function(){
10          console.log("Bonjour"+this.nomComplet());
11      }
12  };
13
14  contact.direBonjour();
```

Retour dans le console

```
1  Bonjour Dupont Pierre
```

Ici on a crée un objet **contact** avec des **propriétés** (les variables nom, prenom) et des **méthodes** (fonctions qui représentent des actions).

Vous remarquerez le mot clé **"this"** que l'on a déjà croisé et qui fait ici référence à l'objet au seing duquel il est utilisé.

1.1.2 Constructeur, prototype et instanciation.

Si l'on a besoin de plusieurs objets qui se ressemblent on va créer un **constructeur** qui est une fonction qui permettra d'**instancier** plusieurs objets ayant les mêmes propriétés et méthodes. Elle permettra de définir un **prototype**.

Trop de gros mots en une seule phrase... Reprenons notre exemple précédent mais, cette fois-ci, nous voulons gérer plusieurs contacts sans avoir à redéfinir les méthodes (fonctions) de chaque contact.

```
1  //déclaration du constructeur
2      /*** NB: La majuscule au constructeur n'est pas une obligation mais
   une convention ***/
3  function Contact(prenom, nom) {
4      this.prenom = prenom;
5      this.nom = nom;
6
7      this.nomComplet = function(){
8          return this.nom+" "+this.prenom;
9      }
10
11     this.direBonjour = function(){
12         console.log("Bonjour "+this.nomComplet());
13     }
14 }
15
16 //instanciation d'objets de type Contact
17
```

```
let contact1 = new Contact("Pierre","Druand");
let contact2 = new Contact("Julie","Martin")



//Utilisation ------------------
contact1.direBonjour();

contact2.direBonjour();
console.log("Mon contact 2 s'appelle "+contact2.nomComplet());

console.log("Julie épouse Pierre");
//on peut modifier ses propriétés ( ici on affect à contact2 le nom de contact1)
contact2.nom=contact1.nom;

console.log("Mon contact 2 s'appelle désormais "+contact2.nomComplet());
```

Résultat

```
Bonjour Druand Pierre
Bonjour Martin Julie
Mon contact 2 s'appelle Martin Julie
Julie épouse Pierre
Mon contact 2 s'appelle désormais Druand Julie
```

On peut agir directement sur le prototype d'un type d'objet et ainsi ajouter à toutes ses instances une propriété ou une methode. Le précédent exemple peut donc aussi s'écrire comme ainsi:

```
//déclaration du constructeur
function Contact(sPrenom, sNom) {
    this.prenom = sPrenom;
    this.nom = sNom;
}

//ajouts de méthodes au prototype de Contact
Contact.prototype.nomComplet = function(){
        return this.nom+" "+this.prenom;
}

Contact.prototype.direBonjour = function(){
        console.log("Bonjour "+this.nomComplet());
    }


//instanciation d'objets de type Contact

```

```
let contact1 = new Contact("Pierre","Druand");
let contact2 = new Contact("Julie","Martin")

//Utilisation
contact1.direBonjour();

contact2.direBonjour();
```

Résultat

```
Bonjour Druand Pierre
Bonjour Martin Julie
```

1.2. la création d'éléments.

Nous avons déjà vu ces notions. Il est possible de créer, déplacer et supprimer des éléments du DOM à l'aides de fonctions.

Nous utiliserons ici 2 fonctions :

- document.createElement : https://developer.mozilla.org/fr/docs/Web/API/Document/createElement
- Node.appendchild: https://developer.mozilla.org/fr/docs/Web/API/Node/appendChild

1.3. Les nombre au hasard avec l'objet Math.

L'objet Math possède une méthode qui permet de générer un nombre flottant entre 0 et 1. Si l'on souhaite un entier entre 0 et 10 il suffit de le multiplier par 10 et de l'arrondir.

Nous aurons ainsi le code suivant :

```
let nombreAuHasard=Math.round( Math.random * 10);
```

1.4. La gestion des événement avec addEventListener

La programmation en Javascript étant basée essentiellement sur de l'événementiel. Nous auron recours à la gestion des événements nottement au travers de la fonction addEventListener(https://developer.mozilla.org/fr/docs/Web/API/EventTarget/addEventListener).

> Vous pouvez retrouver des explications complètes sur la gestion des événements sur le site de Pierre Giraud : https://developer.mozilla.org/fr/docs/Web/API/EventTarget/addEventListener

Nous utiliserons toutefois ici un nouvel élément qui va nous permettre d'intercepter les frappes clavier ainsi nous pourrons choisir de ne prendre en compte que les frappes de valeur numérique. Il n'existe pas de fonction is_numeric en javascript toutefois la fonction isFinite() vérifie si un Number ou une String est nombre fini.

Ainsi, le code suivant écoute les pressions sur les touches du clavier et déclanche une fonction qui vérifie si ce sont bien des touches numériques.

```
function logNumeric(evenement)){
  if (isFinite(evenement.key)){
         console.log (evenement.key)
         return true;
  }
  return false;

}
addEventListener("keydown",logNumeric);
```

2. C'est Parti

> Il est conseillé tout au long de ce tutorial de ne pas faire de copier/coller et de bien prendre le temps à chaque étape.
>
> A chaque étape on retrouvera le code intégral.

2.1. La page de support HTML5

Pour débuter nous allons commencer par créer une simple page html avec juste un block *"id=playground"*.

Nous ne nous occupons pas ici du style de la page. C'est vous, le WebDesigner, qui l'habillerez par la suite (cf. 3.2.).

```html
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
        <title>Les additions</title>

    </head>

    <body>
        <main id="playground"></main>
    </body>

</html>
```

2.2. Création d'un objet AdditionGame et lancement

Nous allons à présent créer le constructeur AdditionGame et sa fonction de démarrage startGame.Mais avant d'aller plus loin observont le comportement de **this** en mettant le code suivant entre des balise javascript à l'intérieur du :

```javascript
function AdditionGame(containerId){

            console.log(this);

            this.startGame = function(){
                console.log(this);

            }

 }

 partie=new AdditionGame("playground");
 window.addEventListener("DOMContentLoaded",partie.startGame)
```

On s'attendrait à ce que this renvoie l'objet dans les 2 cas. Et bien non... Si le premier renvoie bien l'objet que nous venons de créer le second renvoie l'objet 'Window' car la fonction est appelée à partir d'un évenement affecté à l'objet 'window'.

Nous allons donc utiliser une astuce qui consiste à enregistrer à affecter this à une nouvelle variable fréquement appelée "self" mais on pourrait l'appeler comme on veut.

De plus on voudra, par la suite enregistrer le container en propriété. Il serait donc bien que le DOM soit chargé avant. c'est pour ça que l'on va créer notre objet partie dans une **fonction anonyme** qui se déclanchera sur l'événement "DOMContentLoaded".

Ainsi on obtiendra le code suivant :

```html
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Les additions</title>


        <script>

        function AdditionGame(domContainerId){

            //affectation de this à self pour être sur de faire référence à l'objet.
            let self = this;

            self.domContainerId = domContainerId;
            self.container = document.getElementById(domContainerId)

            self.startGame=function(){
               self.container.innerHTML="<h1>Bienvenue, le jeux des additions.</h1>";
            }

        }


        window.addEventListener(
                    "DOMContentLoaded",
                    function(){
                        partie=new AdditionGame("playground");
                        partie.startGame();
                    }
        );
        </script>


    </head>

    <body>
        <main id="playground"></main>
    </body>

</html>
```

2.3. Ajout d'un bouton en dessous du titre et création et affichage du calcul.

Nous allons à présent :

- ajouter un bouton qui va nous permettre de débuter la partie et de faire afficher la première question. Pour cela on va utiliser createElement. Modifier le innerHTML peut paraitre plus simple de prime abord, mais il n'en est rien car il nous faudra ajouter un événement sur ce bouton.
- ajouter une propriété "score" à notre objet et l'initialiser à 0 dans la methode startGame.
- scinder notre container en 2 partie:
  - scoreDiv dans laquelle on affichera le score
  - Blackboard pour afficher les calculs.

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Les additions</title>


        <script>

        function AdditionGame(domContainerId){

            //affectation de self à self qui permettra d'être toujours sur de faire référence à l'objet.
            let self = this;

            self.domContainerId = domContainerId;
            self.container = document.getElementById(domContainerId);

            //on va recréer 2 un blocks à l'intérieur du container  score et blackboard.

            self.scoreDiv=document.createElement("div");
            self.container.appendChild(self.scoreDiv);

            self.blackboard=document.createElement("div");
            self.container.appendChild(self.blackboard);




            self.score=0;

            self.a = null;
            self.b = null;

```

```
            self.updateScore=function(){
                self.scoreDiv.innerHTML="Score : "+ self.score;
            }

            self.startGame=function(){

                self.score=0;
                self.updateScore();

                self.blackboard.innerHTML="<h1>Bienvenue, le jeux des additions.</h1>";
                let btnStart = document.createElement("button");
                btnStart.innerText="C'est parti";
                btnStart.addEventListener("click",self.nextQuestion);
                self.blackboard.appendChild(btnStart);

            }

            self.nextQuestion=function(){



                self.a=Math.round(Math.random()*10);
                self.b=Math.round(Math.random()*10);

                self.blackboard.innerHTML=self.a+" + "+self.b+" = ";
            }
        }


        window.addEventListener(
                    "DOMContentLoaded",
                    function(){
                        partie=new AdditionGame("playground");
                        partie.startGame();
                    }
        );
        </script>


    </head>

    <body>
        <main id="playground"></main>
    </body>

</html>
```

2.4. Gestion de ma réponse.

Pour gérer la réponse on va ajouter :

- Un eventListener sur keypress.
- Une propriété reponse que l'on va affecter à notre objet et initialiser à ""
- Une fonction de validation de la réponse.

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Les additions</title>


        <script>

        function AdditionGame(domContainerId){

            //affectation de self à self qui permettra d'être toujours sur de faire référence à l'objet.
            let self = this;

            self.domContainerId = domContainerId;
            self.container = document.getElementById(domContainerId);

            //on va recréer 2 un blocks à l'intérieur du container  score et blackboard.

            self.scoreDiv=document.createElement("div");
            self.container.appendChild(self.scoreDiv);

            self.blackboard=document.createElement("div");
            self.container.appendChild(self.blackboard);




            self.score=0;

            self.a = null;
            self.b = null;
            self.reponse=null;
```

```
39              self.updateScore=function(){
40                  self.scoreDiv.innerHTML="Score : "+ self.score;
41              }
42
43              self.startGame=function(){
44
45                  self.score=0;
46                  self.updateScore();
47
48                  self.blackboard.innerHTML="<h1>Bienvenue, le jeux des
    additions.</h1>";
49                  let btnStart = document.createElement("button");
50                  btnStart.innerText="C'est parti";
51                  btnStart.addEventListener("click",self.nextQuestion);
52                  self.blackboard.appendChild(btnStart);
53
54              }
55
56              self.validerReponse=function(){
57
58                  //on supprime le listener sinon il continue de
    s'executer.
59                  removeEventListener("keydown",self.keyPressed);
60
61                  let reponseJuste=Number(self.a)+Number(self.b);
62                  console.log(reponseJuste);
63
64                  if (reponseJuste==Number(self.reponse)){
65                      self.blackboard.innerHTML='<p>Bravo !</p>';
66                      self.score+=1;
67                      self.updateScore();
68                  }
69                  else{
70                      self.blackboard.innerHTML='<p>Dommage <br> La bonne
    réponse était '+reponseJuste+" :-(</p>"
71                  }
72
73                  let btnSuivant = document.createElement("button");
74                  btnSuivant.innerText="Calcul suivant";
75                  btnSuivant.addEventListener("click",self.nextQuestion);
76                  self.blackboard.appendChild(btnSuivant);
77
78              }
79
80              self.keyPressed=function(e){
81                  console.log(e.key);
82
83                  e.preventDefault(); //sinon backspace renvoie à la page
    précédente
```

```
84
85                  //  si la valeur de la touche est un nombre fini.
86                  if (isFinite(e.key)){
87                      self.blackboard.innerHTML+=e.key;
88                      self.reponse+=e.key;
89                  }
90                  else if(e.key=="Enter"){
91                      self.validerReponse();
92                  }
93              }
94
95              self.nextQuestion=function(){
96
97
98                  self.reponse="";
99                  self.a=Math.round(Math.random()*10);
100                 self.b=Math.round(Math.random()*10);
101
102                 self.blackboard.innerHTML=self.a+" + "+self.b+" = ";
103
104                 addEventListener("keydown",self.keyPressed);
105
106             }
107
108
109         }
110
111
112         window.addEventListener(
113                     "DOMContentLoaded",
114                     function(){
115                         partie=new AdditionGame("playground");
116                         partie.startGame();
117                     }
118         );
119         </script>
120
121
122     </head>
123
124     <body>
125         <main id="playground"></main>
126     </body>
127
128 </html>
```

3. A vous de jouer

Nous avons un 'jeu de calcul' fonctionnel. Toutefois je ne pense pas que l'on attire beaucoup de candidats avec ça. Vous allez donc devoir l'améliorer.

Voici les donc les ameliorations qu'il va falloir apporter... La première est indispensable, mais les suivantes peuvent être réalisées dans l'ordre que vous souhaitez.

3.1. Limiter le nombre de Questions

Tout de suite le jeu tourne sans fin. Il faut ajouter une fin au bout d'un certain nombre de tours. 10 par exemple.

> **Aide:** Ajouter une variable "compteur" en propriéte de votre objet. Réinitialisez la à 0 au debut de chaque partie (dans la fonction AdditionGame.startGame).Incrémentez la de 1 au moment de la validation de la réponse. Si elle vaut 10 la partie est finie. On affiche alors le score et un commentaire.

3.2. Styler l'application.

Pour l'instant la présentation est austère. Il faut en faire un jeu attractif.

> **Aide** : 2 choses peuvent vous aider :
>
> - Ecrire le rendu html de votre page quant tout est ouvert
> - Pour 'linstant le container est scinder en 2 block mais il ne tient qu'a vous d'ajouter tout les éléement nécessaire à une bonne présentation.

3.3. Ajouter un pavé numérique.

Le mode de saisie avec le frappe au clavier fait que l'on ne peut pas facilement jouer sur tablette ou mobile.

L'intégration d'un pavé numerique serait un plus.

> **Aide** : Pour ajouter le pavé numérique, il suffit de céer un 3ème block à l'intérieur du container qui contiendra un ensemble de boutons sur lesquels on mettra un événement "click" qui declenchera une fonction alternative de *keyPressed* que l'on pourra appeler *btnPressed*.