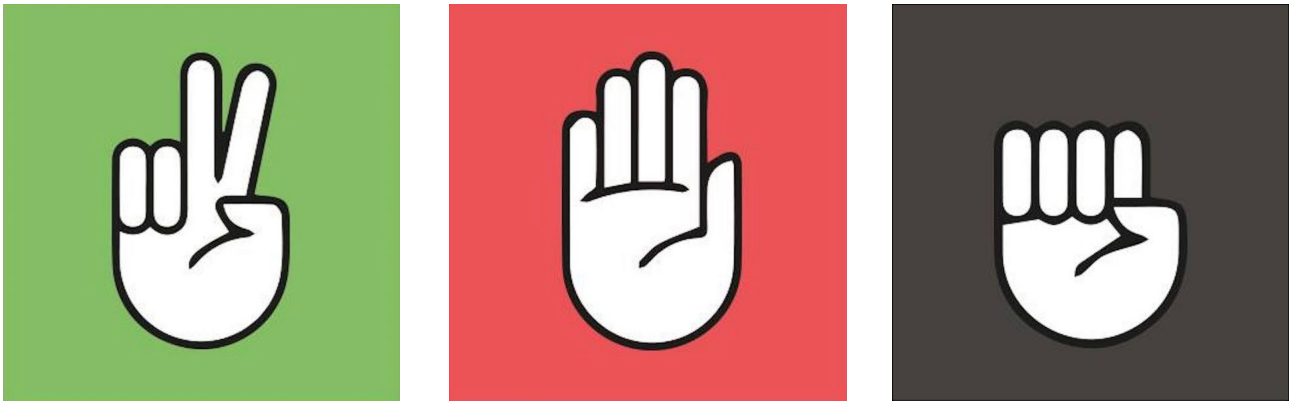
CODING CLUB

Pierre Feuille Ciseaux



Auteurs
Lucas Thevenard
Lucas Obadia

Introduction

L'objectif est de réaliser le célèbre jeu du "Pierre Feuille Ciseaux" (chifoumi pour les intimes) dans une version web. Pour réaliser ce jeu, nous allons utiliser le **Javascript** en plus du HTML/CSS.

Qu'est-ce que l'HTML et le CSS ?

L'HTML, acronyme signifiant « HyperText Markup Language », est un **langage de balisage** conçu pour représenter des pages web. On est aujourd'hui à la version 5 de l'html, noté HTML5.

Exemple d'une balise p en HTML

```
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p>
```

Dans l'exemple ci-dessus, on délimite la phrase « Lorem ipsum dolor sit amet, consectetuer adipiscing elit. » par la balise `p` (définissant un paragraphe) au moyen d'une balise d'ouverture et d'une balise de fermeture .

Le **CSS**, acronyme de « Cascading Style Sheets » est un langage informatique qui décrit la présentation des documents html. Concrètement, l'html est chargé du contenu de la page, et le CSS est chargé de **mettre en forme le contenu**. Actuellement, on est à la version 3 de css, noté CSS3

Exemple d'une feuille de style qui mettra la couleur rouge sur le texte de toutes les balises p

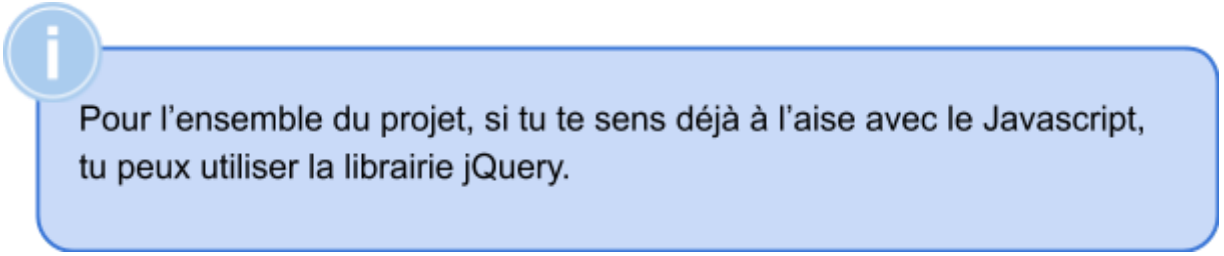
```
p {
  color: red;
}
```

Qu’est-ce que le Javascript ?

Le Javascript est un **langage de programmation**, c’est une forme de code qui permet de dicter à l’ordinateur quoi faire. On trouve la majorité du code Javascript dans les **pages web** pour les rendre **dynamiques** / interactives, même si vous ne le voyez pas s’afficher. En effet, c’est le seul langage qui permet de dicter à un navigateur web (Internet Explorer, Chrome, Firefox…) ce qu’il doit faire sans rien installer préalablement. On peut dire que la majorité des navigateurs web “parlent” le Javascript.

Voici des exemples de ce que le Javascript nous permet de faire

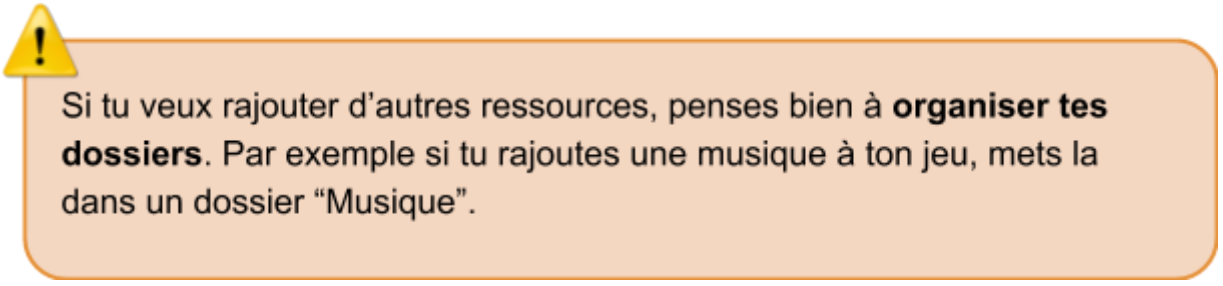
- Faire bouger, apparaitre, disparaitre des éléments de la page (titre, menu, image…)
- Mettre à jour des éléments de la page sans la recharger
- Faire une action spécifique au moment d’un événement de l’utilisateur (cliquer sur un élément, entrée clavier, bouger la souris…)



Environnement de travail

Nous t'avons donné préalablement une clé usb avec un dossier **Pierre Feuille Ciseaux** qui contient un fichier html, un fichier css, et un dossier img contenant toutes les images de ton site web.

A la racine de ce dossier, commence par créer un fichier Javascript qui te permettra de coder la logique de ton jeu. Un fichier javascript est un fichier dont l'extension est **.js**, tu peux évidemment nommer ton fichier comme tu veux tant que tu respecte l'extension.



Pour éditer ton fichier Javascript, on te recommande d'utiliser un IDE, tu peux voir ça comme un gros éditeur de texte avec plein d'outils spécialement pensés par et pour les développeurs. Il en existe plusieurs tels que Visual Studio Code, Sublime Text ou encore WebStorm. Tu peux très bien utiliser aussi un éditeur de texte classique, mais ton expérience en tant que développeur sera biaisé.

Maintenant que tu as créé le fichier Javascript, il faut l'importer depuis le fichier html. Les imports js se font toujours à la fin de la balise **body** du fichier html. Je te laisse compléter cette ligne du fichier html en ajoutant le nom de ton fichier dans **src** :

```
<script src="monfichier.js" type="text/javascript"></script>
```

Sans cet import, ton fichier javascript ne sera jamais pris en compte dans ta page web.

Crée ton jeu

Scénario

L'utilisateur joue contre l'ordinateur, le premier en 10 points gagne ! Tour par tour, le joueur et l'ordinateur sélectionnent un des trois coups possibles (Pierre | Feuille | Ciseaux). La pierre bat les ciseaux, les ciseaux battent la feuille, la feuille bat la pierre. A chaque coup réussi, le gagnant récupère un point.

Les composants

Je t'invite à ouvrir le fichier index.html dans un navigateur web. On peut diviser la page en plusieurs partie

Header avec un titre (1)
Scoreboard (2)
Message de statut (3)
Boutons pour sélectionner le coup joué (4)

Les variables

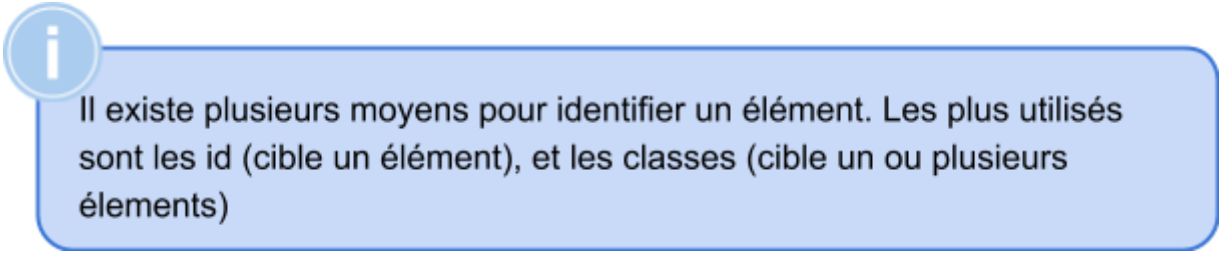
On va stocker dans des variables les éléments de notre page. Dans la page html, on a donné des **id** sur les **div**. Par exemple la div responsable d'afficher le bouton de la pierre, a pour id "p".

Un id permet d'identifier, grâce au nom donné, **un et un seul élément** de la page web. Voici comment récupérer un élément html en Javascript :

```
var maVariable = document.querySelector("#monId"); // on met un # pour préciser que c'est un id
var maVariable = document.getElementById('p'); // Autre façon de le faire
```

La variable **document** est une variable globale qui existe dès lors le chargement de votre page, et qui stocke l'ensemble de votre page web sous forme d'objet. On peut donc récupérer à partir de cette variable, n'importe quel élément de notre page, n'est-ce pas magnifique ?

Cette variable a ce qu'on appelle des **méthodes** (fonctions), telle que **querySelector** qui est responsable de retourner l'élément cibler grâce à l'id par exemple.



Dans le fichier js, crée une fonction avec le nom que tu veux et appelle cette fonction. Dans cette fonction, crée **3 variables** qui vont contenir respectivement :

- L'élément html qui a comme id 'p'
- L'élément html qui a comme id 'f'
- L'élément html qui a comme id 'c'

On veut aussi récupérer l'élément html qui contient le message de statut qui a la **classe "resultat"**. Stock la dans une variable, cette variable ne doit pas être dans une fonction, c'est ce qu'on appelle des **variables globales**, elles pourront être utilisées dans n'importe quelle fonction.

De la même façon, crée 3 variables globales dont 2 qui vont être chargées de compter le score du joueur et de l'ordinateur. Assigne la valeur 0 à ces deux variables.
La 3e variable est un **objet javascript** mettant en relation le code du coup joué avec le nom complet (p → Pierre | f → Feuille | c → Ciseaux)

Tu dois avoir quelque chose comme ceci :

```
// Variables globales
var resultat = document.querySelector(".resultat > p");
var choix = { "p": "Pierre", "f": "Feuille", "c": "Ciseaux" };
var joueur_score = 0;
var computer_score = 0;

function main() {
   const pierre = document.getElementById('p');
   const feuille = document.getElementById('f');
   const ciseaux = document.getElementById('c');
}

main(); // Execute la fonction 'selection'
```

Un bon développeur est un développeur qui ne cesse de se documenter. Google est ton ami !

Les évènements

On appelle un événement toutes interactions entre l'utilisateur et le jeu. Par exemple si le joueur clique sur le bouton "Ciseaux", cela va déclencher un événement **'click'**. A nous d'en faire ce que l'on veut, on peut soit l'ignorer, soit récupérer l'élément html de l'élément cliqué, et bien d'autres choses encore..

Crée une fonction que tu vas appeler **game** et qui prend en paramètre **user** (user sera le choix du coup joué de l'utilisateur). Dans la fonction game rajoute juste cette ligne :

```
console.log(user); // Print dans la console du navigateur le parametre user
```

Dans la première fonction que tu as créé, en dessous des 3 variables (pierre, feuille, ciseaux), fait en sorte que quand on clique sur l'un des trois boutons que ça appelle la fonction game() en passant en paramètre le code du choix cliqué.

Bouton cliqué	Pierre	Feuille	Ciseaux
Paramètre	p	f	c

Retourne sur ton jeu et ouvre la console du navigateur. Tu peux maintenant voir que si tu clique sur le bouton Feuille, le 'f' s'affiche bel et bien.

Choix de l’ordinateur

Maintenant que nous avons récupéré le choix de l’utilisateur, nous devons aussi faire la même chose avec l’ordinateur.

Crée une fonction **computerChoix** qui va être responsable de te renvoyer **au hasard** le coup choisis par l’ordinateur et ce, au même format que le choix de l’utilisateur (‘p’, ‘f’, ‘c’) et appelle cette fonction au début de ta fonction game. Pour générer aléatoirement un coup joué, il faut que tu crée un tableau contenant les 3 choix possibles, soit : ‘p’ ; ‘f’; ‘c’.

Une fois qu’on a le tableau des choix, il faut que tu génère **un nombre aléatoire** entre [0;2]. Pourquoi entre 0 et 2 ? A cause du comportement d’un tableau en informatique.

Un **tableau** permet de stocker dans chaque case du tableau une valeur qu’on peut récupérer individuellement par la suite. Dans notre cas, la taille de notre tableau vaut 3. Il y a tout de même une petite particularité, les ordinateurs et donc les **tableaux commencent les décomptes à 0**.

Je veux le...	1	2	3
Je dois utiliser l’index...	0	1	2

Prenons un exemple concret avec notre tableau des choix, pour accéder à la case contenant le choix ‘f’, nous devons écrire :

```
choix[1] // Renvoie ‘f’
```

C’est pourquoi, on doit générer un nombre entre 0 et 2 :)

Il ne te reste plus qu’à renvoyer la valeur du tableau à l’index du nombre généré, et la fonction computerChoix remplira son rôle !

Combinaison et comparaison

Bien joué ! Nous avons maintenant grâce à toi, le coup de l'utilisateur et le coup de l'ordinateur. Qu'allons nous en faire ? Les comparer et voir qui a la main gagnante !

Dans la fonction game, tu devrais avoir ça :

```
function game(user) {
  var computer = computerChoix(); // Récupérer le choix de l'ordinateur(p/f/c)
}
```

Crée une nouvelle variable qui va **concaténer** le choix de l'utilisateur et le choix de l'ordinateur (respecte bien l'ordre). Grâce à cette nouvelle variable, nous allons comparer toutes les combinaisons pour voir qui à la main gagnante. Mais avant cela, crée 3 fonctions qui prennent 2 paramètres (le choix user et le choix computer) qui s'appellent respectivement → win ; lose ; par

Puisqu'il y a 3 fins possibles à chaque coup joué (gagné, perdu, égalité), nous allons en fonction de la combinaison des coups joués, appeler telle ou telle fonction.

Dans la fonction game, utilise les **conditions** pour, en cas de victoire pour l'utilisateur, appeler la fonction **win** ; En cas de défaite, la fonction **lose** ; Et en cas d'égalité, la fonction **par**. N'oublies pas d'envoyer les 2 paramètres attendus par ces fonctions ;)

Voici l'exemple d'une combinaison gagnante pour le joueur → **fp**

Win | Lose | Par

Nous savons à présent si le joueur est dans un cas de victoire, de défaite ou d'égalité face à l'ordinateur. En fonction de ça, tu dois réfléchir aux changements que cela implique. Il faut se poser les bonnes questions :

- Dans quel cas je donne un point et surtout à qui je donne le point ?
- Comment incrémenter la valeur que l'on voit afficher dans le scoreboard ?
- Comment mettre à jour le message de statut pour qu'il soit de la forme : "*choixUser* contre *choixComputer* : PERDU" ?

Code les fonctions *win*, *lose* et *par*, qui, comme tu l'aura deviné, doit répondre aux questions précédentes. Les variables globales créées tout à leur seront surement utilisées dans ces fonctions… ;)

Fin de jeu

Notre jeu fonctionne ! Mais à aucun moment il ne s'arrête, on est condamné à vie à jouer contre l'ordinateur ! 😱

Il faut donc dire à notre programme que si le joueur ou l'ordinateur atteint 10 points, on affiche une alerte pour dire "Victoire !" si l'utilisateur a gagné ou "Défaite !" si l'utilisateur a perdu. Une fois l'alerte affichée, il faut recharger la page (en javascript) pour redémarrer une partie.

Pour aller plus loin...

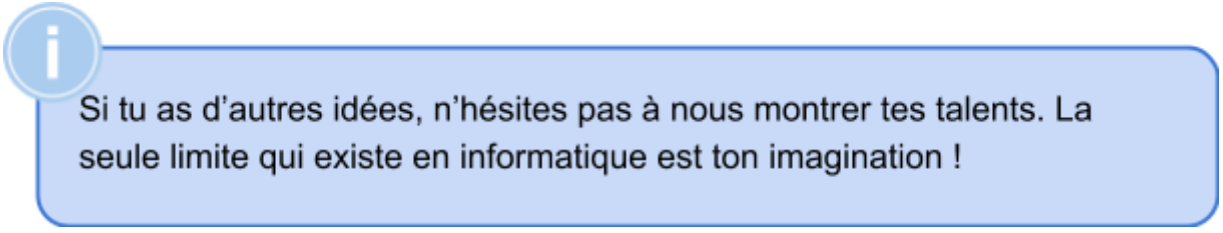
La session n'est pas encore finie et tu as déjà fini le jeu ?! Tu as gagné le droit de souffler un bon coup et si tu le souhaites d'aider les autres :)

Si tu veux améliorer ton jeu, voici quelques idées que tu peux mettre en place:

- Il manque un peu de musique tu trouves pas.. ? ;)

- Quand tu clique sur l'un des 3 boutons, affiche un cercle de couleur pendant 0.5 seconde au niveau du bouton cliqué pour indiquer le résultat. Vert si tu as gagné, rouge si tu a perdu, gris si tu as fais égalité. Des classes ont déjà été créées dans le css pour cet effet dans la partie "Définition des bordures de cercles".

- Mettre son pseudo au début du jeu et utiliser le pseudo dans l'alerte à la fin, par exemple "Emma, tu as gagné !"



[Lien Github Chifoumi]