

## Modèle de copie : Dynamiser vos sites web avec Javascript



**GDWFSDVSWEBAJAVAEXAIII1A**

**Ceci est un modèle de copie. N'oubliez pas de renseigner vos prénom/nom, ainsi que le nom et le lien vers le projet.**

**Vous pouvez bien sûr agrandir les cadres pour répondre aux questions sur la description du projet si nécessaire.**

**Prénom :** Rémi

**Nom :** FAURE

**ATTENTION ! PENSEZ À RENSEIGNER VOS NOM ET PRÉNOM DANS LE TITRE DE VOS FICHIERS / PROJETS !**

Nom du projet : Dice

Lien Github du projet : <https://github.com/rf33350/dice>

Lien Drive du projet (si nécessaire) : .....

URL du site (si vous avez mis votre projet en ligne) : <https://dice-rf-study.netlify.app/>

### Description du projet

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions. Dans cette rubrique, le jury cherche à voir comment vous procédez : comment vous organisez votre travail, comment vous réalisez concrètement la tâche ou l'opération pas à pas.  
Utiliser un langage professionnel. Employez-le « je », car vous parlez en votre nom. Vous pouvez écrire au temps présent.

Pour coder ce projet d'application web de jeu de dés, je me suis organisé de la manière suivante. En premier lieu, j'ai codé le html ainsi que la partie sommaire du CSS pour placer les éléments et ainsi avoir un rendu ressemblant à la vue donnée dans l'énoncé de l'évaluation. Une fois un premier visuel obtenu, je me suis consacré au développement du code javascript en charge de dynamiser les différents éléments du site J'ai traduit les règles données en algorithme, puis je me suis mis à coder chaque fonction pas à pas en prenant soin de les tester une à une, avant de tester le jeu en entier.  
Pour finir j'ai optimisé le CSS afin d'obtenir un rendu « responsive » et « mobile-first ».



## 1- Le HTML :

A partir d'un squelette vierge html (raccourci « ! » avec Emmet) j'ai renseigné les balises <meta> du <head>. J'ai ainsi renseigné le titre du site, l'auteur du site ainsi que sa description :

```
<meta name="description" content="Un site de jeu de dés">
<meta name="author" content="Rémi FAURE">
<title>Dice</title>
```

J'ai également ajouté un favicon représentant des dés :

```
<link rel="icon" href="/assets/img/dice.ico">
```

J'ai ensuite ajouté les feuilles de styles CSS :

```
<link href="/assets/css/bootstrap.min.css" rel="stylesheet">
<link href="/assets/css/style.css" rel="stylesheet">
```

J'ai utilisé le Framework CSS Bootstrap pour l'agencement des éléments du HTML en grille. J'ai également créé un fichier CSS « style.css » pour personnaliser l'application.

J'ai également déclaré la police d'écriture « Lato » tel qu'il a été demandé dans l'énoncé :

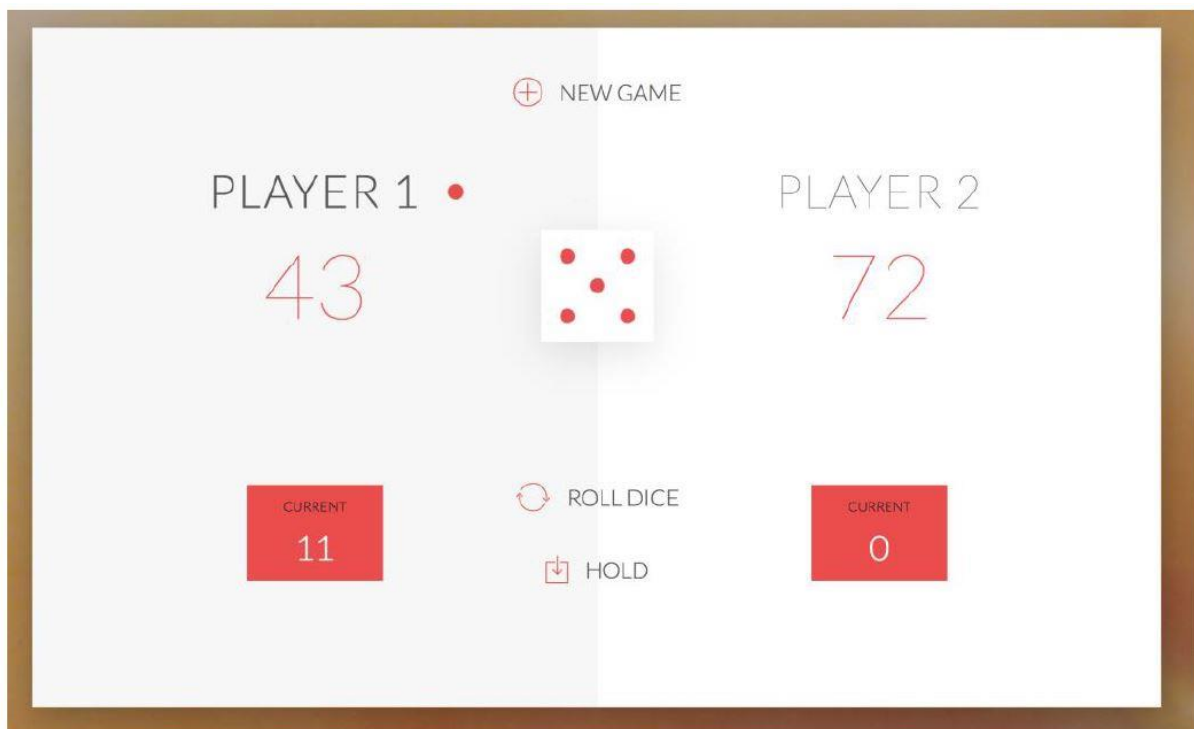
```
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Open+Sans&family=Raleway:ital,wght@0,400;0,600;1,300&display=swap" rel="stylesheet">
```

J'ai suivi les indications données sur le site de google fonts :

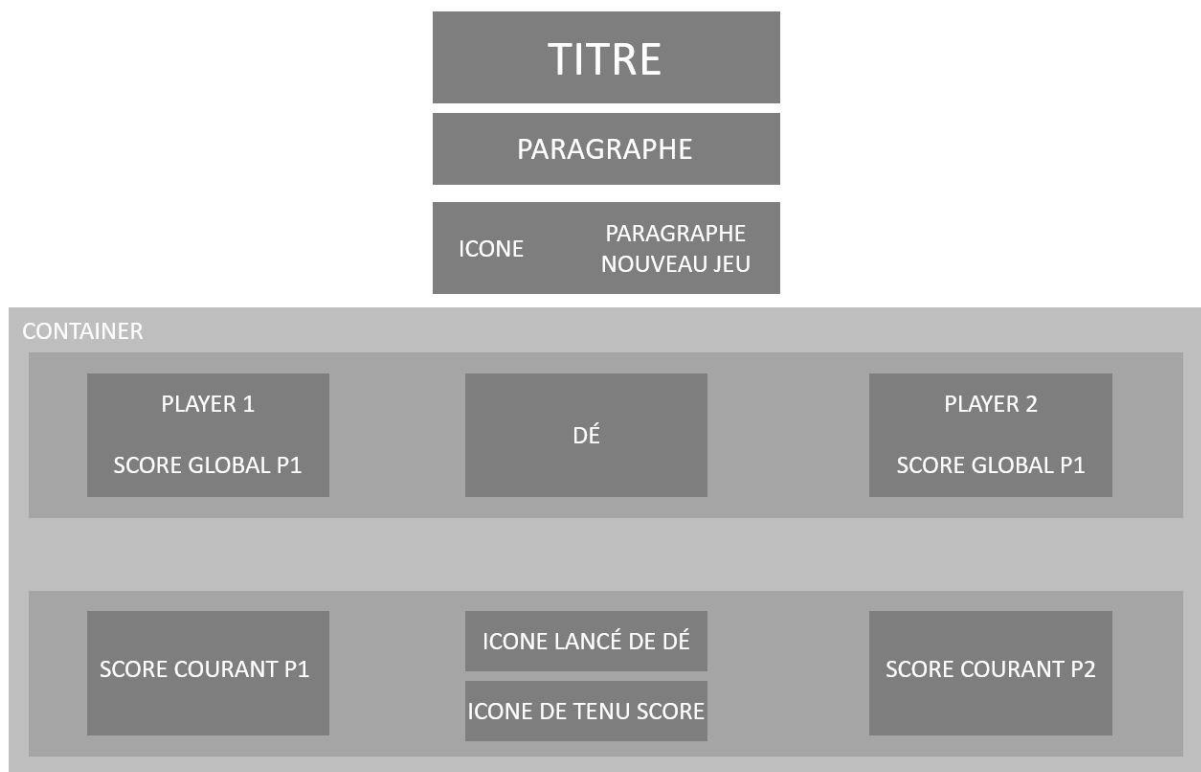
<https://fonts.google.com/specimen/Lato>

Dans le <body> du site, voulant respecter la charte graphique de l'énoncé.

Je suis parti de l'énoncé graphique :



J'ai découpé le visuel en briques élémentaires :



J'ai suivi ce schéma pour construire le cœur du HTML. Je décris la manière dont j'ai procédé pour traduire ce schéma en HTL :

J'ai créé un titre <h1> :

```
<h1>Dice</h1>
```

J'ai ensuite ajouté un paragraphe de sous-titre :

```
<p class="main-p">Jeu de dés</p>
```

J'ai ensuite implémenté une <div> de manière à intégrer une icône et un texte qui servira à démarrer une partie :

```
<div class="new_game_container">
  
  <p class="new_game_par">NEW GAME</p>
</div>
```

A noter que j'ai ajouté un attribut « id » qui me servira pour l'appel de cet élément par le code javascript. Il en sera de même pour tous les éléments de ce code HTML comportant cet attribut.

J'ai ensuite intégré les différents éléments visuels du jeu. Pour cela je me suis aidé du Framework Bootstrap et notamment de sa grille. Comme le schéma précédent le montre, j'ai décidé de découper les éléments en 2 lignes et 3 colonnes chacune, ce qui donne dans le HTML :

```
<div class="container">
  <div class="row top_container">
    <div class="col-4 player1">...
  </div>
  <div class="col-4 dice">...
  </div>
  <div class="col-4 player2">...
  </div>
</div>
<div class="row low_container">
  <div class="col-4 low_container_player1_container">...
  </div>
  <div class="col-4 actions_container">...
  </div>
  <div class="col-4 low_container_player2_container">...
  </div>
</div>
</div>
```

Nous pouvons voir que le conteneur global contient la classe Bootstrap « container ».

Chaque ligne contient la classe Bootstrap « row », et chaque colonne contient la classe Bootstrap « col-4 ». La grille Bootstrap divisant en 12 la largeur globale de l'écran, « col-4 » nous indique que j'ai divisé les espaces des 3 éléments en 3 parties égales, la gestion du responsive est facilitée par cette propriété.

Je vais maintenant détailler chaque colonne de la grille Bootstrap :

J'ai créé une première ligne (classe « row », et « top-container » pour le CSS) de grille :

```
<div class="row top_container">
```

J'ai créé une première colonne sous la forme d'une balise <div> ayant la classe « player1 » renferme les infos de label « player1 », le point qui désigne quel joueur a la main dans le jeu, ainsi que son score global. Le code HTML résultant est celui-ci :

```
<div class="col-4 player1">
  <div class="player1_subinfos">
    <p class="player_id">PLAYER 1</p>
```

```

</div>
<p class="player1-global_score" id="global-p1">0</p>
</div>
```

Pour que la représentation colle au mieux avec la maquette, et pour me faciliter l'intégration dans le CSS, j'ai rassemblé le label « player1 » ainsi que le point dans une même div « player1\_subinfos ».

J'ai créé ensuite la deuxième colonne sous forme d'une balise <div> « dice » referme la représentation du dé :

```
<div class="col-4 dice">
    
</div>
```

J'ai déclaré une balise <img> en renseignant la source via l'attribut « src » ainsi que l'attribut « alt » contient une description textuelle de l'image utile pour l'accessibilité.

J'ai créé une troisième colonne sous forme d'une balise <div> ayant la classe « player2 ». Je ne la détaille pas car elle a exactement la même structure que la colonne « player1 » :

```
<div class="col-4 player2">
    <div class="player2_subinfos">
        <p class="player_id">PLAYER 2</p>
        
    </div>
    <p class="player2-global_score" id="global-p2">0</p>
</div>
```

J'ai ensuite créé une seconde ligne (classe « row », et « low-container » pour le CSS) de grille :

```
<div class="row low_container">
```

J'ai créé une première colonne sous la forme d'une balise <div> ayant la classe « low\_container\_player1\_container ». Cette balise regroupe le label « current » ainsi que le score courant :

```
<div class="col-4 low_container_player1_container">
    <p class="low_container_player1">CURRENT</p>
    <p class="low_container_player1_current-score" id="current-
p1">0</p>
</div>
```

J'ai ensuite créé une colonne regroupant les boutons d'action :

```
<div class="col-4 actions_container">
    <div class="roll_container">
        
        <p class="roll-par">ROLL DICE</p>
    </div>
    <div class="hold_container">
        
    </div>
</div>
```

```
<p class="hold-par">HOLD</p>
</div>
</div>
```

J'ai découpé cette colonne en deux balises <div> regroupant chacune une image et le label (balise <p>) de l'action.

A noter j'ai ajouté à chaque élément html une classe de manière à décrire au mieux son utilité, et permettant aussi la stylisation dans le CSS.

Pour finir avec la grille Bootstrap j'ai ajouté une troisième colonne sous la forme d'une balise <div> ayant la classe « low\_container\_player2\_container ». Comme dans la première ligne j'ai utilisé exactement le même mécanisme pour créer le score courant du joueur 2 :

```
<div class="col-4 low_container_player2_container">
  <p class="low_container_player2">CURRENT</p>
  <p class="low_container_player2_current-score" id="current-
p2">0</p>
</div>
```

J'ai ajouté un paragraphe devant apparaître lorsqu'un joueur a gagné :

```
<div class="win_container">
  <p class="win_p" id="win">You are the winner</p>
</div>
```

Enfin j'ai intégré le script javascript à la page HTML :

```
<script src="/assets/js/index.js"></script>
```

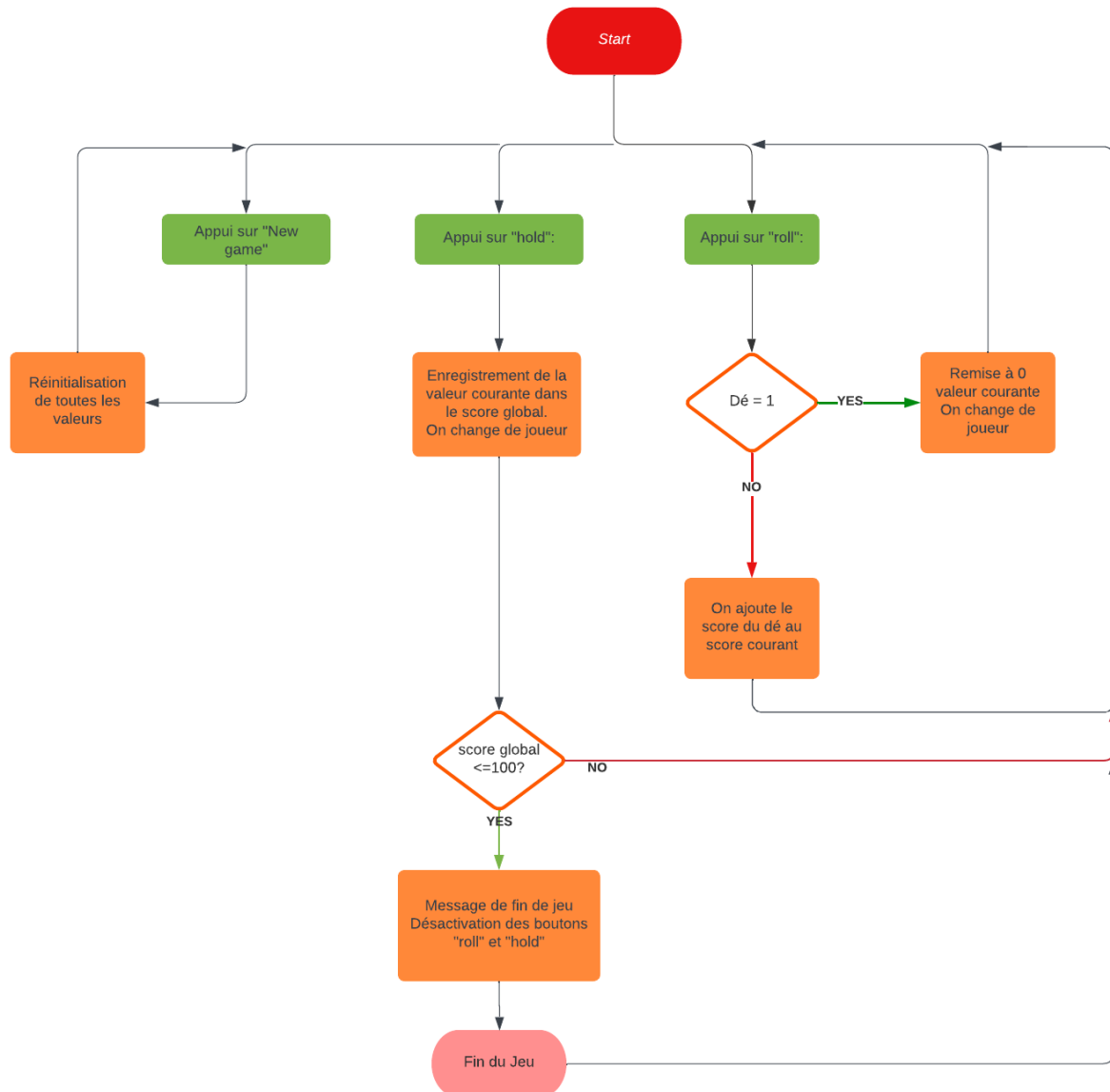
Je vais maintenant m'attarder à décrire le fonctionnement du programme javascript « index.js » permettant la mise en place du jeu de dés.

2- Le code Javascript :

Après analyse du besoin et des règles du jeu de dés, j'ai résumé le fonctionnement de l'application que je dois faire dans le diagramme fonctionnel ci-dessous :

### Dice

faure remi | September 21, 2022



En vert sont représentées les actions utilisateur. En orange les fonctions qui ont été mises en place dans le programme javascript. Les flèches représentent l'enchaînement des différentes fonctions entre elles.

#### 2.1- Gestion du « roll » :

J'ai tout d'abord codé la fonction « roll » de lancé de dés.

J'ai créé une fonction répondant aux critères :

- Détection du clic sur l'image/bouton ↻
- Générer un chiffre entre 1 et 6 (tirage du dé)
- Afficher la face du dé sur la page web
- Gérer les conditions : si la valeur du tirage dé est égale à 1, On remet à 0 la valeur courante du joueur et on change de joueur. Si la valeur du tirage dé est différente de 1 ; on ajoute la valeur du tirage dé au score courant du joueur qui a la main.

Pour détecter le clic sur l'image/bouton ↵, j'ai tout d'abord déclaré l'élément « roll ».  
Je me suis servi du DOM. Le « Document Object Model » qui est une interface de programmation pour les documents HTML.

```
var roll = document.getElementById('roll');
```

A noter que pour la manipulation du DOM par ce programme javascript, j'ai ajouté l'attribut « id » à chaque élément de la page html qui doit être dynamique.

A partir de cette variable « roll » j'ai pu donc manipuler l'image et créer une fonction à la détection du clic :

```
roll.addEventListener("click", () => {
```

J'ai créé une fonction spécifique qui génère le chiffre entre 1 et 6.

```
function generateRandomInt() {
    return Math.floor(Math.random() * 6 + 1);
}
```

Puis je l'ai appelé dans la fonction qui gère le clic, et j'affecte la valeur générée à la variable « generatedInt ».

```
generatedInt = generateRandomInt();
```

J'ai ensuite affiché l'image du dé correspondant à la valeur générée. J'ai au préalable utilisé le DOM pour manipuler l'élément « dice » :

```
var dice = document.getElementById('dice');
```

J'ai déclaré les différentes images des faces d'un dé dans un tableau :

```
let tab = ["/assets/img/1.JPG", "/assets/img/2.JPG", "/assets/img/3.JPG",
"/assets/img/4.JPG", "/assets/img/5.JPG", "/assets/img/6.JPG"];
```

Enfin la source de l'élément « dice » prend l'élément du tableau à l'indice de la valeur générée moins 1 :

```
dice.src = tab[generatedInt - 1];
```

Ensuite j'exécute un enchaînement de fonctions suivant si le joueur 1 ou 2 a la main :

```
if (player === "p1") { ...
}
else if (player === "p2") { ...
}
```

Je décris ici le fonctionnement pour le joueur 1 :

Si le tirage dé est égal à 1 on remet à 0 la valeur du score courant du joueur qui a la main :

```
if (generatedInt === 1) {
    player1current = 0;
```

Puis on remet à 0 l'affichage sur la page web :

```
currentP1.innerHTML = "0";
```

A noter que l'élément « currentP1 » récupère l'élément du DOM qui a comme Id « currentP1 » :

```
var currentP1 = document.getElementById('current-p1');
```

Puis il faut changer visuellement le joueur de main, c'est-à-dire que le point côté joueur 1 disparaît et que le point côté joueur 2 apparaît :

```
pointP1.style.display = "none";
```



```
pointP2.style.display = "block";
```

Les points des joueurs 1 et 2 ont été récupérés via le DOM précédemment dans le code :

```
var pointP1 = document.getElementById('point-p1');
var pointP2 = document.getElementById('point-p2');
```

Il faut enfin passer la main au joueur 2 :

```
player = "p2";
```

La variable « player » a été déclarée précédemment et a été définie par défaut à « p1 » :

```
var player = "p1";
```

Maintenant on gère le cas où le tirage dé est égal différent de 1.

On ajoute la valeur générée au score courant du joueur qui a la main. Puis on affecte cette valeur à l'élément HTML qui affiche cette valeur courante :

```
player1current = player1current + generatedInt;
currentP1.innerHTML = player1current;
```

A noter que la variable « player1current » a été déclarée précédemment dans le code :

```
let player1current = 0;
```

Cette fonction donne :

```
if (generatedInt === 1) {
    player1current = 0;
    currentP1.innerHTML = "0";
    pointP1.style.display = "none";
    pointP2.style.display = "block";
    player = "p2";
} else {
    player1current = player1current + generatedInt;
    currentP1.innerHTML = player1current;
}
```

La gestion complète du clic sur « roll » donne :

```
roll.addEventListener("click", () => {

    generatedInt = generateRandomInt();
    dice.src = tab[generatedInt - 1];

    if (player === "p1") {

        if (generatedInt === 1) {
            player1current = 0;
            currentP1.innerHTML = "0";
            pointP1.style.display = "none";
            pointP2.style.display = "block";
            player = "p2";
        } else {
            player1current = player1current + generatedInt;
            currentP1.innerHTML = player1current;
        }
    }
});
```

```

    }

    }
    else if(player === "p2") {


        if (generatedInt === 1) {
            player2current = 0;
            currentP2.innerHTML = "0";
            pointP1.style.display = "block";
            pointP2.style.display = "none";
            player = "p1";
        } else {
            player2current = player2current + generatedInt;
            currentP2.innerHTML = player2current;
        }
    }
}
})

```

## 2.2- Gestion du « hold » :

Le bouton « hold » a pour but :

- D'ajouter le score courant du joueur au score global
- Remettre à 0 le score courant
- Changer la main
- Si le joueur a un score global supérieur à 100, gérer le cas de victoire

Tout d'abord nous devons capter le clic sur le bouton . Comme pour le bouton de « roll » j'ai récupéré l'élément via le DOM et le placer dans une variable :

```
var hold = document.getElementById('hold');
```

J'ai ensuite codé la fonction qui capte le clic sur cette image :

```
hold.addEventListener("click", () => {
```

A l'intérieur de cette fonction j'ai implémenté les différentes étapes listées précédemment :

Tout d'abord, comme dans le cas du « roll » j'ai ajouté des conditions si la main est au joueur 1 ou 2 :

```

    hold.addEventListener("click", () => {
>     if (player === "p1") { ...
>     }
>     else if(player === "p2") { ...
>     }
    })

```

Les cas joueur 1 et 2 étant quasiment identiques, j'expliquerai le cas où le joueur 1 a la main.

J'ai ajouté le score courant du joueur au score global :

```
player1global = player1global + player1current;
```

Puis j'ai modifié l'élément HTML du score global :

```
globalP1.innerHTML = player1global;
```

J'ai déclaré « globalP1 » précédemment dans le code :

```
var globalP1 = document.getElementById('global-p1');
```

J'ai ensuite remis à 0 le score courant du joueur qui a la main :

```
player1current = 0;
currentP1.innerHTML = "0";
```

Puis j'ai changé la main vers le joueur suivant :

```
pointP1.style.display = "none";
pointP2.style.display = "block";
...
player = "p2";
```

Enfin je contrôle le score global : s'il atteint au moins 100 Il y a victoire pour le joueur qui a la main :

```
if (player1global >= 100) {
    victory();
}
```

J'appelle ici la fonction « victory() ».

La fonction entière donne :

```
hold.addEventListener("click", () => {
    if (player === "p1") {
        player1global = player1global + player1current;
        globalP1.innerHTML = player1global;
        player1current = 0;
        currentP1.innerHTML = "0";
        pointP1.style.display = "none";
        pointP2.style.display = "block";
        if (player1global >= 100) {
            victory();
        }
        player = "p2";
    }
    else if(player === "p2") {
        player2global = player2global + player2current;
        globalP2.innerHTML = player2global;
        player2current = 0;
        currentP2.innerHTML = "0";
        pointP1.style.display = "block";
        pointP2.style.display = "none";
        if (player2global >= 100) {
            victory();
        }
        player = "p1";
    }
})
```

Je détaille maintenant la fonction « victory() ».

Je fais apparaître tout d'abord le paragraphe qui annonce la victoire du joueur qui a la main :

```
win.style.display = "block";
```

J'ai déclaré précédemment la variable « win » qui fait référence à l'objet du DOM ayant l'id « win » :

```
var win = document.getElementById('win');
```

Ensuite, selon le joueur qui a la main (condition if), je modifie la phrase de victoire correspondant :

```
if (player === "p1") {
    win.innerHTML = Player 1 wins! Press NEW GAME to restart;
} else if (player === "p2") {
    win.innerHTML = Player 2 wins! Press NEW GAME to restart';
}
```

Ensuite je désactive les boutons de « roll » et de « hold » pour éviter les interactions du jeu alors que la partie est terminée :

```
roll.style.display = "none";
hold.style.display = "none";
```

Puis enfin je joue un son (ce qui n'était pas demandé dans le cahier des charges)

```
music.play();
```

J'ai déclaré la bande son précédemment dans le code :


```
const music = new Audio('/assets/sounds/win.wav');
```

La fonction « victory() » donne :

```
function victory() {
    win.style.display = "block";
    if (player === "p1") {
        win.innerHTML = 'Player 1 wins! Press NEW GAME to restart';
    } else if (player === "p2") {
        win.innerHTML = 'Player 2 wins! Press NEW GAME to restart';
    }
    roll.style.display = "none";
    hold.style.display = "none";
    music.play();
}
```

## 2.3- Gestion du « new game » :

Le clic sur ce bouton « new game » a pour but de faire une remise à 0 des états de toutes les variables, et ainsi pouvoir recommencer une partie comme il se doit, y compris après victoire de l'un des joueurs.

Tout d'abord nous devons capter le clic sur le bouton . Comme pour les autres boutons j'ai récupéré l'élément via le DOM et le placer dans une variable :

```
var new_game = document.getElementById('reset');
```

Puis j'ai codé la fonction chargée de surveiller le clic. A chaque clic la fonction « newGame() » va être exécutée :

```
new_game.addEventListener("click", () => {
    newGame();
})
```

Je décris la fonction « newGame() » :

```
function newGame() {
```

Tout d'abord j'ai codé la remise à 0 des scores courants et globaux du jeu :

```
player1current = 0;
player2current = 0;
player1global = 0;
player2global = 0;
```

Ainsi que leurs contenus HTML :

```
currentP1.innerHTML = "0";
globalP1.innerHTML = "0";
currentP2.innerHTML = "0";
globalP2.innerHTML = "0";
```

Je redonne la main au joueur 1, en désactivant le point du joueur 2 et en activant celui du joueur1 :

```
pointP1.style.display = "block";
pointP2.style.display = "none";
player = "p1";
```

Je désactive la phrase de victoire :

```
win.style.display = "none";
```

Et enfin je réactive les boutons de « roll » et de « hold » :

```
roll.style.display = "block";
hold.style.display = "block";
```

La fonction « newGame() » donne :

```
function newGame() {
    player1current = 0;
    player2current = 0;
    player1global = 0;
    player2global = 0;
    currentP1.innerHTML = "0";
    globalP1.innerHTML = "0";
    currentP2.innerHTML = "0";
    globalP2.innerHTML = "0";
    pointP1.style.display = "block";
    pointP2.style.display = "none";
    player = "p1";
    win.style.display = "none";
    roll.style.display = "block";
    hold.style.display = "block";
}
```

Le programme javascript résultant est celui-ci :

```

/* Variables declaration */
let player1current = 0;
let player2current = 0;
let player1global = 0;
let player2global = 0;
let tab = ["/assets/img/1.JPG", "/assets/img/2.JPG", "/assets/img/3.JPG",
"/assets/img/4.JPG", "/assets/img/5.JPG", "/assets/img/6.JPG"];
var player = "p1";

/* Variables of DOM objects declaration */
var new_game = document.getElementById('reset');
var roll = document.getElementById('roll');
var hold = document.getElementById('hold');
var win = document.getElementById('win');
var pointP1 = document.getElementById('point-p1');
var pointP2 = document.getElementById('point-p2');
var globalP1 = document.getElementById('global-p1');
var globalP2 = document.getElementById('global-p2');
var currentP1 = document.getElementById('current-p1');
var currentP2 = document.getElementById('current-p2');
var dice = document.getElementById('dice');

const music = new Audio('/assets/sounds/win.wav');

/* functions used in the script */
function generateRandomInt() {
    return Math.floor(Math.random() * 6 + 1);
}

function newGame() {
    player1current = 0;
    player2current = 0;
    player1global = 0;
    player2global = 0;
    currentP1.innerHTML = "0";
    globalP1.innerHTML = "0";
    currentP2.innerHTML = "0";
    globalP2.innerHTML = "0";
    pointP1.style.display = "block";
    pointP2.style.display = "none";
    player = "p1";
    win.style.display = "none";
    roll.style.display = "block";
    hold.style.display = "block";
}

function victory() {

```

```

win.style.display = "block";
if (player === "p1") {
    win.innerHTML = 'Player 1 win! Press reset to start a new game';
} else if (player === "p2") {
    win.innerHTML = 'Player 2 win! Press reset to start a new game';
}
roll.style.display = "none";
hold.style.display = "none";
music.play();
}

/* click functions */

new_game.addEventListener("click", () => {
    newGame();
})

roll.addEventListener("click", () => {

    generatedInt = generateRandomInt();
    dice.src = tab[generatedInt - 1];

    if (player === "p1") {

        if (generatedInt === 1) {
            player1current = 0;
            currentP1.innerHTML = "0";
            pointP1.style.display = "none";
            pointP2.style.display = "block";
            player = "p2";
        } else {
            player1current = player1current + generatedInt;
            currentP1.innerHTML = player1current;
        }

    }

    else if (player === "p2") {

        if (generatedInt === 1) {
            player2current = 0;
            currentP2.innerHTML = "0";
            pointP1.style.display = "block";
            pointP2.style.display = "none";
            player = "p1";
        } else {
            player2current = player2current + generatedInt;
            currentP2.innerHTML = player2current;
        }

    }
}

```

```

    }
  })

hold.addEventListener("click", () => {
  if (player === "p1") {
    player1global = player1global + player1current;
    globalP1.innerHTML = player1global;
    player1current = 0;
    currentP1.innerHTML = "0";
    pointP1.style.display = "none";
    pointP2.style.display = "block";
    if (player1global >= 100) {
      victory();
    }
    player = "p2";
  }
  else if (player === "p2") {
    player2global = player2global + player2current;
    globalP2.innerHTML = player2global;
    player2current = 0;
    currentP2.innerHTML = "0";
    pointP1.style.display = "block";
    pointP2.style.display = "none";
    if (player2global >= 100) {
      victory();
    }
    player = "p1";
  }
})

```

### 3- Le code CSS :

J'ai codé le CSS de manière à ce que le site soit « mobile first » et « responsive ». C'est-à-dire que l'affichage par défaut est celui pour les smartphones, et qu'il s'adapte à toutes les autres tailles d'écrans.

Les propriétés CSS suivent cet ordre :

=>Propriétés générales + les propriétés pour les écrans avec une largeur inférieure à 500px (smartphones)

```
@media (min-width: 500px) {
```

=>Propriétés pour les écrans avec une largeur supérieure à 500px et inférieure à 1024px (tablettes)

```
@media (min-width: 1024px) {
```

=>Propriétés pour les écrans avec une largeur supérieure à 1024px (ordinateurs de bureau)

Dans la partie des propriétés générales :



J'ai fait un « reset » du CSS et j'ai déclaré la police d'écriture tel qu'indiqué dans la documentation de Google Fonts (référence précédemment citée) :

```
* {
  margin: 0;
  padding: 0;
  font-family: 'Open Sans', sans-serif;
}
```

J'ai stylisé le titre principal de manière à ce qu'il soit centré :

```
h1 {
  text-align: center;
  margin-top: 20px;
}
```

J'ai stylisé le paragraphe en sous-titre du <h1> de manière à ce qu'il soit centré et avec une taille de police de 30px :

```
.main-p {
  text-align: center;
  margin-top: 10px;
  font-size: 30px;
}
```

J'ai stylisé le conteneur du bouton de nouvelle partie de manière à ce que l'image soit à côté du texte, et qu'il soit centré :

```
.new_game_container {
  display: flex;
  justify-content: center;
  margin-top: 50px;
}
```

J'ai dimensionné l'image ainsi que le label de nouvelle partie :

```
.new_game_img {
  width: 30px;
  height: 30px;
  cursor: pointer;
}

.new_game_par {
  margin-top: 2px;
  margin-left: 5px;
  font-size: 20px;
}
```

J'ai décalé vers le bas le conteneur « top\_container » comprenant les scores globaux et le dé :

```
.top_container {
  margin-top: 20px;
}
```

J'ai stylisé le conteneur du nom des joueurs de manière à ce que le point de main soit à côté du nom du joueur, et que ce qu'il y a à l'intérieur de ce conteneur soit centré :

```
.player1_subinfos, .player2_subinfos {
  display: flex;
  justify-content: center;
}
```

J'ai dimensionné les noms des joueurs ainsi que les points de main. J'ai caché le point du joueur2 car par défaut c'est le joueur 1 qui commence :

```
.player_id {
  font-size: 15px;
}

.player1_point {
  width: 25px;
  height: 25px;
}

.player2_point {
  width: 25px;
  height: 25px;
  display: none;
}
```

J'ai stylisé les scores globaux des deux joueurs :

```
.player1-global_score, .player2-global_score {
  font-size: 40px;
  text-align: center;
  color: rgb(163, 105, 105);
}
```

J'ai stylisé le conteneur « dice » ainsi que l'image du dé qu'il contient :

```
.dice {
  display: flex;
  justify-content: center;
}

.dice-img {
  margin-top: 10px;
  width: 80px;
  height: 80px;
}
```

J'ai stylisé le conteneur « low\_container » comprenant les scores courants ainsi que les boutons de « roll » et « hold » de manière à ce qu'ils soient sur la même ligne et espacés de manière égale :

```
.low_container{
  display: flex;
  justify-content: space-around;
  margin-bottom: 20px;
}
```

```
margin-top: 20px;
}
```

J'ai stylisé les conteneurs des boutons « roll » et « hold » de manière à ce que les images et les textes soient sur la même ligne :

```
.actions_container {
  width: 140px;
  height: 70px;
}

.roll_container {
  display: flex;
  justify-content: center;
}

.hold_container {
  display: flex;
  justify-content: center;
}
```

J'ai dimensionné les images et les noms des actions « roll » et « hold » :

```
.roll-img, .hold-img {
  width: 30px;
  height: 30px;
  cursor: pointer;
}

.roll-par, .hold-par {
  margin-top: 4px;
  margin-left: 5px;
  font-size: 15px;
  text-align: center;
}
```

J'ai dimensionné les conteneurs des scores courants et stylisé les labels et scores courants :

```
.low_container_player1_container, .low_container_player2_container {
  background-color: brown;
  width: 100px;
  height: 70px;
  padding-top: 10px;
  padding-left: 10px;
  padding-right: 10px;
  padding-bottom: 10px;
  border-radius: 2px;
}

.low_container_player1, .low_container_player2 {
  font-size: 10px;
}
```

```

text-align: center;
margin-bottom: 10px;
}

.low_container_player1_current-score, .low_container_player2_current-score {
text-align: center;
font-size: 20px;
color: white;
}

```

Enfin j'ai stylisé et centré la phrase qui apparait en cas de victoire :

```

.win_p {
font-size: 30px;
text-align: center;
display: none;
}

```

Comme vu dans précédemment, j'ai utilisé une média query pour adapter le style de certains items pour les écrans avec une largeur supérieure à 500px et inférieure à 1024px (tablettes) :

```

@media (min-width: 500px) {

```

J'ai redimensionné l'image et le label du bouton de « new\_game » :

```

.new_game_img {
width: 50px;
height: 50px;
}

.new_game_par {
margin-top: 4px;
margin-left: 5px;
font-size: 30px;
}

```

J'ai redimensionné l'image du dé :

```

.dice-img {
width: 100px;
height: 100px;
}

```

J'ai redimensionné l'image du point pour identifier qui a la main sur le jeu :

```

.player1_point, .player2_point {
margin-top: 10px;
}

```

J'ai modifié le style des labels des joueurs ainsi que du score global :

```

.player_id {
margin-top: 1px;
font-size: 30px;
}

```

```
}

.player1-global_score, .player2-global_score {
    font-size: 50px;
}
```

J'ai redimensionné le conteneur des boutons d'actions ainsi que le conteneur des scores courants :

```
.low_container_player1_container, .low_container_player2_container {
    width: 170px;
    height: 120px;
}

.actions_container {
    width: 170px;
    height: 120px;
}

.roll_container {
    margin-top: 20px;
}
```

J'ai redimensionné les images ainsi que les labels des boutons d'action :

```
.roll-img, .hold-img {
    width: 30px;
    height: 30px;
}

.roll-par, .hold-par {
    margin-top: 2px;
    margin-left: 5px;
    font-size: 20px;
    text-align: center;
}

.low_container_player1_current-score, .low_container_player2_current-score {
    font-size: 35px;
}
```

J'ai redimensionné la phrase qui apparait en cas de victoire :

```
.win_p {
    font-size: 40px;
}
```

J'ai utilisé une média query pour adapter le style de certains items pour les écrans avec une largeur supérieure à 1024px (ordinateurs de bureau) :

```
@media (min-width: 1024px) {
```

J'ai redimensionné l'image du dé :

```
.dice-img {
  width: 200px;
  height: 200px;
}
```

J'ai redimensionné les labels des joueurs ainsi que le score global :

```
.player_id {
  margin-top: 1px;
  font-size: 30px;
}
.player1-global_score, .player2-global_score {
  font-size: 70px;
}
```

J'ai redimensionné le conteneur des boutons d'actions ainsi que le conteneur des scores courants :

```
.low_container {
  margin-top: 100px;
}

.low_container_player1_container, .low_container_player2_container {
  width: 180px;
  height: 150px;
}

.actions_container {
  width: 250px;
  height: 150px;
}

.roll_container, .hold_container {
  margin-top: 20px;
}
```

J'ai redimensionné les images ainsi que les labels des boutons d'action :

```
.roll-img, .hold-img {
  width: 50px;
  height: 50px;
}

.roll-par, .hold-par {
  margin-top: 2px;
  margin-left: 15px;
  font-size: 30px;
  text-align: center;
}
```

J'ai redimensionné les labels ainsi que les scores courants :

```
.low_container_player1, .low_container_player2 {
    font-size: 15px;
}

.low_container_player1_current-score, .low_container_player2_current-score {
    margin-top: 40px;
    font-size: 45px;
}
```

Pour finir, j'ai redimensionné la phrase qui apparait en cas de victoire :

```
.win_p {
    font-size: 60px;
}
```

2. Précisez les moyens utilisés. Expliquez tout ce dont vous avez eu besoin pour réaliser vos tâches : langages de programmation, frameworks, outils, logiciels, documentations techniques, etc...

Afin de réaliser ce site, j'ai utilisé le logiciel Visual Studio Code ainsi que les modules comme Emmet et Prettier pour me faciliter l'écriture du code.

J'ai utilisé les langages de programmation HTML et CSS pour coder les pages web ainsi que les feuilles de style.

J'ai utilisé le langage de programmation Javascript pour rendre le site dynamique, rendre possible des interactions avec des éléments du site et ainsi mettre en place un jeu.

J'ai utilisé la librairie Bootstrap et la grille pour la mise en page du site web et l'agencement des éléments les uns par rapport aux autres.

J'ai utilisé le logiciel de gestion de versions GIT pour gérer les versions du site web et le mettre sur un dépôt distant de Github.

J'ai utilisé Powerpoint pour le schéma de la partie HTML du site.

J'ai utilisé le logiciel en ligne Ludichart pour créer le schéma fonctionnel du jeu (Javascript).

J'ai utilisé le service Netlify pour la publication du site.

3. Contexte. Les noms des organismes, entreprises ou associations, dans lesquels vous avez exercé vos pratiques

NB: Pour le cas des exercices et évaluations demandées sur la plateforme Studi, il s'agit de...Studi.

Ce site a été créé dans le cadre de l'évaluation demandée sur la plateforme STUDI.

4. Informations complémentaires (*facultatif*)