Manel MOKHTARI https://devmanel.github.io/Dice-Game.io/

ÉVALUATION D’ENTRAÎNEMENT : Dynamiser vos sites web avec Javascript

GDWFSDVSWEBAJAVAEXAIII1A

Studi

Création d’un petit jeu sur navigateur web à l’aide du DOM

Studi
DIGITAL EDUCATION

Sommaire

Table des matières

I. Introduction.

1. Compétence référentiel couvert par l’évaluation

Développer la partie front-end d’un jeu sur navigateur web à l’aide du DOM.

– Réaliser une interface utilisateur web statique et adaptable.

– Développer une interface utilisateur web dynamique.

Le jeu doit être un site web statique et adaptable, d’où le choix de développer avec en premier lieux les langages HTML, qui permet de représenter le contenu d’une page web et le langage CSS qui aide à concevoir une mise en page. Malgré cela, il est nécessaire de proposer un jeu fonctionnel avec des interactions . J’ai donc fait en sorte que le site soit dynamique d’où l’utilisation du langage JavaScript à l’aide du DOM qui permet de faire le lien entre un document web (HTML, CSS) et les scripts (JS) et de manipuler et modifier le contenu du document.

2. Résumé du projet.

Dans le cadre de ma formation Développeur web full stack, j’ai réalisé un jeu sur navigateur web à l’aide du DOM.

Ce jeu est un site statique dynamisé, l’utilisateur en naviguant sur le site, accède à une page web statique qui présente un plateau de jeu avec 2 joueurs sur un seul et même écran, pour chaque joueur un score temporaire et un score global sont affichés, à chaque tour le joueur a son score temporaire initialisé à 0 et peut lancer un dé autant de fois qu’il le souhaite, le résultat du lancé est ajouté au score temporaire, le plateau présent également 3 boutons; «New Game » qui permet de relancer une nouvelle partie, «ROLL DICE» permet de lancer le dé enfin«HOLD» qui permet d’envoyer les points du score temporaire vers le score global.

Lors d’un tour, le joueur peut donc décider à tout moment d’envoyer les points de son score temporaire vers son score global avec le bouton «HOLD», mettre fin à son tour et permettre par ce biais à l’autre joueur de jouer.

Un lancé de dé avec 1 comme résultat, fera perdre le nombre de point gagner dans le score temporaire et mettra fin au tour du joueur. Le premier joueur à atteindre les 100 points dans le score global gagne la partie.

J’ai réalisé ce site en totale autonomie, cela m’a permis de mettre en pratique les leçons théoriques de la formation, mais également d’en gagner de nouvelles tout en s’aidant et en me référant aux différentes documentations.

II. Le projet.

1. Spécifications techniques

Environnement de travail.

Je travaille sur un ordinateur portable car j’ai besoin d’être mobile. En effet, je me rends souvent aux bibliothèques de la villes afin d’avoir un espace de travaille calme.

2. Les technologies et les langages utilisés.

a. Partie Front-end.

– Le langage **HTML** qui utilise un système de balises pour définir les éléments du site et permettre ainsi de partager du texte avec les visiteurs du site et une navigation sans problème sur le site pour les utilisateurs.

– Le langage **CSS** qui utilise des styles permettant de créer des pages web avec une apparence soignée.

– Le framework front-end **IONIC**, un outil qui permet de créer un code multisupport en utilisant des outils web comme HTML, CSS, JavaScript, afin de de générer des applications IOS, Android, Chrome et bien d'autres

– Le **JavaScript**, est un langage de script léger, orienté objet, il permet de concevoir des sites web interactifs.

b. Autres Outils et logiciels.

– L'outil de versionnage **Git**, qui permet de développer les fonctionnalités sur une nouvelle branche, et le site **Github** utilisé pour déposer mon code après chaque commit, ainsi que le déploiement de mon site autant qu'hébergeur.

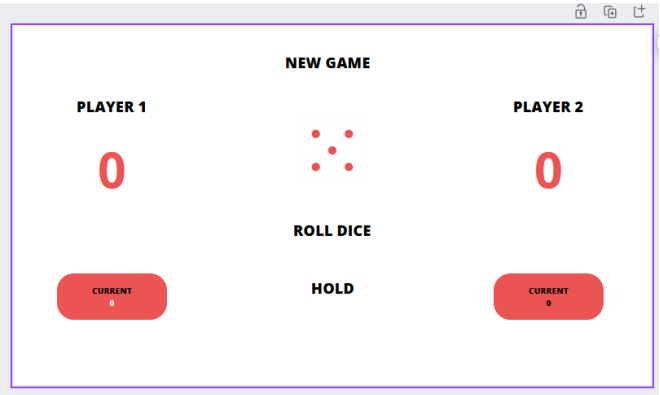
– **Visual Studio Code**, éditeur de code permettant d'écrire mes codes dans les différents langages utilisés.

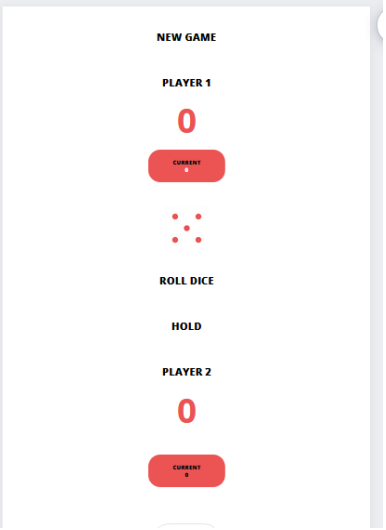
– Le **jQuery**, une bibliothèque JavaScript qui permet d'ajouter diverses fonctionnalités au site web et faciliter ainsi l'écriture de scripts côté client.

III. La réalisation du projet.

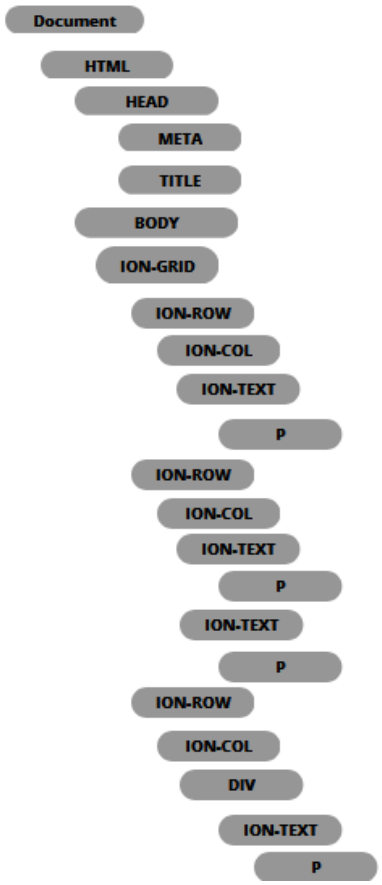
1. conception de la maquette.

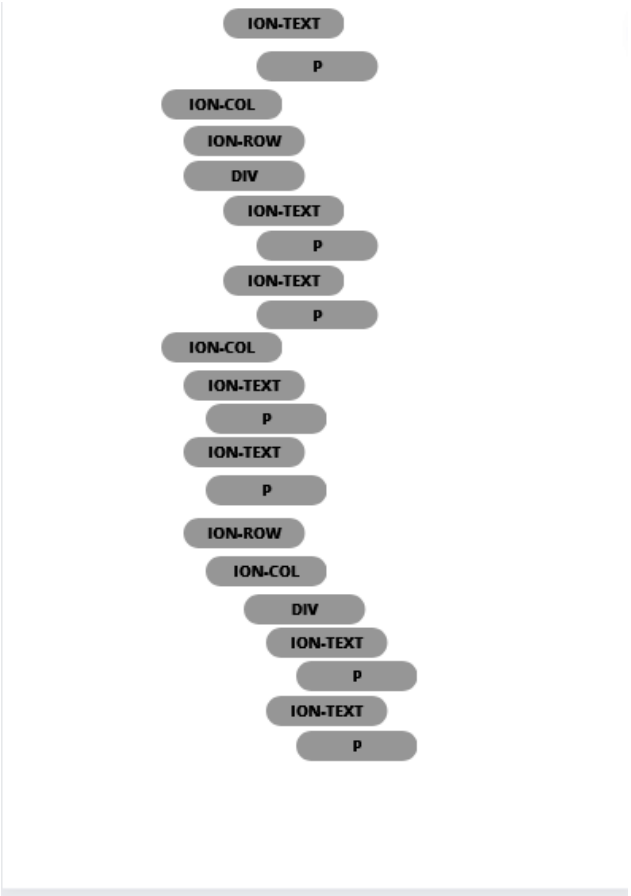
Tout d'abord, j'ai commencé par créer un design sur le site de design canva et de dessiner un pré visuel, en se posant la question de comment faire le layout pour afficher le plateau comme recommandé dans l'énoncé, et de l'adapter ensuite à des petits écrans.



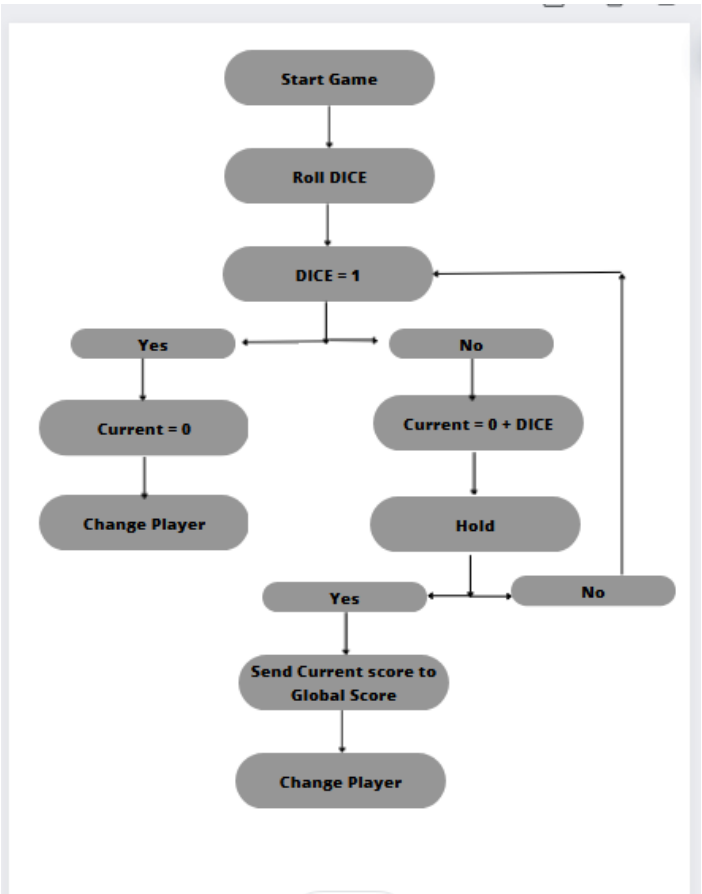


Ensuite, j'ai commencé par établir le schéma qui pourrait représenter la page web représentée par le code HTML.





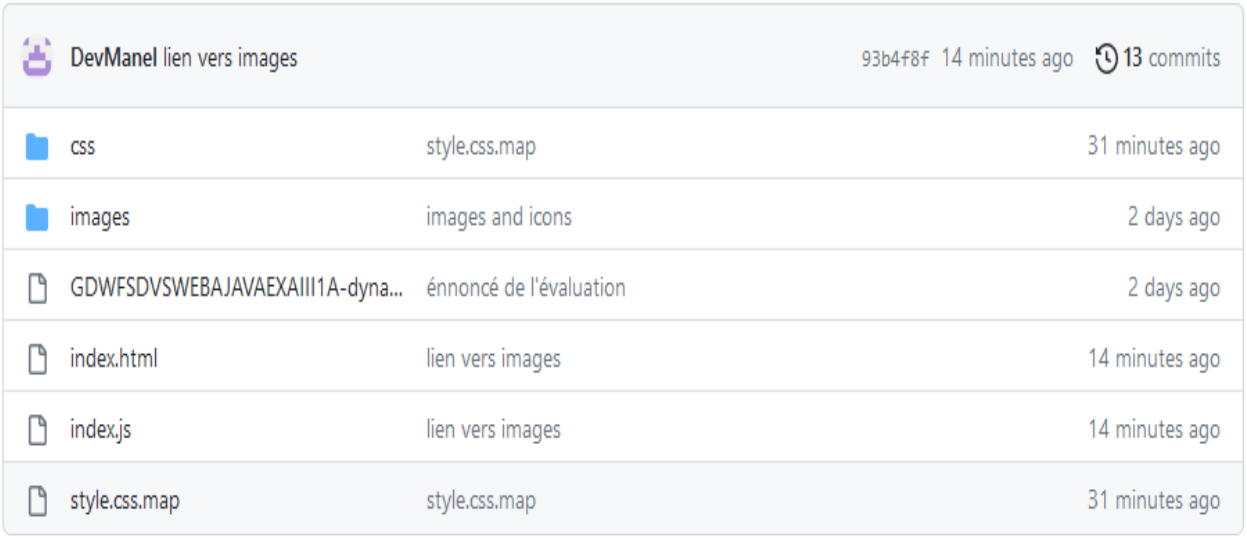
La question suivante, concernait la dynamisation du code et du jeu afin qu'il soit interactif et la mise en place des boucles et des fonctions ainsi que les itérations à implémenter. En effectuant les recherches sur les bibliothèques en lignes, le choix était tombé sur l'utilisation de fonctions et de conditions, qui faciliteront les différentes actions.
L'utilisation d'algorithme a aidé à visualisé la mise en place de l'ensemble du code.



3. Développement de la partie front-end.

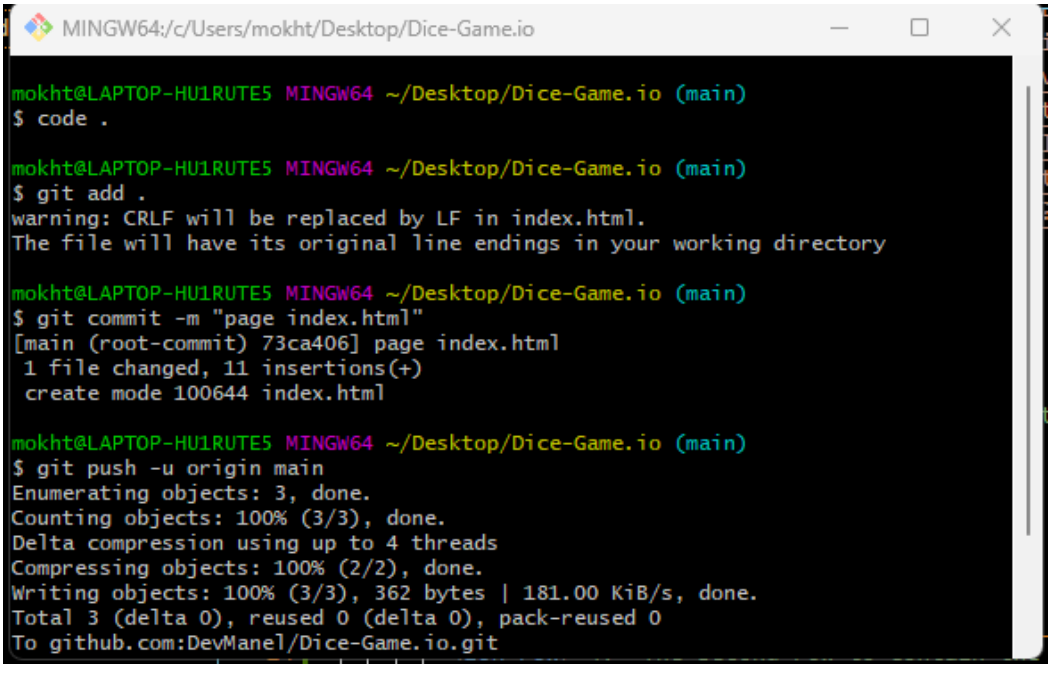
a. Mise en place de l'environnement de développement.

Git étant déjà installé sur mon ordinateur, j'ai donc commencer directement par initialisé le projet par le biai de ***git bash*** et la commande (**mkdir ‘’nom-du-dossier”**), ainsi que la création du fichier ***index,html***, qui va contenir les éléments importants statiques de la page web. J'ai également crée un dossier css, qui contiendra le fichier ***style.css*** pour appliquer un style et une mise en page, le fichier ***index.js*** afin de mettre en place les fonctions et le code nécessaire pour dynamiser le site, un dossier images contenant les images utilisées dans le site.



b. Initialisation du projet.

Visual Studio Code étant déjà installé sur mon ordinateur, j'ai commencer à coder grâce à la commande (***code .***) de git. J'ai initialisé le fichiers HTML (Index.html), le fichier CSS (style.css) et le fichier JavaScript (index.js), j'ai initialisé aussi le fichier



(style.css.map), les fichiers css.map permettent ensemble avec les navigateurs les plus modernes de réaliser des tâches ou d'inspecter les styles des éléments du DOM.
il sert de carte mémoire pour les fichiers compressés et le fichier style.scss permet de générer dynamiquement des feuilles de style et il apporte de nouvelles fonctionnalités non disponibles dans CSS.. Une fois l'initialisation du projet faite, avec la commande (git add .) et le commit, j'ai pu commencé le développement.

c. La page HTML.

J'ai donc choisi de développer à l'aide du ***framework IONIC*** qui permet comme cité plus haut, de créer un code multisupport en utilisant des outils web comme HTML, CSS, JavaScript pour mettre en page différents composants, tel que, la navigation, les boutons, les blocs et le système de grille, ainsi que la possibilité de rendre facilement un site responsive design et donc d'adapter le site à tous les écrans existant : ordinateur, mobile, tablette.

Les Metas

J'ai commencé par coder les différents <meta> qui me serviront pour le site :

- `<meta charset="UTF-8">`

Détermine la façon dont le texte est transmis et stocké et afin que le navigateur utilise le jeu de caractère universel Unicode et ainsi pouvoir désigner chaque caractère sans ambiguïté.

- `<meta http-equiv="X-UA-Compatible" content="IE=edge">`

Permet aux auteurs de sites web de choisir la version d'internet Explorer dans laquelle la page doit être rendue.

- `<meta name="viewport" content="width=device-width, initial-scale=1.0">`

Permet d'indiquer comment le navigateur doit afficher la page sur différents appareils.

Les link

En premier j'ai mis les codes CDN de ionic avant le fichier styles.css afin de bénéficier des styles et d'un graphisme standardisé pour tous les éléments habituels de ionic avant d'appliquer les styles personnalisés dans le fichier Css crée:

Link et Script ***Ionic*** **:**

-
```
<script type="module" src="https://cdn.jsdelivr.net/npm/@ionic/core/dist/ionic/ionic.esm.js"></script>
<script nomodule src="https://cdn.jsdelivr.net/npm/@ionic/core/dist/ionic/ionic.js"></script>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@ionic/core/css/ionic.bundle.css" />
```

Inclure le Ionic frameword CDN, afin d'utiliser tous les composants déjà installés dans ce framework.
J'ai utilisé jsdelivr afin d'accéder à la dernière version du framework.

-
```
<link rel="preconnect" href="https://fonts.gstatic.com">
```

Aide à implémenter web fonts dans notre projet, ce sont des styles déjà présent localement sur google.

-
```
<link href="https://fonts.googleapis.com/css2?family=Lato:wght@100;300;400&display=swap" rel="stylesheet">
```

Utiliser l'API google Fonts qui fournit la version standard des polices par défauts.

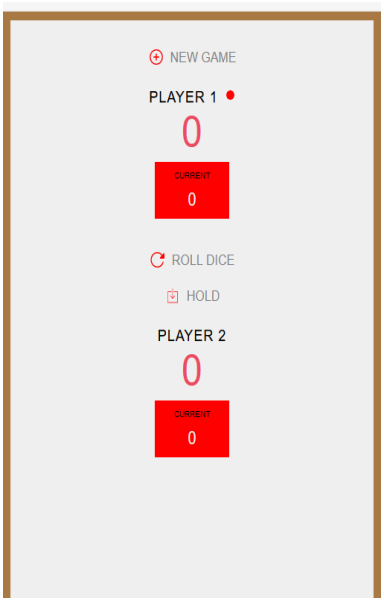
Link fichier **css** :

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

Utiliser pour relier le fichier CSS au fichier Html.

d. La navigation au sein du jeu et Le Système Grid.

La navigation au sein du site mobile est différente de la navigation sur le site web ordinateur : pour les écrans mobile appelés ‘’screen’’ l’affichage a été fait de façon à ce que les éléments s’empilent les uns sur les autres.



J’ai utilisé le ‘’Grid system’’, un puissant flexbox grid pour construire un layouts permettant de gérer la mise en page et la mise en forme de l’affichage du plateau de jeu grâce à son système de 12 colonnes et sa responsivité par défault. J’ai donc utilisé dans un premier lieux le système de grid avec le framework ionic ***<ion-grid>*** qui est un puissant système de mobile first flexbox pour construire des layouts, basé comme Bootstrap sur 12 colonne avec différents points de ruptures selon la taille de l'écran utilisé. La balise <ion-grid> contient deux rangés horizontales avec <ion-row> dont chaqu’une contient différentes colonnes qui aide à la configuration du plateau de jeu.
Dans mon fichier Index.html, j’ai donc créé cette arborescence :

```
<body>
  <ion-grid>
    <ion-row>
      <ion-col>
        <ion-text class="ion-text-center ion-text-uppercase">
          <p id="new-game" class="button"><span class="icons icon-replay"></span>new game</p>
        </ion-text>
      </ion-col>
    </ion-row>
    <ion-row>
      <ion-col size-xs="12" size-sm="5">
        <ion-text class="ion-text-center ion-text-uppercase">
          <p class="players name-0">Player 1<span class="player--0 active-player"></span></p>
        </ion-text>
        <ion-text class="ion-text-center" color="danger">
          <p id="score-0" class="score">0</p>
        </ion-text>
        <ion-row>
          <ion-col>
            <div class="current">
              <ion-text class="ion-text-center ion-text-uppercase">
                <p class="current-label">current</p>
              </ion-text>
              <ion-text class="ion-text-center" color="light">
                <p id="current-0" class="current-score">0</p>
              </ion-text>
            </div>
          </ion-col>
        </ion-row>
      </ion-col>
      <ion-col size-sm="2">
        <ion-row id="dice" class="ion-justify-content-center">
        </ion-row>
        <div class="action">
```

```
          </ion-row>
          <div class="action">
            <ion-text class="ion-text-center ion-text-uppercase">
              <p id="roll" class="button"><span class="icons icon-play"></span>roll dice</p>
            </ion-text>
            <ion-text class="ion-text-center ion-text-uppercase">
              <p id="hold" class="button"><span class="icons icon-hold"></span>hold</p>
            </ion-text>
          </div>
        </ion-col>
        <ion-col size-xs="12" size-sm="5">
          <ion-text class="ion-text-center ion-text-uppercase">
            <p class="players name-1">Player 2<span class="player--1"></span></p>
          </ion-text>
          <ion-text class="ion-text-center" color="danger">
            <p id="score-1" class="score">0</p>
          </ion-text>
          <ion-row>
            <ion-col>
              <div class="current">
                <ion-text class="ion-text-center ion-text-uppercase">
                  <p class="current-label">current</p>
                </ion-text>
                <ion-text class="ion-text-center" color="light">
                  <p  id="current-1" class="current-score">0</p>
                </ion-text>
              </div>
            </ion-col>
          </ion-row>
        </ion-col>
      </ion-row>
    </ion-grid>
    <script src="js/index.js"></script>
  </body>
```

Le ***framework Ionic*** possède également différentes classes et propriétés qui lui sont propres. Ainsi, les classes ionic déclarées tout au long du code à l'intérieur des balises <ion-…> jouent un rôle important dans la conception du plateau de jeu.

– `class="ion-text-center ion-text-uppercase"`

Dans l'élément <ion-text>, j'ai utilisé la classe ion-text-center et la classe ion-text-uppercase qui permet de mettre le texte en majuscule et de le centrer.

– `size-xs="12" size-sm="5"`

ce sont des propriétés basées sur un layout de 12 colonnes avec différents points de ruptures, ici size indique donc le nombre de colonnes utilisé sur les 12 par rangée, pour les écrans de largeur 0 minimum pour xs, et de 576px minimum pour sm.

J'ai donc procédé à la mise en place dans un premier temps du bouton <new game> grâce à la classe ''button'', ensuite dans un second temps la mise en place de la partie droit du plateau ''Player 1'' avec ses scores temporaires et globales, du centre avec la mise en place du dé et des boutons ''Roll dice'' et ''Hold'', et de la partie gauche ''Player 2'' avec également ses scores temporaires et globales.

c. Les idetifiants et Les classes.

Avec le DOM, il est possible d''utilisation des éléments du code HTML afin de les lier aux fichiers css et js.
L'application d'identifiants doit être unique à un seul éléments HTML contrairement aux classes qui elles peuvent être utilisées par plusieurs éléments.
J'ai donc utilisé ici des identifiants et des classes en attribuant des styles et des actions spécifiques.

```
<p id="new-game" class="button"><span class="icons icon-replay"></span>new game</p>
```

```
/*Button*/
.button {
    font-weight: 400;
    color: gray;
    cursor: pointer;
  }
```

```
// Get new game button
const newGame = document.querySelector("#new-game");
```

d. JavaScript.

Le langage JavaScript sert à rendre un site dynamique, il est souvent utilisé aussi avec le DOM pour être relier avec le fichier HTML et pouvoir ainsi manipuler ou modifier les éléments HTML.j'ai donc créé le fichier index.js.

Pour accéder aux éléments du fichier html on utilise le mot document. qui représente le contenu de la page web, et qui permet de parcourir les éléments, en ajoutant ou en supprimant.
J'ai commencé tout d'abord par déclarer la variables qui vont être utiles pour la suite du programme.

– let randomNumber = 0; est une variable que je vais utilisé comme générateur de numéro aléatoire pour le lancé du dé.
– let roundScore = 0; une variable pour le score temporaire lors des lancés de dé
– let activePlayer = 0; une variable pour le nombre de point cumulés lorsqu'un joueur est entrain de jouer.
let scores = [0, 0]; une variable pour le score global de chaque joueur en forme de tableau.

```
// variables
let randomNumber = 0; //choose a random number when rolling athe dice
let roundScore = 0;
let activePlayer = 0;
let scores = [0, 0];
```

Par la suite, j'ai récupérer des éléments du fichier HTML en utilisant le DOM et leurs ai appliqué des constances.
// Get new game button
const newGame = document.querySelector("#new-game");
const dice = document.querySelector("#dice");
// Get roll button
const roll = document.querySelector("#roll");
// Get hold button
const hold = document.querySelector("#hold");

Ce sont des méthodes d'interface selector qui permettent de sélectionner le 1er élément qui correspondent aux sélecteurs CSS ici les id de dice, new-game, roll et hold.

```
//Get buttons by using the querySelector to select the id in html
// Get new game button
const newGame = document.querySelector("#new-game");
// Get roll button
const roll = document.querySelector("#roll");
// Get hold button
const hold = document.querySelector("#hold");

// Get the dice element
const dice = document.querySelector("#dice");

  // Get the players
const player0 = document.querySelector(".player--0");
  const player1 = document.querySelector(".player--1");
```

L'utilisation de fonctions ensuite était nécessaire pour la dynamisation des éléments HTML. J'ai ainsi utilisé une première fonction pour avoir des nombres aléatoires à chaque lancé de dé et l'ai attribué à la variable randomNuber .
– randomNumber = Math.floor(Math.random() * 6) + 1;

Math.floor(), permet de prendre un nombre décimal et de le convertir en nombre entier, Math.random() quant à lui crée un nombre aléatoire entre 0 et 0,9 qui sera multiplié par 6 et ensuite arrondis par Math.floor()
+1 est ajouté afin de créer un nombre entre 1 et 6.
On a donc par exemple : (un nombre aléatoire entre 0 et 0,9)0,7 * 6 = 4,2. qui sera arrondi à 4 par Math.floor et auquel on rajoutera 1. le résultat final lacé par le dé sera 5.

J'ai fait appel à la propriété innerHTML qui permet d'afficher l'arborescence du DOM à partir de l'élément souhaité, ici à partir de l'élément #dice;
– dice.innerHTML.
Ici la propriété de l'élément récupère récupérer les images des différentes faces de dé dans le dossier images auquel on applique la variable randomNumber avec le calcul aléatoire permettant de lancer le dé.

La condition mise en place:

```
if (randomNumber !== 1) {
    roundScore += randomNumber;
    // Display round score
    document.querySelector(`#current-${activePlayer}`).textContent = roundScore;
  } else {
    changePlayer();
  }
};
```

Si le nombre aléatoire n'est pas égale à 1; le score temporaire ou round score est additionné avant un autre lancé aléatoire du dé dans current qui est affiché pour le jour qui a la main.
Sinon, on change de joueur avec la fonction switchPlayer(), qui permet de changer de joueur en réinitialisant son score temporaire à 0, on précise donc que le joueur commence

avec un current 0, si le joueur actif lance un dé et que celui-ci affiche 1 le score temporaire est remis à 0 et un changment de joueur sera effectué, à l'aide de la classListe.toggle. Elle retourne une collection direct DOMTokenList des attributs de classe de l'élément. C'est une alternative à la propriété element.className qui renvoie une chaîne composée de la liste des class séparées par des espaces.

```
// function to Roll the dice and display the round score
const rollDice = function () {
    // Create a random number
    randomNumber = Math.floor(Math.random() * 6) + 1;

    // Display dice
    dice.innerHTML = `<img class="dice" src="./images/dice/dice-${randomNumber}.png" alt="dice ${randomNumber}">`;

    // Round score
    if (randomNumber !== 1) {
      roundScore += randomNumber;
      // Display round score
      document.querySelector(`#current-${activePlayer}`).textContent = roundScore;
    } else {
      switchPlayer();
    }
  };
```

```
  // Switch player
const switchPlayer = function () {
    roundScore = 0;
    document.querySelector(`#current-${activePlayer}`).textContent = 0;
    activePlayer = activePlayer === 0 ? 1 : 0;
    player0.classList.toggle("active-player");
    player1.classList.toggle("active-player");
  };
```

player0.classList.toggle("active-player");
player1.classList.toggle ("active-player");

Ce code retourne donc une collection direct DOMTokenList des attributs de classe de l'élément. C'est une alternative à la propriété element.className qui permet de renvoyer une chaîne composée de la liste des classes séparées par des espaces. Elle permet donc ici d'effectuer le changement du point rouge à côté du mot ''Player''.

hold score.
*const holdScore = function () {
  // add current score
  scores[activePlayer] += roundScore;
  // display score
  document.querySelector(`#score-${activePlayer}`).textContent = scores[activePlayer];

  // check player score
  if (scores[activePlayer] >= 100) {
    document.querySelector(`.playerName-${activePlayer}`).classList.add("winner-player");
    document.querySelector(`.playerName-${activePlayer}`).innerHTML = ` winner
! `;

  } else {
    // Change player

```
    changePlayer();
  }
};
```

```
// Hold the score
const holdScore = function () {
    // add current score
    scores[activePlayer] += roundScore;
    // display score
   document.querySelector(`#score-${activePlayer}`).textContent = scores[activePlayer];

    // check player score
    if (scores[activePlayer] >= 100) {
      document.querySelector(`.name-${activePlayer}`).classList.add("winner-player");
      document.querySelector(`.name-${activePlayer}`).innerHTML = `<p>winner !</p>`;

    } else {
      // switch player
      switchPlayer();
    }
  };
```

J’ai crée également la fonction ci-dessus, qui va permettre dans un premier temps pour le joueur qui a la main d’additionner son score temporaire qui est dans le current après chaque lancé de dé.
Avec document.querySelector(`#score-${activePlayer}`).textContent = scores[activePlayer], textContent définit ou renvoie le contenu textuel du nœud spécifié, et de tous ses enfants, ici ça permet d’afficher le score obtenu et permet au joueur actif d'ajouter s'il le souhaite le nombre obtenu dans le current scroe au score global, en cliquat sur le bouton ‘’hold’’.
ensuite, j'ai mis une condition, si au moment d'appuyer sur hold pour additionner le current score au score globale, le score du joueur actif est supèrieur ou égal à 100, alors le nom du joueur sera remplacé par le mot Winner! qui sera en gras et en rouge.
sinon, on change de joueur, et le current score sera remis à zéro.

Cette fonction, sert à créer une nouvelle partie lorsqu'on clique sur le bouton ‘’New Game’’.

Enfin, j’ai mis en place une liste d’évènement lors du clique sur les boutons.

e. Les Media Queries.

```
@media (min-width: 576px) {
  .button {
    margin: 20px 0 20px 0;
  }
  .dice {
      margin-top: 50px;
  }
  .player {
      font-size: x-large;
      margin: 15px 0 0 0;
  }
  .current {
      margin-top: 120px;
  }
}
```

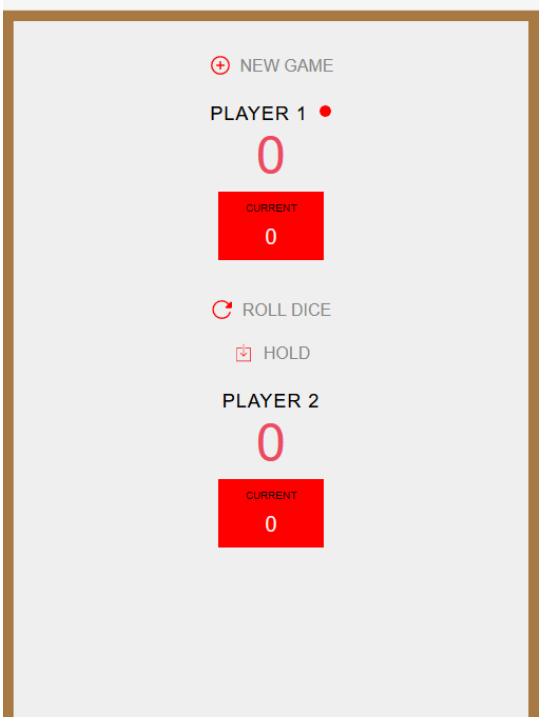
J’ai utilisé les media Queries dans le fichier css pour avoir la possibilité à l’utilisateur de jouer sur des petits écrans.
Pour les écrans mobile de taille pixel minimum, j’ai donc attribué des marges aux boutons, au dé, aux mots ‘’Player’’ et au ‘’Current’’ de sorte qu’ils restent aligner au centre avec des marges et des tailles de polices afin de les adapter pour des petits écrans.

IV. écrans développés.

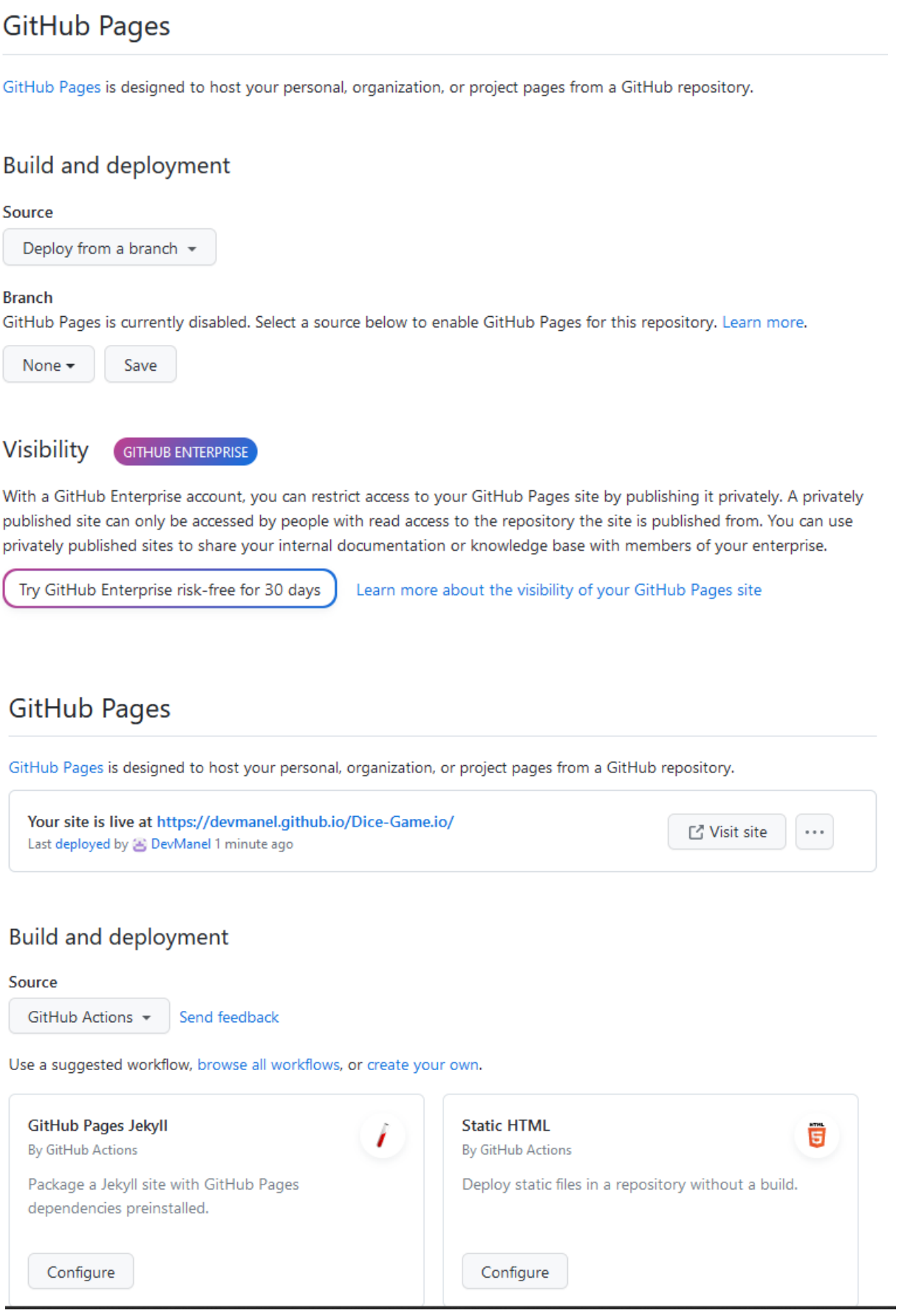
Voici le détail des écrans que j’ai pu développer.

Accueil

À son arrivée dans le site, l’utilisateur arrive sur un plateau de jeu, Avec des boutons au centre, le bouton ‘’New Game’’ qui relance une nouvelle partie, le bouton ‘’Roll Dice’’ qui fait apparaître un dé avec un nombre aléatoire à chaque clique, le bouton ‘’Hold’’ pour sauvegarder le nombre de point accumulé dans le current. À droite et à gauche de l’écran les joueurs avec en dessous le score globale et le score temporaire, pour les écrans larges.
Pour les petits écrans les éléments sont fait de sorte à s’afficher les uns sur les autres.



Enfin, j’ai déployé le site grâce à Github :



v. Conclusion.

Concernant le site, les fonctionnalités de base sont développées. C’est un plateau de jeu statique, mais qui présente tout de même une part de mon projet personnel.

En codant en autonomie, je savais qu’il serait un défi pour moi.j’ai abordé les langages comme, Html, Css, JavaScript ainsi que le framework Ionic, même si mes connaissances ne me permettaient pas à elles-seules de procéder au développement du site assez facilement.

En effet, le développement du site a été un défi tous les jours, car il m’a fallu apprendre plus à l’aide des cours de la formation, de la documentation officielle, de tutoriels vidéo et reproduire ce que je viens d’apprendre sur mon projet. Ce que je trouve être la meilleure méthode de progresser rapidement, et une bonne façon d’entrevoir ce qu’est le métier de développeur.