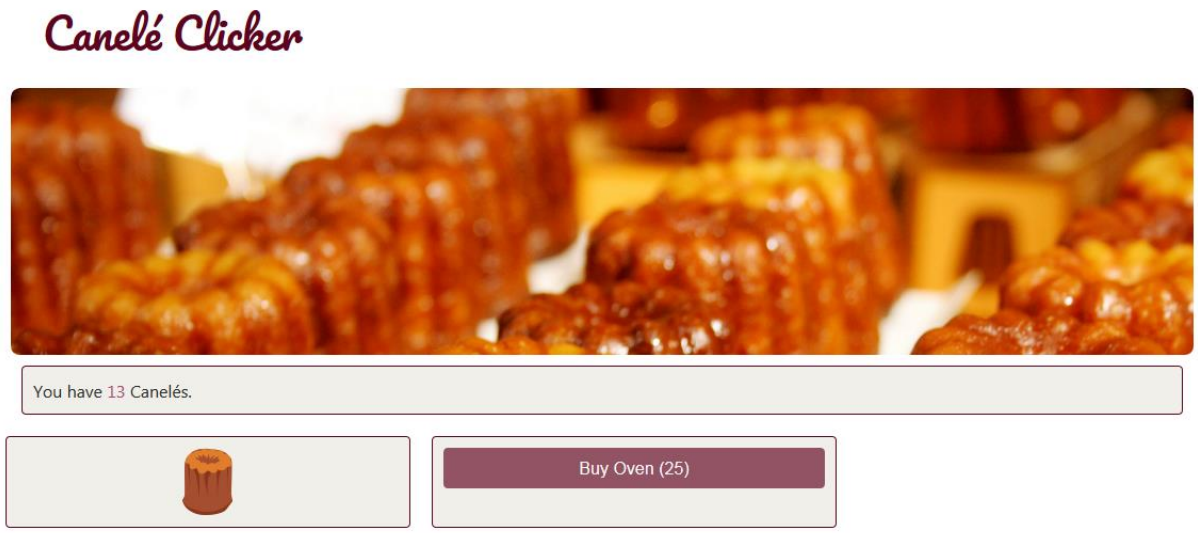
CaneléClicker

Vous allez apprendre à concevoir un petit jeu de navigateur où le but est simple, cliquer pour faire grimper le score.

A la clef, apprentissage des technologies du web soit le HTML, le CSS et le Javascript. Vous serez donc capable de pousser le jeu un peu plus loin ou vous lancer dans votre propre site web !



I. Matériel requis

Pour faire du web et suivre ce tuto, voici ce dont vous avez besoin:

- Un navigateur récent (Firefox ...)
- Un éditeur de fichier texte (Bloc note, Notepad++, Atom ...)
- Ces fichiers

En utilisant un éditeur de texte avancé, il vous sera plus simple de vous repérer dans le code (Coloration syntaxique, arborescence du projet, ...).

Télécharger Atom

2. Le HTML

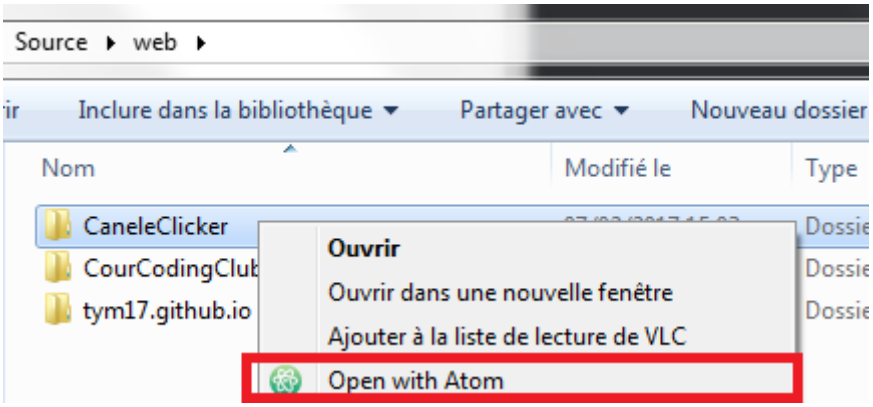
Maintenant que vous avez préparé votre navigateur et lancé votre éditeur de texte. Nous allons pouvoir créer le premier fichier de notre site.

- Créer un nouveau dossier pour mieux s'organiser.
- Créer un fichier "index.html".

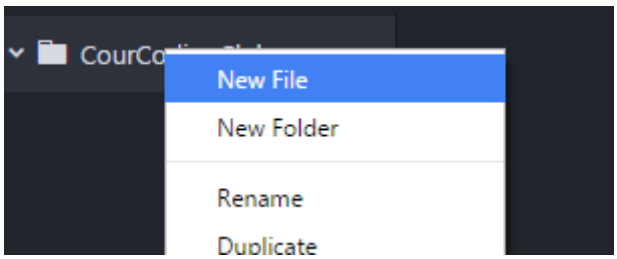
Vous noterez que ce fichier se termine par ".html", ce qui signifie qu'il contient du HTML. C'est le langage qui permet à votre navigateur de savoir quels éléments seront affichés sur la page.

Attention ! Si vous voyez un fichier texte (ex: index.html.txt), la création n'a pas été exécutée correctement, vous avez créé un fichier texte lambda.

Voici le déroulement de cette étape avec Atom:



- Faire un clic droit sur le dossier que vous avez créé.
- Ouvrir avec Atom.



- Faire un clic droit sur le dossier affiché dans l'arborescence du projet et créer un nouveau fichier.
- Entrer le nom du fichier donc "index.html".
- Votre fichier va donc s'ouvrir et, surprise, il est vide. A nous de le remplir !

2.I Le HTML

Ce fichier HTML est le minimum que tout navigateur demande pour commencer à afficher du contenu. Cependant, il n'y a pas seulement du contenu dans ce fichier.
Il y a aussi des informations sur le type d'alphabet utilisé, les indications concernants l'échelle à utiliser si l'utilisateur a un petit écran ou est sur son smartphone. Ainsi que les polices d'écritures et autres éléments que la page va utiliser.

Voici donc le corps minimal d'un fichier HTML:

```
<!DOCTYPE html>
<html>

<head>
</head>

<body>
</body>

</html>
```

Avant de remplir notre fichier, regardons un peu l'architecture de base d'une page. Chaque élément **"<NomDeL'Element> </NomDeL'Element>"** représente un élément délimité par une ou deux balises.

- Une balise est constituée de son nom entouré de chevrons *(ex: <body>)*
- Une balise comportant un **"/"** après son chevron d'ouverture est une balise de fin. Elle annonce la fin de l'élément *(ex: </body>)*.

- Certaines balises peuvent être à la fois ouvrantes et fermantes. Elles représentent à elles seul l'élément. On les remarquera par l'absence de balise de fin ou la présence du **"/"** avant le chevron de fermeture. *(ex: <!DOCTYPE html>)*
- Une balise peut aussi contenir des informations complémentaires sur elle même *(ex: <button class="rouge"> Bouton </button>).*

Nous avons donc 4 éléments: !DOCTYPE, html, head et body. Les plus important ici sont head et body.
L'élément head est l'élément que votre navigateur va utiliser pour préparer le chargement des éléments visuels ainsi que des préférences pour afficher la page.
L'élément body, lui, va contenir le contenu que le navigateur va afficher.

2.3 Head

Nous allons donc remplir les informations concernant notre entête de page web. Plaçons du code entre ces deux balises head pour informer notre navigateur.

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<title>CaneleClicker</title>
<link rel="shortcut icon" href="img/canele-vec.png">
```

Les deux premières lignes vont nous permettre d'indiquer le type d'alphabet à utiliser (UTF-8) et la mise à l'échelle pour les écrans. Les deux autres vont renseigner au navigateur comment nommer l'onglet et quelle icône lui attribuer.

Mais quelle icône lui attribuer ?
Nous allons lui donner une image toute faite qui se trouve dans le dossier que vous avez téléchargé au début.

- Créer un dossier **"img"** à côté de notre fichier index.html
- Copier/coller le contenu du dossier **"img"** de l'archive zip dans votre dossier **"img"**.

Revenons sur la balise <link>, la propriété "href" établi un lien vers un fichier que votre navigateur va charger depuis l'emplacement de votre fichier. Donc en chargeant "img/canele-vec.png", il chargera l'image **"canele-vec.png"** située dans le dossier **"img"** par rapport à l'emplacement de notre fichier html.

Maintenant, glissez-déposez votre fichier HTML dans votre navigateur moderne favoris et observer l'onglet prendre le nom et l'icône que nous lui avons donné.

Continuons à ajouter des fichiers à charger. Nous allons inclure un fichier qui va nous permettre d'utiliser une police d'écriture différente de celle présente par défaut ainsi que des feuilles de style. Ces feuilles de style sont un fichier dans le langage CSS. Ils permettent de décrire l'aspect graphique des éléments HTML (ex: Ce texte sera rouge).

```
<link rel="stylesheet" type="text/css" href="css/bootstrap.min.css">
<link rel="stylesheet" type="text/css" href="css/new.css">
<link rel="shortcut icon" href="img/canele-vec.png">
<link href="https://fonts.googleapis.com/css?family=Neucha|Pacifico" rel="stylesheet">
```

Nous allons créer le fichier “new.css” dans le chapitre dédié au CSS mais nous utiliserons le fichier “bootstrap.min.css” présent dans l’archive que vous avez téléchargé au début. Il faudra donc, comme vous l’avez fait pour l’image, la copier dans un dossier “css” à côté de votre fichier “index.html”.

Bien, il ne nous manque plus qu'à charger une chose dans notre entête. Le fichier de Javascript qui va nous permettre de programmer la logique de notre jeu. Car le HTML et le CSS permettent de décrire la page et son aspect. Le Javascript, lui permet de prendre des actions et d’interagir dans l’environnement créé par le HTML.

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

2.4 Body

Passons au concret, ajoutons des éléments à afficher. L’objectif de cette étape est d’ajouter des éléments qui s’imbriquent entre eux pour déterminer l’architecture de la page. Placez votre curseur à l’intérieur des balises body pour commencer à le modifier.

Commençons par créer une zone centrale découpable en 12 colonnes (Cette fonctionnalité vient de la feuille de style “bootstrap” que nous avons inclu auparavant):

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

</div>
</div>
```

Les balises “div” sont des balises invisibles si elles n’ont pas étée modifiées par le CSS. Nous ne voyons donc toujours rien. Pourtant, notre terrain est préparé.

Continuons en ajoutant une division qui va prendre les 12 colonnes dans laquelle nous aurons une image ainsi que le texte qui affichera le nombre de canelés que notre joueur possède.

```
<div class="col-12">
        <h1>Canelé Clicker</h1>
        <div class="image-cadre">
          <img src="img/Caneles_stemilion.jpg" />
        </div>
        <p class="elem">You have <span id="number">0</span> Canelés.</p>
</div>
```

Attention à bien insérer ce code à l’intérieur de notre colonne centrale! Donc au milieu des deux balises “div”.

Nous pouvons ici apercevoir différentes balises:

- **h1** permet de placer un titre.
- **img** permet de placer une image.
- **p** permet de placer un paragraphe.
- **span** permet d'entourer une portion de texte. (Semblable à div)

Depuis que nous avons commencé, la plupart des balises possèdent une propriété "class". Cette propriété sert à indiquer au navigateur d'aller se référencer à la feuille de style par rapport au contenu de cette propriété pour décorer l'élément. Par exemple, la classe "col-12" indique que cet élément "div" fera 12 colonnes de large. La classe "image-cadre" que nous allons créer, contiendra l'image.

Continuons et ajoutons les boutons qui vont permettre d'interagir avec le jeu.

```
<div class="col-4 elem">
  <img onclick="bake()" class="clicky" src="img/canele-vec.png" />
</div>
<div class="col-4 elem">
  <button class="btn btn-block btn-buy" id="ovenBtn" onclick="buyOven()"
disabled="true">Buy Oven (25)</button>
</div>
```

Vous remarquerez de nouvelles propriétés:

- **id** représente une adresse censée être unique pour votre élément.
- **onclick** représente la fonction Javascript à appeler lorsque votre utilisateur cliquera sur cet élément.

- **disabled** désactive ou active le bouton et permet ou non à votre utilisateur d'interagir avec.

Normalement, à partir de maintenant, vous devriez avoir une page web avec quelques éléments graphiques. Cependant, l'absence de style rend notre page peu agréable. Passons donc au chapitre CSS pour arranger tout ça !

Appelez notre équipe au secour si vous avez un problème !

3. le CSS

3.I Les éléments statiques

Maintenant que nous avons une page HTML, il faut la rendre jolie ! Nous allons donc y appliquer du CSS.

- Créez un fichier "new.css" dans le dossier "css"

Faites bien attention à créer un fichier ".css" et non ".css.txt" !

Le CSS est un autre langage que le HTML, même si il sert lui aussi à décrire quelque chose. Voici à quoi ressemble du CSS:

```
h1
{
   color: #5F021F;
   font-family: 'Pacifico', cursive;
   margin: 30px;
}
```

C'est d'ailleur le code que nous allons utiliser pour styliser notre balise **h1** ! Cette classe CSS permet d'informer notre navigateur sur 3 choses par rapport à nos balises **h1**:

- Le texte sera de la couleur #5F021F (Bordeaux)
- La police d'écriture utilisée sera "Pacifico"
- Il y aura une marge extérieure de 30 pixels.

Cette notation pour la couleur (#RRGGBB) représente un mélange de rouges, verts et bleus. Chaque couleur peut être présente à un indice de 255 soit FF en base hexadécimale. #FFFFFF correspond donc au blanc, #FF0000 au rouge et #0000FF au bleu.
Actualisez/Rechargez votre page pour voir si votre titre change d'apparence et ressemble plus au titre présent sur l'image de couverture.

Passons à l'image qui occupe une grande partie de votre écran. Nous allons d'abord styliser l'élément HTML possédant la classe "image-cadre" car il contient l'image. Il va donc pouvoir cacher le dépassement de l'image. Ajoutons donc ces deux classes CSS à la suite des informations pour notre balise **h1.**

```
.image-cadre
{
   width: 100%;
   height: 250px;
   overflow: hidden;
   border-radius: .5rem;
}
```

On apprend donc sur l'élément qui possède cette classe "image-cadre" que:

- Sa largeur occupe toute celle de l'élément supérieur à lui.
- Il fait 250 pixels de haut.
- Il cache le surplus de contenu.
- Ses bords sont arrondis de 0.5 rem (rem: unité qui varie avec la taille de l'écran).

La présence d'un point au début du nom de cette classe n'est pas un hasard. Il existe différents sélecteurs pour informer quels éléments styliser. Par exemple, lorsque l'on met le nom d'une balise sans point ni #, on sélectionne toutes les balises du même nom. Alors que si l'on met un point comme ici, on sélectionne tous les éléments qui ont une classe dont le nom est égal à celui noté après le point. Le # permet de sélectionner un élément à partir de son "id".

```
.image-cadre img
{
   display: block;
   width: 100%;
   height: auto;
   border-radius: .5rem;
}
```

Ici on apprends donc que lorsqu'une balise "img" sera contenue dans un élément portant la classe "image-cadre", cette balise sera affichée tel un block, sa hauteur sera ajustée automatiquement et elle aura une largeur de 100% de son élément supérieur ainsi qu'un coin arrondi de 0.5 rem.

Mettons maintenant en forme la page ainsi que les multiples éléments "div" qui possèdent la classe "elem" à l'aide de ces classes CSS que l'on ajoutera toujours à la suite.

```
body
{
 font-family: 'Neucha', cursive;
}

#number
{
 color: #A47;
}

img
{
 display: block;
```

```
  margin: auto;
}
```

On apprends donc que les éléments à l'intérieur de la balise body auront la police "Neucha", que le contenu de l'élément qui a pour "id": "number" sera d'un rouge un peu plus clair (#A04070) et que les balises img simple seront affichées en block et auront une marge extérieure automatique et seront donc centrée par rapport à leur élément supérieur.

```
.elem
{
  margin: 10px;
  border: 1px solid #5F021F;
  padding: 10px;
  border-radius: .25rem;
  background-color: #F1EFE9;
}
```

Pour la classe "elem", nous allons retrouver des attributs que nous avons déjà vu plusieurs fois mais aussi:

- **border** qui informe que l'élément va avoir une bordure entière d'une largeur d'1 pixel et de la couleur #5F021F.
- **padding** indique la marge intérieure, donc 10 pixels.
- **background-color** indique la couleur du fond du block.

3.2 Les éléments dynamiques

Jusqu'à présent, nous avons stylisé notre page mais le canelé cliquable n'est pas cliquable et le bouton n'est pas non plus très ergonomique.

Pour ajouter un peu d'animation, il existe plusieurs moyens, dont deux simples.

- Le premier consiste à utiliser une feuille de style déjà existante qui implémente déjà des comportements spécifiques. C'est le cas avec la feuille "bootstrap" qui régit le look de notre bouton "Oven" avec la classe "btn".
- Le deuxième consiste à utiliser les différentes possibilités que le CSS nous offre.

3.2.1 Construire sur quelque chose d'existant

Etant donné que tout a déjà été fait pour nous, il ne nous reste plus qu'à modifier ou ajouter ce qui nous manque.

```
.btn-buy
{
 background-color: #5F021F;
 color: #FFFFFF;
}

.btn-buy:hover:enabled
{
 background-color: #A12;
 cursor: pointer;
}
```

La classe ".btn-buy:hover:enabled" est la même classe que celle au dessus: "btn-buy". En ajoutant ":hover:enabled" on informe le navigateur qu'il y a un comportement différent à adopter lorsque le bouton est activé et que l'utilisateur passe sa souris dessus.

3.2.2 Partir de zéro

Je vous rassure, on ne part pas totalement de zéro. Cependant, il va falloir détailler un peu plus le comportement de notre élément.

```
.clicky
{
 transition-duration: 0.2s;
 cursor: pointer;
 opacity: 0.9;
}

.clicky:active
{
 transform: translateY(10px);
 opacity: 1;
}

.clicky:hover
{
 opacity: 1;
}
```

Nous retrouvons ici une seule et même classe avec trois comportements différents. Le comportement de base, le comportement lorsque la souris est dessus (hover) et celui lorsque l'utilisateur clique dessus (active).
On retrouve aussi d'autres attributs:

- **transition-duration** permet d'établir une transition linéaire entre les comportements de ces éléments.

- **cursor** permet de changer l'icône du curseur. Ici, la même que lorsque l'on survole un lien.
- **opacity** renseigne la transparence de l'élément entre 0 et 1.
- **transform** établi une action à réaliser. Ici, déplacer l'élément de 10 pixels vers le bas (axe Y).

Voila ! Vous devriez avoir à ce stade une page web qui ressemble comme deux gouttes d'eau à celle présente sur la page de présentation. Libre à vous de continuer à personnaliser la feuille de style. Vous pouvez trouver beaucoup plus d'attributs en recherchant sur internet comme notamment sur le site que Mozilla a mis à disposition des développeurs.
Faites attention cependant, tous les attributs ne sont pas tous disponibles sur tous les navigateurs. C'est pour cette raison que lorsque vous ouvrez votre site favoris avec un navigateur ancien, il peut avoir quelques défauts.

4. Javascript

Nous allons donc passer au Javascript. Ce langage est contenu dans des fichiers ".js" et il permet d'ajouter des comportements spécifiques à la page que le navigateur va charger. Dans notre cas, nous allons nous en servir pour implémenter la logique de notre jeu.

4.I Bouton principal

Revenons un peu à notre HTML, vous souvenez vous de l'image que nous avons placé avec un attribut "onclick" ?

```
<img onclick="bake()" class="clicky" src="img/canele-vec.png" />
```

Lorsque nous avons écrit cette balise, nous avons informé notre navigateur d'utiliser la fonction "bake" de notre Javascript. Une fonction est une action que le navigateur va lancer lorsqu'elle sera appelée. "bake" sera donc appelé si l'utilisateur clique sur cet élément.

- Créer un dossier "js" à côté de vos dossiers "img" et "css".
- Créer un fichier "script.js" dans ce dossier et ouvrez le.
- Ajouter ce code.

```
function bake()
{
   alert('bonjour');
}
```

En ajoutant le code ci dessus dans votre fichier Javascript et en rechargeant la page, vous devriez voir une fenêtre pop-up s'afficher lorsque vous cliquez sur le canelé.
Maintenant que vous savez à quoi sert une fonction, modifions notre fonction "bake" pour qu'elle affiche le score du joueur. Cependant, il va aussi falloir stocker ce score quelque part. Nous allons donc créer

une variable. C'est une valeur que l'on peut modifier et utiliser dans nos fonctions.

Ajoutez donc ceci au début de votre fichier:

```
var score = 0;
var adding = 1;
```

Nous avons créé deux variables, une "score" et une autre "adding". "Adding" va nous servir à savoir de combien augmenter le score à chaque clic. Modifions donc notre fonction "bake" en conséquence.

```
function bake()
{
    score += adding;
    document.getElementById('number').innerHTML = score;
}
```

Si vous testez votre code, vous verrez que le nombre contenu dans l'élément span avec l'id "number" augmentera avec les clics.

You have 35 Canelés.

4.2 Améliorations

Pour ajouter un peu de piquant à notre jeu, ajoutons un moyen d'obtenir plus de canelés en un seul clic. Nous allons donc donner vie à notre bouton "Buy Oven (25)" que nous avons ajouté tout à l'heure.

```
<button class="btn btn-block btn-buy" id="ovenBtn" onclick="buyOven()"
disabled="true">Buy Oven (25)</button>
```

Comme précédemment, nous avons un attribut "onclick". Cette fois-ci, il appelle une autre fonction: "buyOven".

Ajoutons donc cette fonction à la suite dans notre fichier javascript.

```
var ovenPrice = 25;

function buyOven()
{
   if (score >= ovenPrice)
   {
      score -= ovenPrice;
      document.getElementById('number').innerHTML = score;
      adding += 1;
   }
}
```

Nous avons ajouté une variable pour définir le prix de cette amélioration ainsi qu'une fonction qui, si la variable "score" est

supérieure à la variable “ovenPrice”, va soustraire la valeur de la variable “ovenPrice” et ensuite changer le contenu de l’élément avec l’id “number” et ajouter 1 à notre variable adding qui est utilisée dans notre fonction “bake”.
Si vous testez votre site, vous verrez que rien ne change et que l’on ne peut pas cliquer sur notre bouton pour améliorer notre production de canelé. Il faut activer le bouton.
Dans quelles situations, serons nous amenés à activer ou désactiver notre bouton ?

- Lorsque nous cliquons sur le bouton principal :
  - Activer le bouton si l’on dépasse le prix de l’amélioration.
- Lorsque nous achetons l’amélioration:
  - Désactiver le bouton si la quantité de canelé est insuffisante.

Comme cette action sera exécutée à la fois dans le code de notre fonction “bake” et celui de notre fonction “buyOven”, nous allons créer une fonction pour éviter d’écrire du code identique. Ajoutons donc à la suite dans notre fichier javascript.

```
function refreshOptions()
{
   if (score < ovenPrice)
      document.getElementById('ovenBtn').disabled = true;
   else
      document.getElementById('ovenBtn').disabled = false;
}
```

Ce code indique que, **si** le “score” est inférieur au prix “ovenPrice”, l’élément avec l’id “ovenBtn” sera désactivé. **Sinon**, il sera activé.

Maintenant il faut appeler cette fonction à la fin de chaque fonction que nous avons déjà fait. Donc ajouter la ligne

```
refreshOptions();
```

avant l'accolade de fin de nos fonctions. Ce qui donnerait ceci pour notre fonction “bake”:

```
function bake()
{
    score += adding;
    document.getElementById('number').innerHTML = score;
    refreshOptions();
}
```

5. Conclusion

Maintenant vous pouvez tester votre jeu et voir cette page prendre vie sous vos yeux ! Libre à vous d’ajouter des améliorations ou d’autres éléments à votre jeu/page, si vous avez besoin d’aide, l’équipe est là pour vous aider !

Voici un lien vers le corrigé:
https://github.com/Tym17/CaneleClicker