HTML, CSS, Javascript
Manipuler des images responsives
Document de travail

C.Turrier - 25 juin 2023

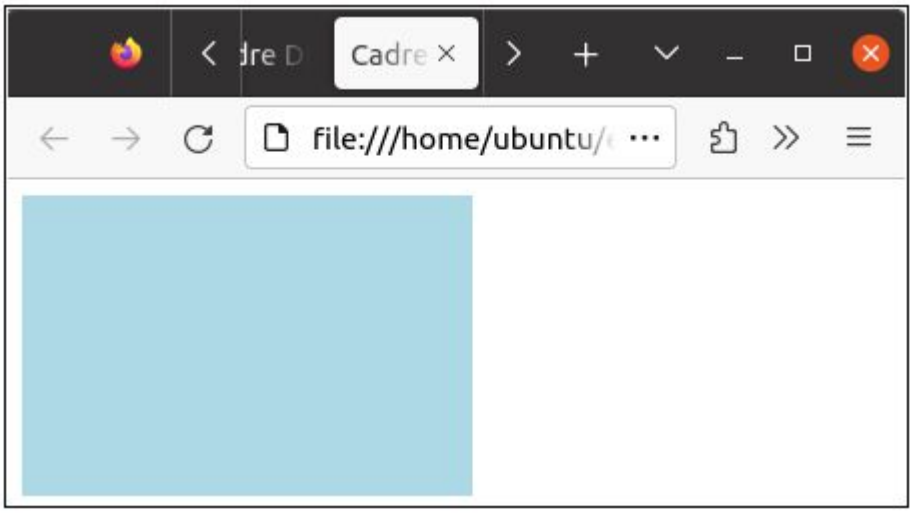
Le présent document de travail est à finaliser. Il propose des exemples de code pour la manipulation d'images ou de morceaux d'images :

- responsives,
- positionnables (dans un cadre et dans des sous-cadres)
- cliquables
- déplaçables

Ces manipulations peuvent constituer une première étape en vue de la création d'un jeu avec des images, en HTML, CSS et Javascript, comme un jeu de pousse pousse responsif par exemple.

1) Afficher un cadre

En HTML et javascript, quel est le code pour afficher, dans la fenêtre du navigateur, un cadre responsive div faisant 50 % de la largeur de l'écran et dont la hauteur fait 2/3 de la largeur ?




```
<!DOCTYPE html>
<html>
<head>
  <title>Cadre Div Responsive</title>
  <style>
    #myDiv {
      width: 50vw; /* 50% de la largeur de la fenêtre */
      height: 33.33vw; /* 2/3 de la largeur de la fenêtre */
      background-color: lightblue;
    }
  </style>
</head>
<body>
  <div id="myDiv"></div>

  <script>
    // Redimensionner le cadre Div lors du chargement de la page et
lors du redimensionnement de la fenêtre
    function resizeDiv() {
      var div = document.getElementById("myDiv");
      div.style.height = "calc(33.33vw)";
    }

    window.addEventListener("load", resizeDiv);
    window.addEventListener("resize", resizeDiv);
  </script>
</body>
</html>
```

Dans ce code, nous utilisons le CSS pour définir la largeur et la hauteur du cadre avec l'ID "myDiv". La propriété width est définie sur 50vw, qui représente 50% de la largeur de la fenêtre (vw signifie "viewport width"). La propriété height est définie sur calc(33.33vw), qui calcule la hauteur comme étant les 2/3 de la largeur de la fenêtre.

Ensuite, nous utilisons JavaScript pour ajuster dynamiquement la hauteur du cadre en fonction de la largeur de la fenêtre. La fonction resizeDiv() est appelée lors du chargement de la page (window.addEventListener("load", resizeDiv)) et lors du redimensionnement de la fenêtre (window.addEventListener("resize", resizeDiv)). Elle met à jour la hauteur du cadre en utilisant la même formule calc(33.33vw).

Ainsi, lorsque vous ouvrez cette page HTML dans un navigateur ou redimensionnez la fenêtre, le cadre s'ajustera automatiquement pour avoir une largeur égale à la largeur de la fenêtre et une hauteur égale aux 2/3 de la largeur. La couleur de fond du cadre est définie sur "lightblue" dans cet exemple, mais vous pouvez le modifier selon vos préférences.

L'unité de mesure vw en CSS signifie "viewport width" (largeur de l'affichage).

L'unité **vw** représente une fraction de la largeur de la fenêtre visible, généralement exprimée en pourcentage. Par exemple, 100vw signifie 100% de la largeur du viewport, 50vw signifie 50% de la largeur du viewport, et ainsi de suite.

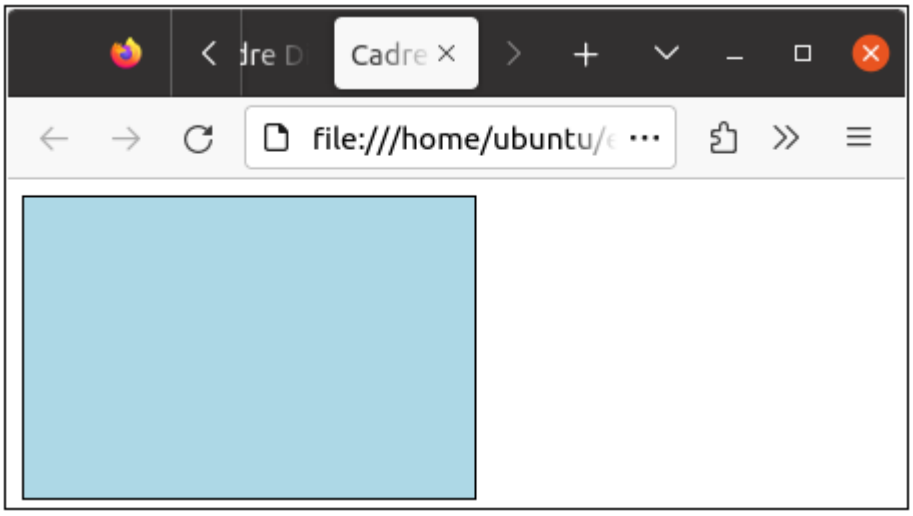
Le viewport fait référence à la partie visible de la fenêtre du navigateur dans laquelle le contenu web est affiché. Il exclut les barres de défilement verticales ou horizontales, les barres d'outils du navigateur ou tout autre élément qui pourrait prendre de l'espace dans la fenêtre.

L'utilisation de l'unité vw est utile pour créer des mises en page fluides et réactives (responsive), où les éléments peuvent être dimensionnés en fonction de la taille de l'écran ou de la fenêtre du navigateur. Par exemple, si vous utilisez width: 50vw; pour un élément, cet élément occupera toujours la moitié de la largeur de la fenêtre, quelle que soit la taille de l'écran.

L'unité vw est prise en charge par la plupart des navigateurs modernes. Il existe également d'autres unités similaires, telles que vh (viewport height, hauteur de l'affichage), qui fonctionne de la même manière mais pour la hauteur du viewport, et vmin (viewport minimum, minimum du viewport width ou height), qui correspond à la plus petite valeur entre la largeur et la hauteur du viewport.

1.1) Afficher une bordure au cadre

Comment faire pour que ce cadre responsive soit entourée d'une bordure noire ?



Pour ajouter une bordure noire autour du cadre responsive , vous pouvez utiliser la propriété CSS **border** pour définir les styles de bordure. Voici le code modifié avec la bordure noire :

```
<!DOCTYPE html>
<html>
<head>
  <title>Cadre Div Responsive</title>
  <style>
    #myDiv {
      width: 50vw; /* 50% de la largeur de la fenêtre */
      height: calc(33.33vw); /* 2/3 de la largeur de la fenêtre */
      background-color: lightblue;
      border: 1px solid black; /* Bordure noire de 1 pixel */
    }
  </style>
</head>
<body>
  <div id="myDiv"></div>

  <script>
    // Redimensionner le cadre Div lors du chargement de la page et
lors du redimensionnement de la fenêtre
    function resizeDiv() {
      var div = document.getElementById("myDiv");
      div.style.height = "calc(33.33vw)";
    }

    window.addEventListener("load", resizeDiv);
    window.addEventListener("resize", resizeDiv);
  </script>
  <br/>
```

```
</body>
</html>
```

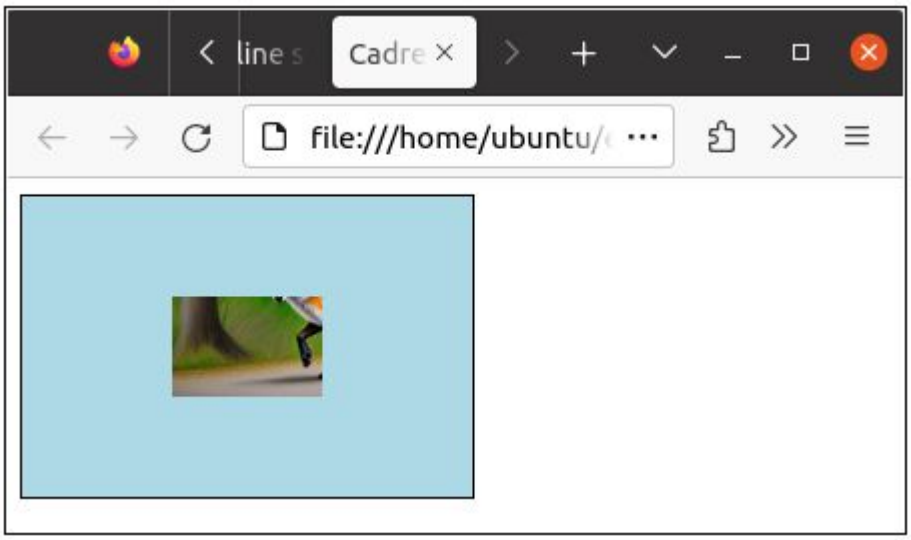
Dans ce code, nous avons ajouté la propriété **border** à la définition de style pour #myDiv. La valeur 1px solid black spécifie une bordure de 1 pixel de large, de type solide, et de couleur noire.

Lorsque vous ouvrez cette page HTML dans un navigateur, vous verrez que le cadre aura une bordure noire d'une largeur de 1 pixel autour de lui.

Vous pouvez personnaliser les styles de la bordure en modifiant les valeurs de la propriété border selon vos préférences. Par exemple, vous pouvez ajuster la largeur de la bordure en modifiant le nombre de pixels, ou changer la couleur en utilisant un autre code de couleur (par exemple, "2px dashed red" pour une bordure rouge pointillée de 2 pixels de large).

2) Afficher une image dans un cadre

Comment afficher, dans le cadre "mydiv", une image "1.jpg" qui occupe 1/3 du cadre en largeur et 1/3 du cadre en hauteur ?



Pour afficher une image dans le cadre myDiv en utilisant le code fourni, vous pouvez ajouter un élément **<img>** à l'intérieur de la **div** et définir ses dimensions en pourcentage par rapport à la taille du cadre. Voici comment vous pouvez le faire :

```
<!DOCTYPE html>
<html>
<head>
  <title>Cadre Div avec Image</title>
  <style>
    #myDiv {
      width: 50vw; /* 50% de la largeur de la fenêtre */
      height: 33.33vw; /* 33.33% de la largeur de la fenêtre */
      background-color: lightblue;
      border: 1px solid black; /* Bordure noire de 1 pixel */
      position: relative; /* Position relative pour positionner
l'image */
    }

    #myDiv img {
      position: absolute; /* Position absolue pour positionner l'image
*/
      width: 33.33%; /* 33.33% de la largeur du cadre */
      height: 33.33%; /* 33.33% de la hauteur du cadre */
      top: 33.33%; /* Position en haut du cadre */
      left: 33.33%; /* Position à gauche du cadre */
    }
  </style>
</head>
```

```
<body>
  <div id="myDiv">
    <img src="1.jpg" alt="Image" />
  </div>
</body>
</html>
```

Dans ce code, nous avons ajouté un élément <img> à l'intérieur de la div myDiv. En utilisant les sélecteurs CSS #myDiv img, nous définissons les styles spécifiques à l'image.

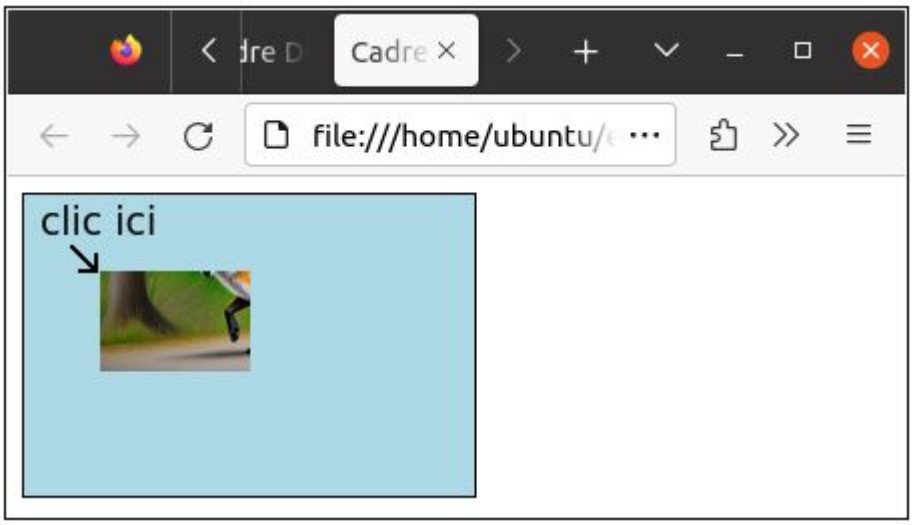
Les propriétés position: absolute sont utilisées pour positionner l'image à l'intérieur du cadre. Les propriétés width et height sont définies à 33.33% pour occuper 1/3 de la largeur et de la hauteur du cadre. Les propriétés top et left sont définies à 33.33% pour positionner l'image dans le coin supérieur gauche du cadre.

Assurez-vous de remplacer "1.jpg" par le chemin ou l'URL réel de votre image. Cela permettra d'afficher l'image dans le cadre myDiv.

3) déplacer une image dans un cadre, en cliquant

Quel code javascript faut il utiliser pour pouvoir déplacer le coin haut et gauche de l'image "1.jpg" dans le cadre myDiv, à l'endroit où l'on clique dans ce cadre myDiv ?

Pour déplacer le coin supérieur gauche de l'image "1.jpg" à l'endroit où l'on clique dans le cadre myDiv, vous pouvez ajouter un code JavaScript pour gérer l'événement de clic et mettre à jour les positions de l'image. Voici un exemple de code JavaScript que vous pouvez utiliser :




```
<!DOCTYPE html>
<html>
<head>
  <title>Cadre Div avec Image</title>
  <style>
    #myDiv {
      width: 50vw; /* 50% de la largeur de la fenêtre */
      height: 33.33vw; /* 33.33% de la largeur de la fenêtre */
      background-color: lightblue;
      border: 1px solid black; /* Bordure noire de 1 pixel */
      position: relative; /* Position relative pour positionner
l'image */
    }

    #myDiv img {
      position: absolute; /* Position absolue pour positionner l'image
*/
      width: 33.33%; /* 33.33% de la largeur du cadre */
      height: 33.33%; /* 33.33% de la hauteur du cadre */
      top: 33.33%; /* Position en haut du cadre */
      left: 33.33%; /* Position à gauche du cadre */
    }
  </style>
```

```
</head>
<body>
  <div id="myDiv">
    <img id="myImage" src="1.jpg" alt="Image" />
  </div>

  <script>
    var myDiv = document.getElementById("myDiv");
    var myImage = document.getElementById("myImage");

    myDiv.addEventListener("click", function(event) {
      var offsetX = event.offsetX;
      var offsetY = event.offsetY;

      myImage.style.left = offsetX + "px";
      myImage.style.top = offsetY + "px";
    });
  </script>
</body>
</html>
```

Dans ce code, nous avons ajouté une balise <script> qui contient le code JavaScript. Nous sélectionnons l'élément myDiv et myImage en utilisant getElementById. Ensuite, nous ajoutons un écouteur d'événement de clic sur myDiv.

Lorsque vous cliquez sur le cadre myDiv, la fonction de rappel de l'événement est déclenchée. Nous récupérons les coordonnées offsetX et offsetY de l'événement de clic, qui correspondent aux coordonnées relatives à myDiv.

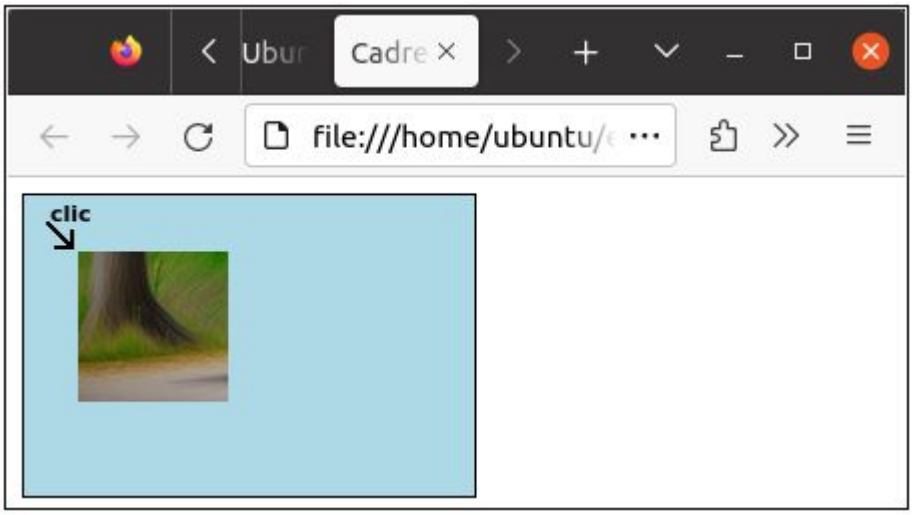
Ensuite, nous mettons à jour les propriétés left et top de myImage en utilisant les nouvelles coordonnées récupérées. Cela déplace l'image à l'endroit où vous avez cliqué dans le cadre myDiv.

Veuillez noter que pour que le déplacement fonctionne correctement, l'image doit être initialement positionnée à l'intérieur du cadre. Dans l'exemple donné, nous avons ajouté un attribut id="myImage" à l'élément <img> pour le sélectionner plus facilement dans le JavaScript.

4) Utiliser des images carrés

Supposons qu'on travaille avec 1 image 600x400 ("image.jpg") découpées en 6 images carré 200x200 (tuiles) ("1.jpg", "2.jpg", "3.jpg", "4.jpg" et "5.jpg"). Comment faut-il modifier le programme précédent pour afficher une image carré 200x200 (tuile) de façon responsive, à l'endroit où l'on clique, dans le cadre myDiv ?

Pour afficher une tuile 200x200 et non plus une tuile 300x200, à l'endroit où l'on clique, dans le cadre myDiv, il suffit de modifier les propriétés de style de #myDiv img {
width: 33.33%; /* 33.33% de la largeur du cadre */
height: 50.00%; /* 50.00% de la hauteur du cadre */
top: 33.33%; /* Position en haut du cadre */
left: 0.0%; /* Position à gauche du cadre */}



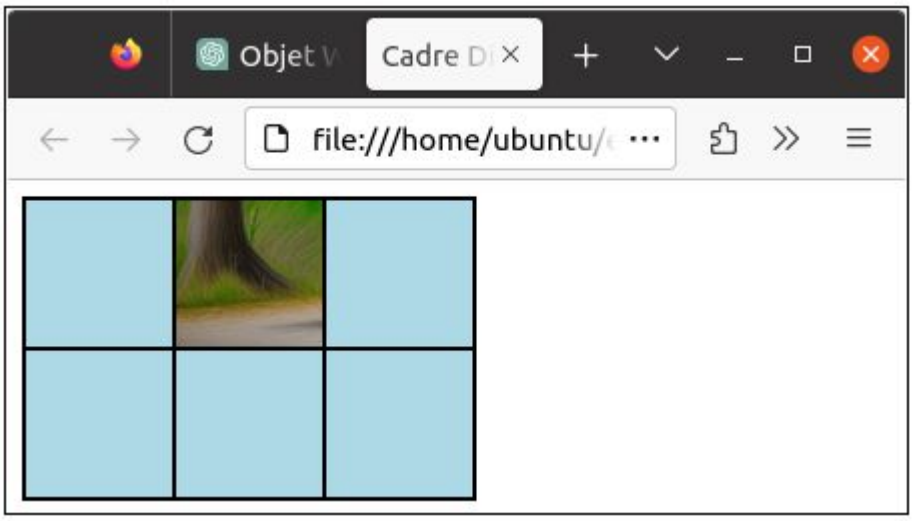

```
<!DOCTYPE html>
<html>
<head>
  <title>Cadre Div avec Image</title>
  <style>
    #myDiv {
      width: 50vw; /* 50% de la largeur de la fenêtre */
      height: 33.33vw; /* 33.33% de la largeur de la fenêtre */
      background-color: lightblue;
      border: 1px solid black; /* Bordure noire de 1 pixel */
      position: relative; /* Position relative pour positionner
l'image */
    }

    #myDiv img {
      position: absolute; /* Position absolue pour positionner l'image
*/
      width: 33.33%; /* 33.33% de la largeur du cadre */
```

```
      height: 50.00%; /* 50.00% de la hauteur du cadre */
      top: 33.33%; /* Position en haut du cadre */
      left: 0.0%; /* Position à gauche du cadre */
    }
  </style>
</head>
<body>
  <div id="myDiv">
    <img id="myImage" src="1.jpg" alt="Image" />
  </div>

  <script>
    var myDiv = document.getElementById("myDiv");
    var myImage = document.getElementById("myImage");

    myDiv.addEventListener("click", function(event) {
      var offsetX = event.offsetX;
      var offsetY = event.offsetY;

      myImage.style.left = offsetX + "px";
      myImage.style.top = offsetY + "px";
    });
  </script>
</body>
</html>
```

5) Utiliser une grille 3x2

comment découper le cadre myDiv en 6 sous cadres situés à 0%, 33,33% 66,66% et 100% en largeur et 0%, 50% et 100% en hauteur puis placer l'image "1.jpg" à l'intérieur du sous cadre que l'on clique ?

Pour découper le cadre myDiv en 6 sous-cadres et placer l'image "1.jpg" à l'intérieur du sous-cadre cliqué, vous pouvez utiliser une combinaison de CSS et de JavaScript. Voici un exemple de code qui illustre cette approche :




```
<!DOCTYPE html>
<html>
<head>
  <title>Cadre Div avec Sous-cadres</title>
  <style>
    #myDiv {
      width: 50vw; /* 50% de la largeur de la fenêtre */
      height: 33.33vw; /* 33.33% de la largeur de la fenêtre */
      background-color: lightblue;
      border: 1px solid black; /* Bordure noire de 1 pixel */
      display: flex;
      flex-wrap: wrap;
    }

    .subDiv {
      width: 33.33%; /* 33.33% de la largeur du cadre */
      height: 50%; /* 50% de la hauteur du cadre */
      border: 1px solid black; /* Bordure noire de 1 pixel */
      box-sizing: border-box;
      position: relative;
    }

    #myDiv img {
```

```
      position: absolute; /* Position absolue pour positionner l'image
*/
      width: 100%; /* 100% de la largeur du sous-cadre */
      height: 100%; /* 100% de la hauteur du sous-cadre */
      top: 0;
      left: 0;
      object-fit: cover;
    }
  </style>
</head>
<body>
  <div id="myDiv">
    <div class="subDiv" onclick="placeImage(this)"></div>
    <div class="subDiv" onclick="placeImage(this)"></div>
    <div class="subDiv" onclick="placeImage(this)"></div>
    <div class="subDiv" onclick="placeImage(this)"></div>
    <div class="subDiv" onclick="placeImage(this)"></div>
    <div class="subDiv" onclick="placeImage(this)"></div>
  </div>

  <script>
    function placeImage(element) {
      var image = document.createElement("img");
      image.src = "1.jpg";
      element.appendChild(image);
    }
  </script>
</body>
</html>
```

Dans ce code, nous utilisons le CSS pour découper le cadre myDiv en 6 sous-cadres. Chaque sous-cadre est représenté par la classe CSS subDiv. Nous utilisons display: flex et flex-wrap: wrap pour créer une disposition flexible qui permet aux sous-cadres de s'ajuster automatiquement en fonction de la taille du cadre parent.

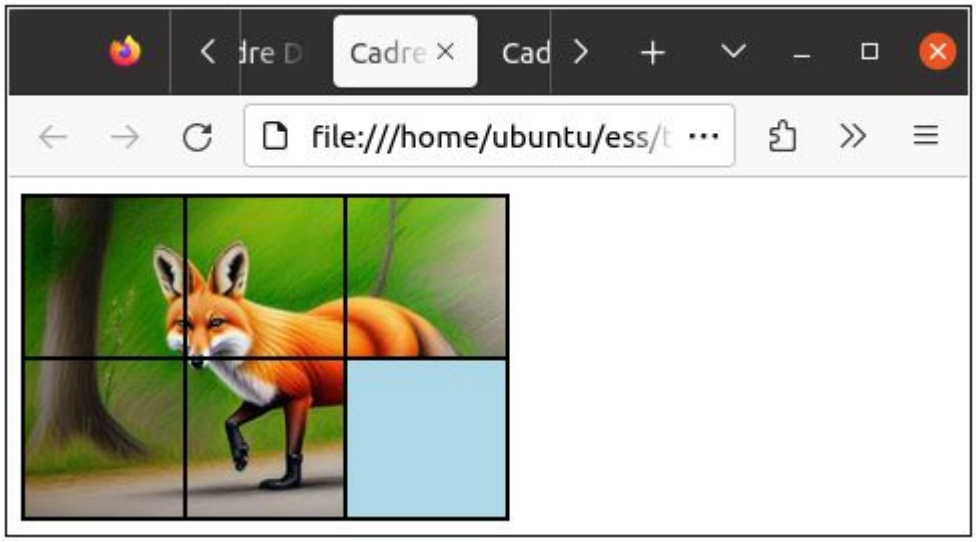
Dans la balise de chaque sous-cadre, nous avons ajouté un attribut onclick qui appelle la fonction JavaScript placeImage(this) lorsqu'on clique sur le sous-cadre. La fonction placeImage crée dynamiquement un élément <img> et définit son attribut src sur "1.jpg". Ensuite, l'image est ajoutée en tant qu'enfant du sous-cadre cliqué en utilisant la méthode appendChild.

L'image sera placée à l'intérieur du sous-cadre qui a été cliqué. Vous pouvez ajuster les styles CSS selon vos préférences pour obtenir l'apparence souhaitée.

Assurez-vous de remplacer "1.jpg" par le chemin ou l'URL réel de votre image. Cela permettra d'afficher l'image à l'intérieur du sous-cadre.

6) Utiliser une grille cliquable 3x2

Comment modifier le script précédent pour que :
si on clique dans le sous cadre 1, on affiche 1.jpg dans ce sous cadre
si on clique dans le sous cadre 2, on affiche 2.jpg dans ce sous cadre
si on clique dans le sous cadre 3, on affiche 3.jpg dans ce sous cadre
si on clique dans le sous cadre 4, on affiche 4.jpg dans ce sous cadre
si on clique dans le sous cadre 5, on affiche 5.jpg dans ce sous cadre
si on clique dans le sous cadre 6, on affiche 6.jpg dans ce sous cadre



Pour afficher des images différentes dans chaque sous-cadre lorsqu'ils sont cliqués, vous pouvez utiliser une approche basée sur l'index du sous-cadre. Voici comment vous pouvez modifier le script :

```
<!DOCTYPE html>
<html>
<head>
  <title>Cadre Div avec Sous-cadres</title>
  <style>
    #myDiv {
      width: 50vw; /* 50% de la largeur de la fenêtre */
      height: 33.33vw; /* 33.33% de la largeur de la fenêtre */
      background-color: lightblue;
      border: 1px solid black; /* Bordure noire de 1 pixel */
      display: flex;
      flex-wrap: wrap;
    }

    .subDiv {
      width: 33.33%; /* 33.33% de la largeur du cadre */
      height: 50%; /* 50% de la hauteur du cadre */
      border: 1px solid black; /* Bordure noire de 1 pixel */
      box-sizing: border-box;
```

```
      position: relative;
    }

    #myDiv img {
      position: absolute; /* Position absolue pour positionner l'image
*/
      width: 100%; /* 100% de la largeur du sous-cadre */
      height: 100%; /* 100% de la hauteur du sous-cadre */
      top: 0;
      left: 0;
      object-fit: cover;
    }
  </style>
</head>
<body>
  <div id="myDiv">
    <div class="subDiv" onclick="placeImage(this, '4.jpg')"></div>
    <div class="subDiv" onclick="placeImage(this, '5.jpg')"></div>
    <div class="subDiv" onclick="placeImage(this, '6.jpg')"></div>
    <div class="subDiv" onclick="placeImage(this, '1.jpg')"></div>
    <div class="subDiv" onclick="placeImage(this, '2.jpg')"></div>
    <div class="subDiv" onclick="placeImage(this, '3.jpg')"></div>
  </div>

  <script>
    function placeImage(element, imageUrl) {
      var image = document.createElement("img");
      image.src = imageUrl;
      element.appendChild(image);
    }
  </script>
</body>
</html>
```

Dans ce code, nous avons ajouté un deuxième argument à la fonction placeImage. Cet argument imageUrl correspond à l'URL de l'image à afficher dans le sous-cadre cliqué.

Lorsque vous cliquez sur un sous-cadre, la fonction placeImage est appelée avec le sous-cadre correspondant et l'URL de l'image appropriée. La fonction crée ensuite dynamiquement un élément <img> et définit son attribut src sur l'URL fournie. L'image est ensuite ajoutée en tant qu'enfant du sous-cadre cliqué.

Par exemple, lorsque vous cliquez sur le sous-cadre 1, la fonction placeImage(this, '1.jpg') est appelée, ce qui affiche l'image "1.jpg" dans ce sous-cadre. De même, les autres sous-cadres afficheront les images correspondantes selon le schéma décrit.

Assurez-vous d'avoir les images correspondantes (1.jpg, 2.jpg, 3.jpg, etc.) dans le même répertoire que votre fichier HTML, ou ajustez l'URL en fonction de l'emplacement réel des images sur votre serveur.

Le script que vous avez fourni est un exemple de code HTML et JavaScript qui crée un cadre divisé en sous-cadres et permet d'afficher des images différentes dans chaque sous-cadre lorsqu'ils sont cliqués.

En résumé, voici comment le script fonctionne :

1. L'élément avec l'ID myDiv est le cadre principal qui contiendra les sous-cadres. Il a une largeur de 50% de la largeur de la fenêtre (width: 50vw) et une hauteur de 33.33% de la largeur de la fenêtre (height: 33.33vw). Il a également une couleur de fond bleu clair (background-color: lightblue) et une bordure noire de 1 pixel (border: 1px solid black).
2. Les sous-cadres sont représentés par des éléments avec la classe subDiv. Chaque sous-cadre a une largeur de 33.33% de la largeur du cadre principal (width: 33.33%) et une hauteur de 50% de la hauteur du cadre principal (height: 50%). Ils ont également une bordure noire de 1 pixel (border: 1px solid black). Ces sous-cadres sont positionnés en utilisant flex-wrap et display: flex pour créer une disposition flexible à l'intérieur du cadre principal.
3. Les images à afficher dans les sous-cadres sont créées dynamiquement en utilisant JavaScript. Lorsqu'un sous-cadre est cliqué, la fonction placeImage est appelée avec deux arguments : l'élément du sous-cadre cliqué (element) et l'URL de l'image à afficher (imageUrl).
4. La fonction placeImage crée un nouvel élément <img> en utilisant document.createElement("img"). Ensuite, l'attribut src de l'image est défini sur l'URL fournie (image.src = imageUrl).
5. Enfin, l'image est ajoutée en tant qu'enfant de l'élément du sous-cadre cliqué en utilisant element.appendChild(image). Cela positionne l'image à l'intérieur du sous-cadre.

Ainsi, lorsque vous cliquez sur l'un des sous-cadres, la fonction placeImage est appelée, ce qui crée et affiche l'image correspondante dans le sous-cadre cliqué.

quelle est l'utilité du code suivant :

```
#myDiv img {
position: absolute; /* Position absolue pour positionner l'image */
width: 100%; /* 100% de la largeur du sous-cadre */
```

```
height: 100%; /* 100% de la hauteur du sous-cadre */
top: 0;
left: 0;
object-fit: cover; }
```

Ce code correspond à la définition des styles CSS pour les images à l'intérieur des sous-cadres. Explorons l'utilité de chaque propriété :

- position: absolute;: Cette propriété positionne l'image de manière absolue à l'intérieur du sous-cadre. Cela signifie que l'image est positionnée par rapport à son premier parent qui a une position relative ou absolue. Dans ce cas, l'élément parent du <img> est le sous-cadre ( ), qui a une position relative. Ainsi, l'image sera positionnée de manière absolue à l'intérieur de ce sous-cadre.
- width: 100%;: Cette propriété définit la largeur de l'image à 100% de la largeur du sous-cadre. Cela permet à l'image de s'ajuster automatiquement à la largeur du sous-cadre, en occupant tout l'espace disponible horizontalement.
- height: 100%;: Cette propriété définit la hauteur de l'image à 100% de la hauteur du sous-cadre. Cela permet à l'image de s'ajuster automatiquement à la hauteur du sous-cadre, en occupant tout l'espace disponible verticalement.
- top: 0; et left: 0;: Ces propriétés spécifient les coordonnées de position de l'image à l'intérieur du sous-cadre. En définissant les valeurs 0, l'image est positionnée au coin supérieur gauche du sous-cadre.
- object-fit: cover;: Cette propriété définit comment l'image doit être ajustée à l'intérieur de son conteneur (dans ce cas, le sous-cadre). La valeur cover indique que l'image doit être redimensionnée pour remplir complètement le conteneur tout en préservant son ratio d'aspect. Cela signifie que l'image peut être rognée (coupée) si nécessaire pour s'adapter parfaitement au sous-cadre sans déformer l'image.

En résumé, ces styles CSS assurent que l'image à l'intérieur du sous-cadre occupera tout l'espace disponible en respectant les dimensions du sous-cadre, sera positionnée au coin supérieur gauche du sous-cadre et sera ajustée pour remplir complètement le sous-cadre sans déformation.

le "img" qui figure dans l'instruction
*var image = document.createElement("**img**"); correspond au img qui figure dans #myDiv **img** { ?*

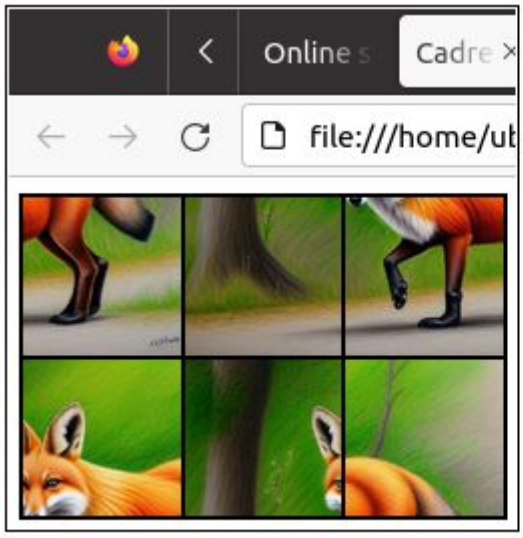
Oui, tout à fait. Dans l'instruction

var image = document.createElement("**img**");,
le "img" fait référence à l'élément <img> que vous avez défini dans votre CSS avec la règle #myDiv **img**.

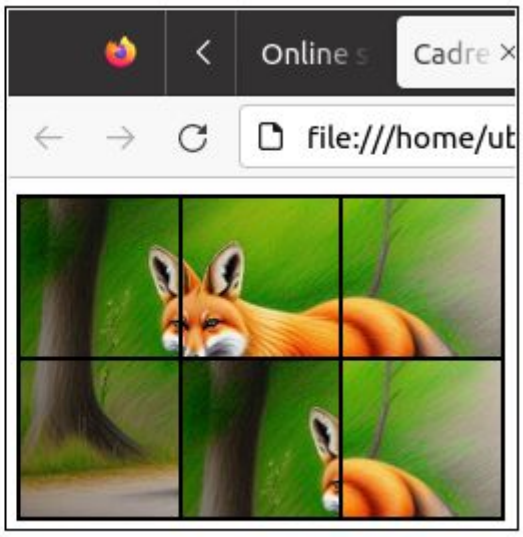
En CSS, #myDiv img cible tous les éléments <img> qui se trouvent à l'intérieur de l'élément avec l'ID myDiv. C'est pourquoi, lorsque vous créez un nouvel élément <img> en JavaScript avec document.createElement("img"), il hérite automatiquement des styles CSS définis pour les images à l'intérieur de #myDiv.
Ainsi, lorsque vous utilisez var image = document.createElement("img");, vous créez un nouvel élément <img> qui aura les mêmes styles que ceux définis pour les images à l'intérieur de #myDiv. Cela garantit que la nouvelle image créée aura la même apparence et les mêmes propriétés CSS que les autres images dans votre mise en page.

7) Préafficher dans une grille cliquable 3x2

Ne peut on pas compléter le code suivant pour que lors de l'ouverture de la page html les images 1.jpg, 2.jpg, 3.jpg, 4.jpg, 5.jpg et 6.jpg, soient affichées dans un ordre donné dans tous les sous cadres ?



au départ



après clic des sous-cadres 1 à 4

Vous pouvez utiliser la fonction `placeImageInAllSubDivs` pour placer différentes images dans lessous-cadres au moment du chargement de la page. Voici comment vous pouvez le faire :

```
<!DOCTYPE html>
<html>
<head>
 <title>Cadre Div avec Sous-cadres</title>
 <style>
  #myDiv {
   width: 50vw; /* 50% de la largeur de la fenêtre */
   height: 33.33vw; /* 33.33% de la largeur de la fenêtre */
   background-color: lightblue;
   border: 1px solid black; /* Bordure noire de 1 pixel */
   display: flex;
   flex-wrap: wrap;
  }

  .subDiv {
   width: 33.33%; /* 33.33% de la largeur du cadre */
   height: 50%; /* 50% de la hauteur du cadre */
   border: 1px solid black; /* Bordure noire de 1 pixel */
   box-sizing: border-box;
   position: relative;
  }
```

```
    #myDiv img {
      position: absolute; /* Position absolue pour positionner l'image */
      width: 100%; /* 100% de la largeur du sous-cadre */
      height: 100%; /* 100% de la hauteur du sous-cadre */
      top: 0;
      left: 0;
      object-fit: cover;
    }
  </style>
</head>
<body>
  <div id="myDiv">
    <div class="subDiv" onclick="placeImage(this, '4.jpg')"></div>
    <div class="subDiv" onclick="placeImage(this, '5.jpg')"></div>
    <div class="subDiv" onclick="placeImage(this, '6.jpg')"></div>
    <div class="subDiv" onclick="placeImage(this, '1.jpg')"></div>
    <div class="subDiv" onclick="placeImage(this, '2.jpg')"></div>
    <div class="subDiv" onclick="placeImage(this, '3.jpg')"></div>
  </div>

  <script>
    function placeImage(element, imageUrl) {
      var image = document.createElement("img");
      image.src = imageUrl;
      element.appendChild(image);
    }

    function placeImageInAllSubDivs() {
      var subDivs = document.getElementsByClassName("subDiv");
      var images = ['3.jpg', '1.jpg', '2.jpg', '5.jpg', '4.jpg', '6.jpg']; // Tableau des
images correspondant aux paires de sous-cadres
      for (var i = 0; i < subDivs.length; i++) {
        placeImage(subDivs[i], images[i]);
      }
    }

    placeImageInAllSubDivs(); // Appel initial pour placer les images lors du
chargement de la page
  </script>
</body>

</html>
```

8) Fabriquer 6 tuiles

On utilise le programme dec6.c suivant pour découper un fichier "image.bmp" de taille 600x400 en 6 fichiers "1.jpg", "2.jpg", "3.jpg", "4.jpg", "5.jpg" et "6.jpg" de taille 200x200 chacun.

```
// gcc -Wall dec6.c -o dec6
// ./dec6
```

Puis on utilise la commande suivante pour convertir les 6 fichiers bmp au format jpg

```
mogrify -format jpg -quality 80 *.bmp
```

```c
// gcc -Wall dec6.c -o dec6
// ./dec6

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

// Structure pour représenter un pixel
typedef struct {
    uint8_t blue;
    uint8_t green;
    uint8_t red;
} Pixel;

// Fonction pour charger une image BMP
Pixel* chargerImageBMP(const char* nomFichier, int* largeur, int* hauteur) {
    FILE* fichier = fopen(nomFichier, "rb");
    if (!fichier) {
        printf("Impossible de charger l'image.\n");
        return NULL;
    }

    // Lire l'en-tête du fichier BMP
    fseek(fichier, 18, SEEK_SET);
    fread(largeur, sizeof(int), 1, fichier);
    fread(hauteur, sizeof(int), 1, fichier);
    fseek(fichier, 54, SEEK_SET);

    // Allouer la mémoire pour les pixels
    int tailleImage = (*largeur) * (*hauteur);
    Pixel* pixels = (Pixel*)malloc(tailleImage * sizeof(Pixel));

    // Lire les pixels de l'image
    fread(pixels, sizeof(Pixel), tailleImage, fichier);

    fclose(fichier);

    return pixels;
}

// Fonction pour sauvegarder une image BMP
void sauvegarderImageBMP(const char* nomFichier, const Pixel* pixels, int largeur, int hauteur) {
    FILE* fichier = fopen(nomFichier, "wb");
    if (!fichier) {
        printf("Impossible de sauvegarder l'image.\n");
        return;
    }

    // Écrire l'en-tête du fichier BMP
    int tailleImage = largeur * hauteur;
    int tailleFichier = tailleImage * sizeof(Pixel) + 54;
    uint8_t enTete[54] = {
        0x42, 0x4D,                     // Signature du fichier BMP
        tailleFichier & 0xFF, (tailleFichier >> 8) & 0xFF, (tailleFichier >> 16) & 0xFF, (tailleFichier >> 24) & 0xFF,  // Taille du fichier
        0x00, 0x00, 0x00, 0x00,         // Réservé
        0x36, 0x00, 0x00, 0x00,         // Offset des données de l'image
        0x28, 0x00, 0x00, 0x00,         // Taille de l'en-tête (40 bytes)
        largeur & 0xFF, (largeur >> 8) & 0xFF, (largeur >> 16) & 0xFF, (largeur >> 24) & 0xFF,  // Largeur de l'image
        hauteur & 0xFF, (hauteur >> 8) & 0xFF, (hauteur >> 16) & 0xFF, (hauteur >> 24) & 0xFF,  // Hauteur de l'image
        0x01, 0x00,                     // Planes (1)
        0x18, 0x00,                     // Bits par pixel (24 bits, 8 bits pour chaque canal RGB)
        0x00, 0x00, 0x00, 0x00,         // Compression (sans compression)
        0x00, 0x00, 0x00, 0x00,         // Taille des données de l'image (0 pour les images non compressées)
        0x13, 0x0B, 0x00, 0x00,         // Résolution horizontale (2835 pixels/mètre)
        0x13, 0x0B, 0x00, 0x00,         // Résolution verticale (2835 pixels/mètre)
        0x00, 0x00, 0x00, 0x00,         // Nombre de couleurs de la palette (0 pour les images non indexées)
        0x00, 0x00, 0x00, 0x00          // Nombre de couleurs importantes (0 pour toutes les couleurs importantes)
    };
    fwrite(enTete, sizeof(uint8_t), 54, fichier);

    // Écrire les pixels de l'image
    fwrite(pixels, sizeof(Pixel), tailleImage, fichier);
```

```
    fclose(fichier);
}

int main() {
    // Charger l'image
    int largeur, hauteur;
    Pixel* image = chargerImageBMP("image.bmp", &largeur, &hauteur);
    if (!image) {
        return 1;
    }

    // Vérifier les dimensions de l'image
    if (largeur != 600 || hauteur != 400) {
        printf("L'image doit avoir une dimension de 600x400 pixels.\n");
        free(image);
        return 1;
    }

    // Créer les régions d'intérêt (ROI) pour les 6 images découpées
    int largeurDecoupee = 200;
    int hauteurDecoupee = 200;

    // Allouer la mémoire pour les images découpées
    Pixel* image1 = (Pixel*)malloc(largeurDecoupee * hauteurDecoupee * sizeof(Pixel));
    Pixel* image2 = (Pixel*)malloc(largeurDecoupee * hauteurDecoupee * sizeof(Pixel));
    Pixel* image3 = (Pixel*)malloc(largeurDecoupee * hauteurDecoupee * sizeof(Pixel));
    Pixel* image4 = (Pixel*)malloc(largeurDecoupee * hauteurDecoupee * sizeof(Pixel));
    Pixel* image5 = (Pixel*)malloc(largeurDecoupee * hauteurDecoupee * sizeof(Pixel));
    Pixel* image6 = (Pixel*)malloc(largeurDecoupee * hauteurDecoupee * sizeof(Pixel));

    // Copier les régions d'intérêt dans les images découpées
    for (int y = 0; y < hauteurDecoupee; y++) {
        for (int x = 0; x < largeurDecoupee; x++) {
            image1[y * largeurDecoupee + x] = image[y * largeur + x];
            image2[y * largeurDecoupee + x] = image[y * largeur + x + largeurDecoupee];
            image3[y * largeurDecoupee + x] = image[y * largeur + x + 2 * largeurDecoupee];
            image4[y * largeurDecoupee + x] = image[(y + hauteurDecoupee) * largeur + x];
            image5[y * largeurDecoupee + x] = image[(y + hauteurDecoupee) * largeur + x + largeurDecoupee];
            image6[y * largeurDecoupee + x] = image[(y + hauteurDecoupee) * largeur + x + 2 * largeurDecoupee];
        }
    }

    // Sauvegarder les images découpées
    sauvegarderImageBMP("1.bmp", image1, largeurDecoupee, hauteurDecoupee);
    sauvegarderImageBMP("2.bmp", image2, largeurDecoupee, hauteurDecoupee);
    sauvegarderImageBMP("3.bmp", image3, largeurDecoupee, hauteurDecoupee);
    sauvegarderImageBMP("4.bmp", image4, largeurDecoupee, hauteurDecoupee);
    sauvegarderImageBMP("5.bmp", image5, largeurDecoupee, hauteurDecoupee);
    sauvegarderImageBMP("6.bmp", image6, largeurDecoupee, hauteurDecoupee);

    // Libérer la mémoire
    free(image);
    free(image1);
    free(image2);
    free(image3);
    free(image4);
    free(image5);
    free(image6);

    printf("Les images ont été découpées avec succès.\n");

    return 0;
}
```

9) Créer un squelette de jeu de pousse pousse

Le code suivant réalise le squelette d'un pousse pousse responsive de format 3/2, de 3 cases de large et 2 cases de haut. Les 6 cases du pousse pousse correspondent aux images carrées 1.jpg, 2.jpg, 3.jpg, 4.jpg, 5.jpg et 6.jpg,

Au départ, les images sont affichées dans le désordre. Quand on clique une case, la bonne image s'affiche dans cette case.

```
<!DOCTYPE html>
<html>
<head>
  <title>Cadre Div Responsive</title>
  <style>

.container {
  display: flex;
  justify-content: center;
  align-items: center;
  max-width: 100%;
  height: auto;
}

    #myDiv {
      width: 50vw;
      height: calc(33.33vw);
      background-color: white;
      border: 0px solid black;
      position: relative;
    }
  </style>
</head>
<body>
<p>Le corbeau et le renard</p>
<div class="container">
  <div id="myDiv"></div>
</div>
<p>Le corbeau et le renard</p>
  <script>
    function createSubFrames() {
      var myDiv = document.getElementById("myDiv");
      var divWidth = myDiv.offsetWidth;
      var divHeight = myDiv.offsetHeight;

      var subFrame1 = createSubFrame(divWidth * 0, divHeight * 0,
divWidth * 0.3333, divHeight * 0.5);
      subFrame1.id = "subFrame1";
      myDiv.appendChild(subFrame1);
```

```
      var subFrame2 = createSubFrame(divWidth * 0.3333, divHeight * 0,
divWidth * 0.3333, divHeight * 0.5);
      subFrame2.id = "subFrame2";
      myDiv.appendChild(subFrame2);

      var subFrame3 = createSubFrame(divWidth * 0.6666, divHeight * 0,
divWidth * 0.3333, divHeight * 0.5);
      subFrame3.id = "subFrame3";
      myDiv.appendChild(subFrame3);

      var subFrame4 = createSubFrame(divWidth * 0, divHeight * 0.5,
divWidth * 0.3333, divHeight * 0.5);
      subFrame4.id = "subFrame4";
      myDiv.appendChild(subFrame4);

      var subFrame5 = createSubFrame(divWidth * 0.3333, divHeight *
0.5, divWidth * 0.3333, divHeight * 0.5);
      subFrame5.id = "subFrame5";
      myDiv.appendChild(subFrame5);

      var subFrame6 = createSubFrame(divWidth * 0.6666, divHeight *
0.5, divWidth * 0.3333, divHeight * 0.5);
      subFrame6.id = "subFrame6";
      myDiv.appendChild(subFrame6);

      loadImages([subFrame6, subFrame5, subFrame4, subFrame3,
subFrame2, subFrame1], ['1.jpg', '2.jpg', '3.jpg', '4.jpg', '5.jpg',
'6.jpg']);
      addClickListeners();
    }

    function createSubFrame(left, top, width, height) {
      var subFrame = document.createElement("div");
      subFrame.style.position = "absolute";
      subFrame.style.left = left + "px";
      subFrame.style.top = top + "px";
      subFrame.style.width = width + "px";
      subFrame.style.height = height + "px";
      subFrame.style.backgroundSize = "cover";
      subFrame.style.backgroundRepeat = "no-repeat";
      subFrame.style.backgroundColor = "lightgreen";
      subFrame.style.border = "1px solid black";

      return subFrame;
    }

    function loadImages(subFrames, images) {
      for (var i = 0; i < subFrames.length; i++) {
        subFrames[i].style.backgroundImage = "url('" + images[i] +
"')";
      }
    }
```

```
    function addClickListeners() {
      var subFrame1 = document.getElementById("subFrame1");
      subFrame1.addEventListener("click", function() {
        this.style.backgroundImage = "url('1.jpg')";
      });

      var subFrame2 = document.getElementById("subFrame2");
      subFrame2.addEventListener("click", function() {
        this.style.backgroundImage = "url('2.jpg')";
      });

      var subFrame3 = document.getElementById("subFrame3");
      subFrame3.addEventListener("click", function() {
        this.style.backgroundImage = "url('3.jpg')";
      });

      var subFrame4 = document.getElementById("subFrame4");
      subFrame4.addEventListener("click", function() {
        this.style.backgroundImage = "url('4.jpg')";
      });

      var subFrame5 = document.getElementById("subFrame5");
      subFrame5.addEventListener("click", function() {
        this.style.backgroundImage = "url('5.jpg')";
      });

      var subFrame6 = document.getElementById("subFrame6");
      subFrame6.addEventListener("click", function() {
        this.style.backgroundImage = "url('6.jpg')";
      });
    }

    window.addEventListener("load", createSubFrames);
  </script>
</body>
</html>
```

Ce programme est une page HTML qui crée un cadre div responsive avec des sous-éléments cliquables affichant des images lorsqu'ils sont cliqués. Voici comment le programme fonctionne étape par étape :

1. La balise <style> contient les styles CSS qui seront appliqués aux éléments de la page. Les sélecteurs .container et #myDiv définissent les styles du conteneur et du cadre div principal.
2. Dans le corps de la page (<body>), il y a un paragraphe <p> avec le texte "Le corbeau et le renard". Ce texte apparaîtra avant le cadre div.
3. Ensuite, il y a un conteneur avec la classe CSS "container". Ce conteneur est utilisé pour centrer le cadre div et son contenu à l'écran à

l'aide des propriétés CSS display: flex; justify-content: center; align-items: center;.

4. À l'intérieur du conteneur, il y a un cadre div avec l'ID "myDiv". Ce cadre div sera utilisé pour afficher les sous-éléments cliquables.
5. Ensuite, il y a un autre paragraphe <p> avec le même texte "Le corbeau et le renard". Ce texte apparaîtra après le cadre div.
6. Le script JavaScript contenu dans la balise <script> s'exécute lorsque la page est chargée (window.addEventListener("load", createSubFrames)). Cette fonction createSubFrames() est responsable de la création des sous-éléments du cadre div.
7. La fonction createSubFrames() commence par récupérer la référence du cadre div principal à l'aide de document.getElementById("myDiv"). Ensuite, elle obtient la largeur et la hauteur du cadre div.
8. La fonction createSubFrame() est appelée pour créer chaque sous-élément. Elle crée un élément et définit ses propriétés CSS (position, dimensions, arrière-plan, bordure, etc.) en fonction des paramètres left, top, width et height passés à la fonction. Chaque sous-élément est ajouté au cadre div principal (myDiv.appendChild(subFrame)).
9. La fonction loadImages() est appelée pour charger les images dans chaque sous-élément. Elle prend en paramètre un tableau d'éléments subFrames et un tableau d'URL d'images images. Elle parcourt ensuite les sous-éléments et définit l'image de fond de chaque sous-élément en utilisant subFrames[i].style.backgroundImage = "url('" + images[i] + "')";.
10. La fonction addClickListeners() ajoute des écouteurs d'événements de clic à chaque sous-élément. Lorsqu'un sous-élément est cliqué, la fonction correspondante est exécutée et modifie l'image de fond du sous-élément en utilisant this.style.backgroundImage = "url('image.jpg')";.
11. Lorsque la page est chargée, la fonction createSubFrames() est appelée, ce qui déclenche la création des sous-éléments, le chargement des images et l'ajout des écouteurs d'événements de clic.

En résumé, ce programme crée un cadre div avec des sous-éléments cliquables affichant des images. Lorsque vous cliquez sur un sous-élément, son image de fond est modifiée. Vous pouvez personnaliser le contenu du cadre div en modifiant les images et les comportements de clic dans le code JavaScript.