



27/10/2022

CSS 3

Exercices : spécificités - responsive



Rudy Lesur
SOFIP

Table des matières

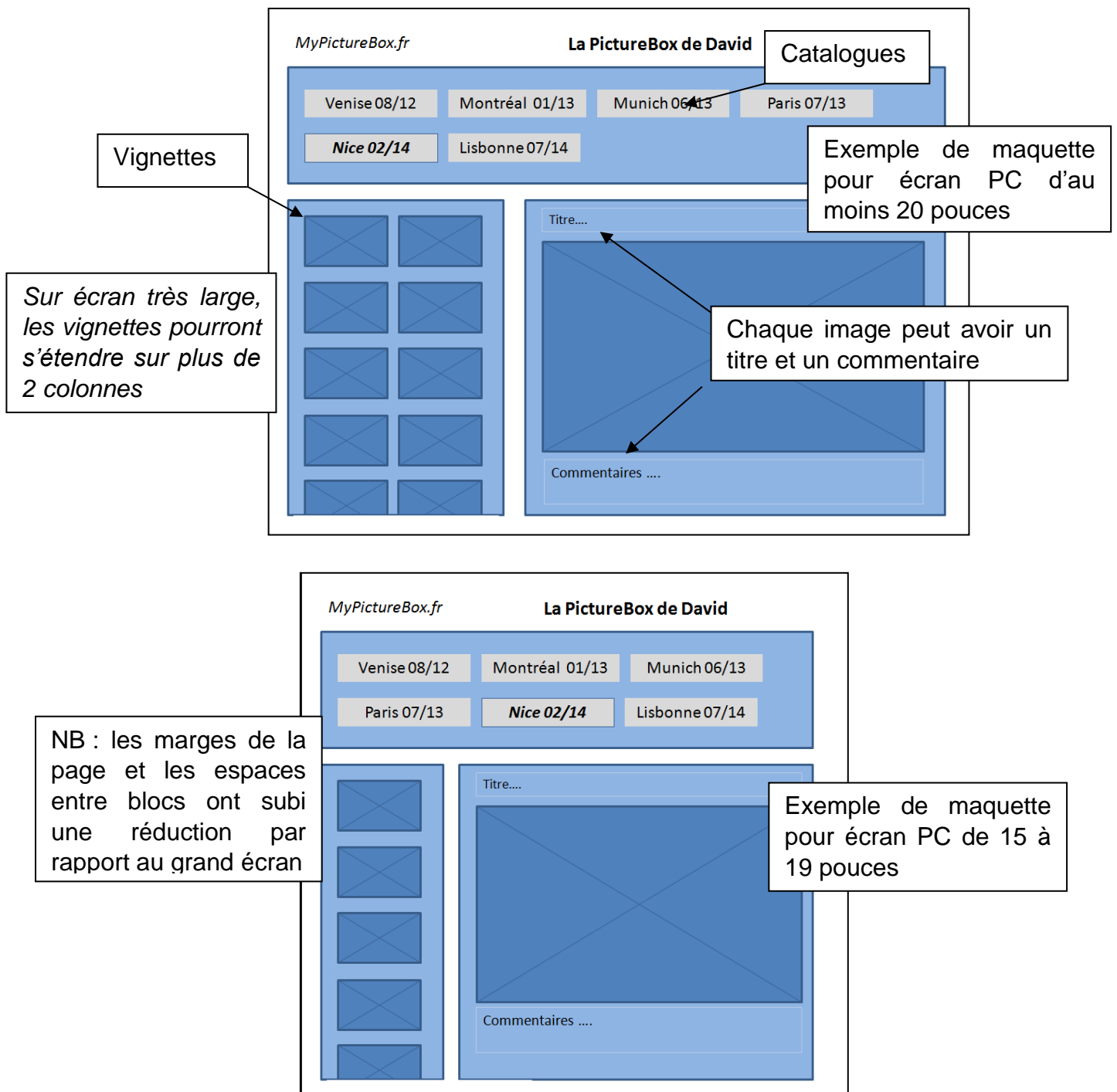
I MyPictureBox	2
II Guide de réalisation.....	7
1/ Analyse et Structuration HTML5	7
2/ Style général.....	9
1) Pour commencer	9
2) Paramétrage général.....	9
3) Titre général	10
4) Navigation	10
5) Miniatures	11
6) Grande image.....	12
7) Variante pour très grand écran	13
3/ Styles tablettes et petits écrans	14
1) Navigation	14
2) Corps de page (main et gallery)	14
4/ Styles smartphone	15
1) Navigation	15
2) Corps de page (main et gallery)	15
III Extension JavaScript.....	16
1/ Afficher dynamiquement la grande image.....	16
2/ Modifier l'apparence de l'album sélectionné	17

I MyPictureBox

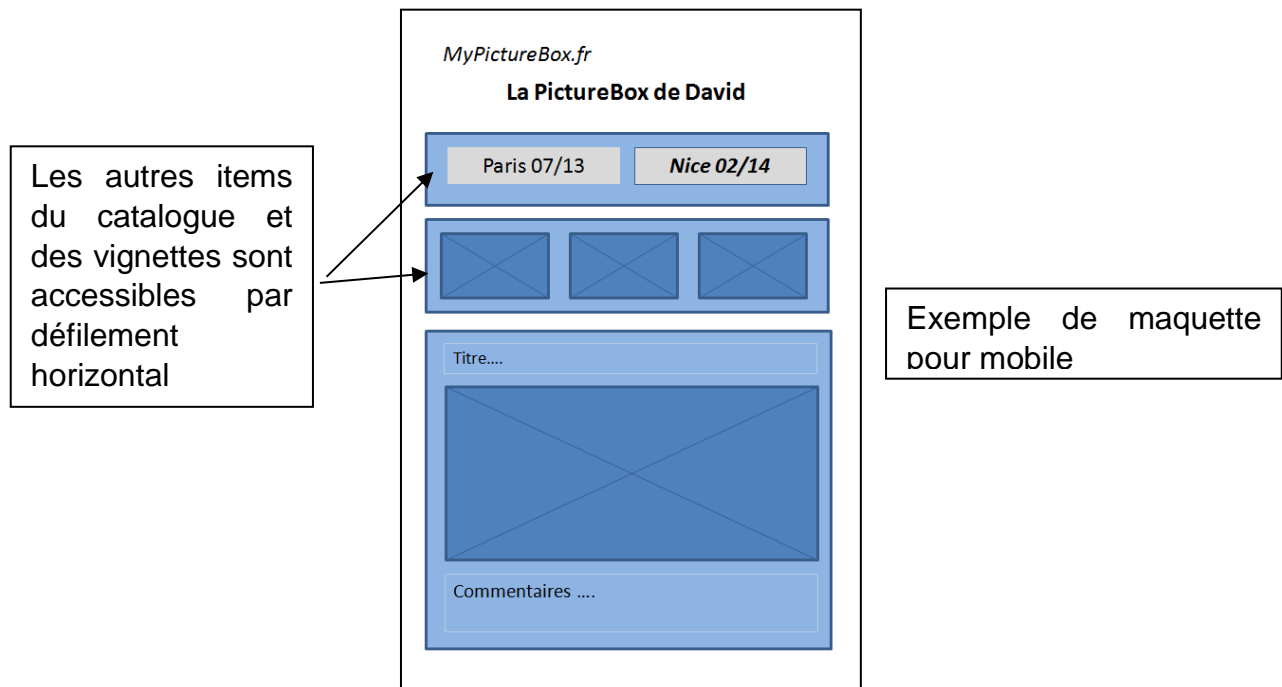
Il s'agit ici de composer une page-type de site fictif de partage de photographies (*MyPictureBox.fr*) qui pourra adopter 2 mises en page différentes, une conçue pour des écrans de PC, l'autre adaptée aux support mobiles. Cette page présente 3 parties distinctes, l'une pour choisir le catalogue de photos, la deuxième pour afficher la liste des images sous forme de vignettes, et la dernière pour visualiser en grande taille l'image sélectionnée dans la liste des vignettes.

On souhaite que ces mises en page types offrent un bon effet de zoom dans leur plage respective de dimensions.

Sur PC, l'affichage des catalogues et vignettes s'adaptera à la place disponible de manière à privilégier l'image en grande taille :



Sur support mobile, on privilégiera une mise en page « verticale » affichant en séquence catalogues, vignettes et grande image. Pour une visualisation plus directe des images (c'est bien le but de ce site !), on limitera la place occupée verticalement par catalogues et vignettes, quitte à nécessiter un défilement horizontal :



Consignes

1. Collecter des (petites) séries d'images en petite taille (vignettes) et grande taille ; on peut toujours fabriquer les vignettes à l'aide d'un outil de traitement d'image, de Microsoft Office Picture Manager ou de l'Outil Capture Windows...
2. Structurer tout d'abord le contenu de la page dans des balises HTML, sans soucis de mise en page ou de présentation finale, mais en tenant compte des variantes de mise en page selon les 2 présentations-types (anticiper sur les sélecteurs CSS des éléments HTML qui seront nécessaires plus tard - on souhaite limiter au maximum l'usage des attributs *class* à l'intérieur des balises HTML de manière à rendre indépendants contenu et présentations).
3. Réaliser dans un premier temps un affichage purement statique (rien d'événementiel : toutes les vignettes et toujours la même image en grand) sur un écran-type de PC (par exemple 1050 pixels de large) ; compléter ou modifier les styles par défaut du navigateur correspondant aux balises HTML utilisées. On peut définir dans un premier temps des dimensions absolues de blocs et marges en pixels ; préférer les dimensions relatives en em pour les textes à partir d'une taille de police de base à définir ; la mise en page sera assurée par la propriété CSS float ; le bandeau des catalogues pourra nécessiter un défilement horizontal si son contenu déborde de la place réservée ; le bandeau des vignettes pourra de même nécessiter un défilement vertical (propriétés CSS overflow et ses variantes).
4. Quand la mise en page convient pour un affichage sur 1050 pixels en largeur, reprendre toutes les dimensions absolues et les redéfinir en relatif (% et em). Penser à ajuster au besoin les espaces par défaut des éléments de manière à obtenir un bon effet de zoom. Vérifier à chaque étape la bonne adaptation de la mise en page en faisant varier la largeur du navigateur et en contrôlant le respect des dimensions d'origine à l'aide du débogueur intégré au navigateur. Pour réaliser un bon effet de zoom assurant un bon 'responsive design', il pourra être utile de rendre proportionnelle la taille des items du catalogue (les noms des villes) et la taille des vignettes, dans une certaine limite tout de même et de prévoir encore et toujours un défilement horizontal.
5. Etablir le jeu de styles nécessaire à la version pour un écran-type de smartphone ; ajouter les directives media queries dans le fichier CSS ; extraire les styles communs dans un media query supplémentaire (screen) ; généraliser en transformant les dimensions en % et em ; tester.

Extension CSS :

- appliquer une animation (rotation légère) ou une transformation (zoom, déformation) de la vignette au passage de la souris ;
- ajouter un léger effet miroir sous la grande image (grâce à un 2° élément HTML et des effets CSS de miroir, transparence, bords floutés...).

Extension Bootstrap :

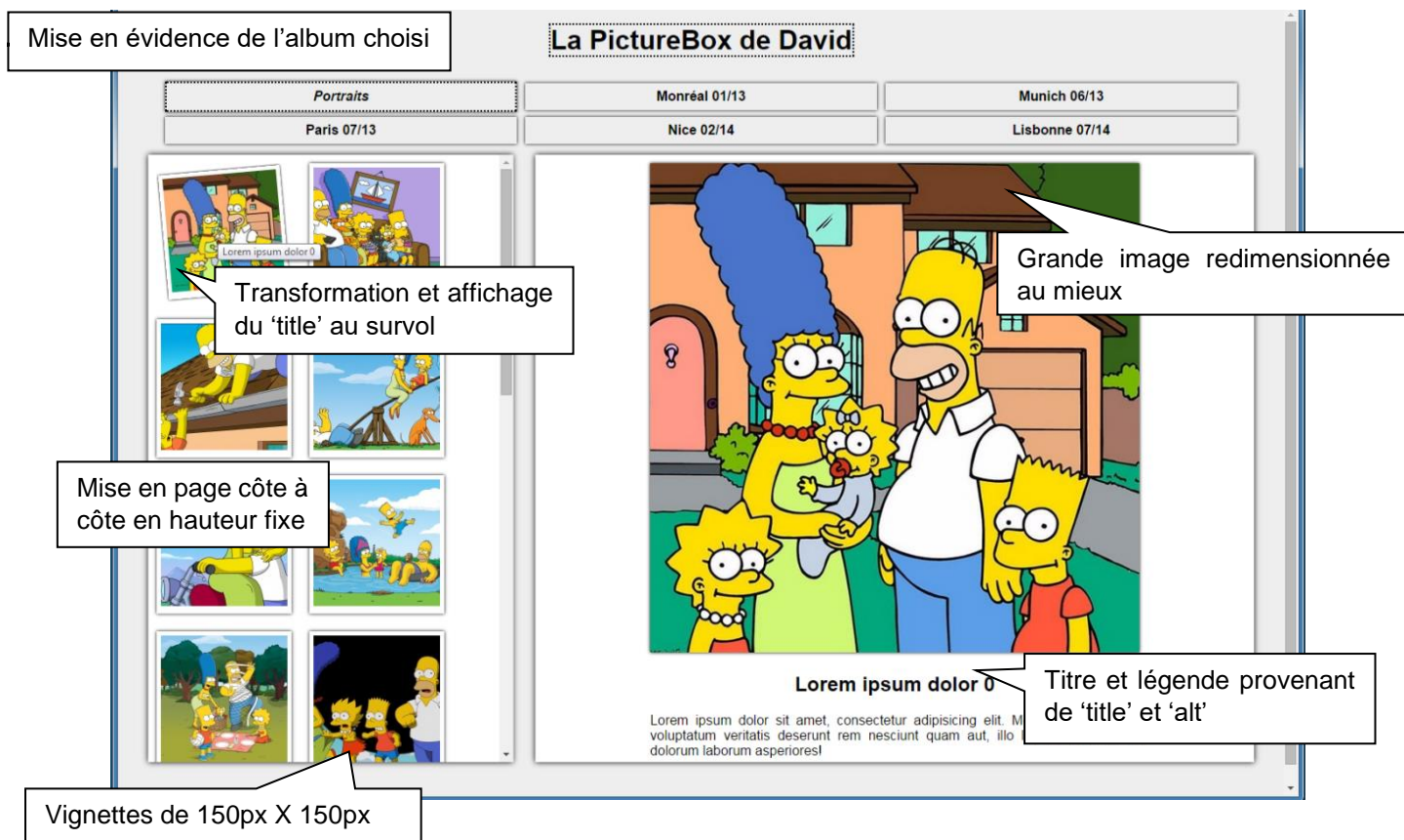
- reproduire l'exercice en réalisant tout d'abord la mise en page HTML à l'aide des structures-types utilisées par Bootstrap (<div class="row">...). Appeler ensuite les séries de classes Bootstrap sur chaque élément HTML de manière à reproduire un comportement similaire pour chacun des 4 types de périphériques identifiés par ce framework. Finaliser éventuellement la présentation en recopiant les styles définis dans la version de base (couleurs, encadrés...).

Extension JavaScript :

- au clic sur une vignette, afficher la grande image correspondant à la vignette ainsi que son titre et son commentaire (le titre pourra être stocké dans l'attribut HTML title, le commentaire en attribut HTML alt qui seront exploités par JavaScript pour injecter le contenu dans les deux éléments HTML prévus à cet effet).
- au survol d'un item d'album, afficher un curseur en forme de main (par CSS) ; au clic sur un item, modifier sa présentation (par JavaScript).

Exemples

Sur écran de PC standard (au moins 19 pouces) :



Sur petit écran ou tablette :



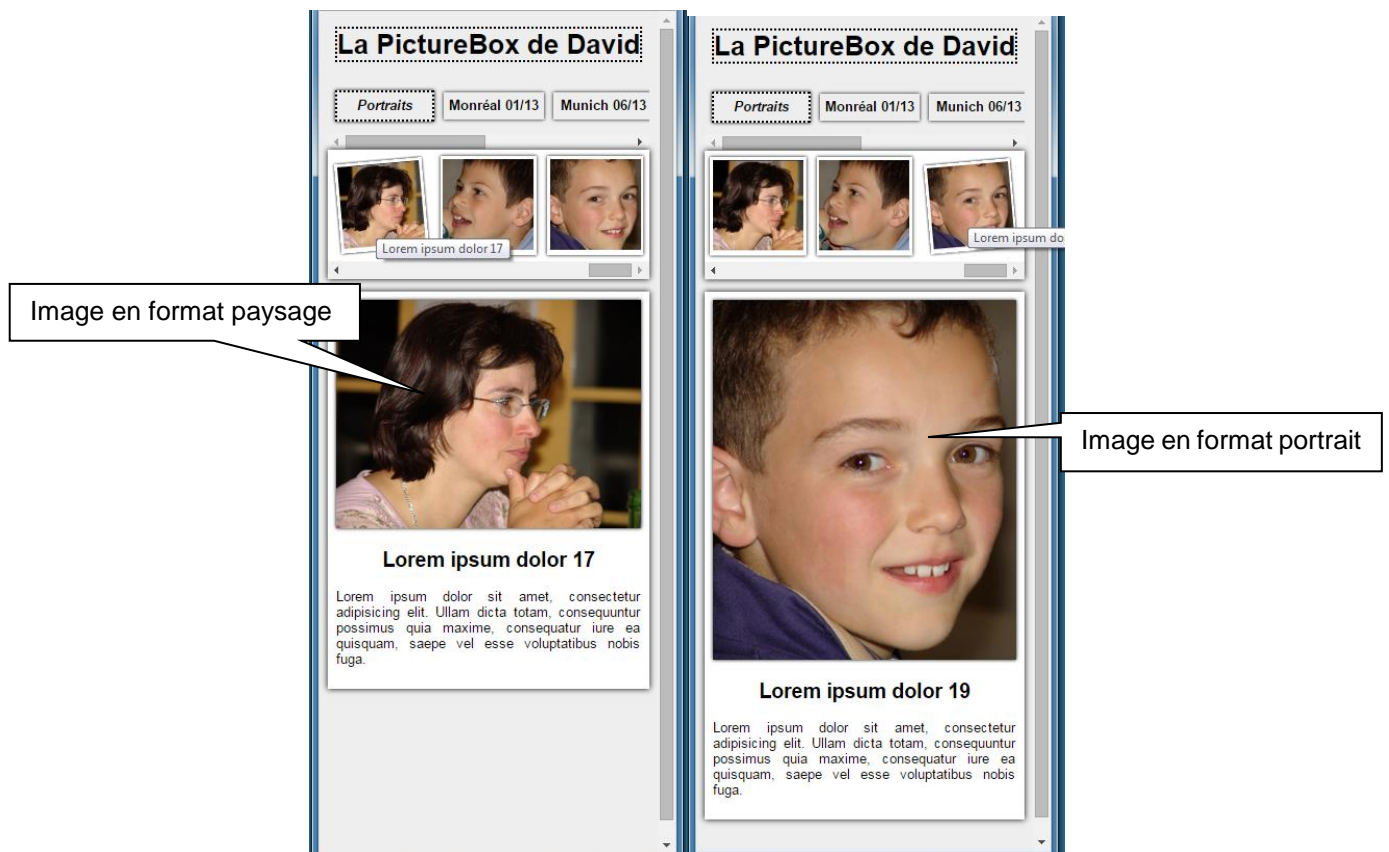
Sur écran très large :



Sur smartphone :



NB : les images ne sont pas forcément toutes de mêmes dimensions ni de mêmes proportions mais le design doit s adapter !



II Guide de réalisation

1/ Analyse et Structuration HTML5

Le titre général peut être supporté par un élément `<h1>` inclus dans une division `<header>`.

Pour pouvoir obtenir un encadrement du seul contenu du titre, il est nécessaire d'englober le texte dans un élément HTML d'affichage 'in-line' comme un `` car l'élément `<h1>`, de type d'affichage 'block', s'étend sur toute la largeur de la page.

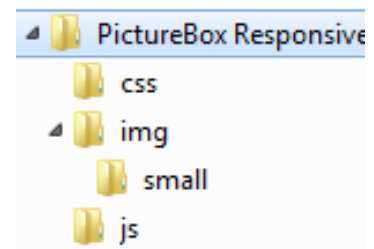
La liste des albums pourra être définie dans une structure `` ou `` et la mise en page en continu de ses éléments `` sera réalisable par CSS.

Les miniatures devront être placées soit en-dessus de l'image soit à sa gauche ; il est donc nécessaire de les inclure dans un HTML d'affichage 'block', par exemple un élément `<div>` (ou encore `<aside>` selon le sens que l'on donne à cette information).

Chaque image miniature devra être sensible au survol de la souris et au clic ; le survol pourra être géré entièrement par CSS, le clic nécessite soit du code JavaScript, soit la présence d'un élément `<a href>`, soit les deux ; comme on choisit toujours de faire simple et de laisser faire à HTML ce qu'il fait si bien, on inclura les éléments `` de chaque image dans un élément `<a href>` ce qui automatiquement les rendra sensible au clic et modifiera l'icône de curseur. Quant au traitement associé au clic, comme on souhaite rester sur la même page, le fonctionnement du `<a href>` devra être contredit par un script JavaScript (voir plus loin). Pour chaque miniature, le titre de l'image sera renseigné en attribut HTML 'title', et le commentaire, en attribut 'alt'

La grande image accompagnée de ses textes devra pouvoir être placée soit à droite, soit en dessous des miniatures ; elle fera donc l'objet d'un autre bloc, et pourquoi pas d'un élément HTML `<main>`.

La grande image est en relation avec sa miniature ; elle peut porter le même nom et être placée dans un dossier différent ou bien porter un nom contenant la même racine que celui de sa miniature (ex : 'img1mini.jpg' et 'img1.jpg') ; pour simplifier l'algorithme JavaScript, on préférera ici qu'elles portent le même nom, stockées dans des dossiers différents, d'où la structure des dossiers pour cette application.



Le titre et le commentaire ont été ici regroupés dans un élément HTML `<div>` placé sous l'image, le titre dans un élément `<h2>`, le commentaire dans un `<p>`.

Le script Javascript sera appelé en bas de page HTML (avant `</body>`) par une balise `<script>...</script>`.

Structure-type de la page :

```
<!DOCTYPE html>
<html lang="fr">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>My PictureBox</title>
  <!--ici on placera les link css correspondant aux différents format -->
</head>

<body>
  <div id="wrapper">
    <!-- enveloppe principale -->
    <header>
      <h1><span>La PictureBox de Rudy</span></h1>
    </header>
    <!-- Navigation albums -->
    <nav>
      <ul>
        <li> Portraits</li>
        ...
      </ul>
    </nav>
    <aside>
      <!-- miniatures -->
      <div id="gallery">
        <a href="">
          
        </a>
        <a href="">
          
        </a>
        ...
      </div>
    </aside>
    <main>
      <!-- grande image et reflet -->

      <!-- textes en surimpression sur le reflet -->
      <div id="textesImage">
        <h2 id="titreGrandeImage">Lorem ipsum dolor 01</h2>
        <p id="texteGrandeImage">Lorem ipsum dolor sit amet, consectetur
        adipiscing elit. Modi quisquam sunt voluptatum veritatis deserunt rem
        nesciunt quam aut, illo labore eligendi, dolorum laborum asperiores!
        </p>
      </div>
    </main>
  </div>
<!-- fin enveloppe principale -->
```

Chaque album est supporté par un classique élément

Chaque <a href > contient un élément

On peut aussi ajouter un/des éléments conteneurs pour éviter les attributs id et class des textes et image au profit de sélecteurs CSS. On illustre ici le difficile équilibre entre attributs HTML et sélecteurs CSS...

2/ Style général

1) Pour commencer

On a choisi ici de définir 3 feuilles de styles CSS :

- screen.css pour les affichages communs et par défaut sur écran de PC (largeur mini 781px),
- tablette.css pour les affichages sur petit écran et tablette (largeur entre 551px et 780px)
- smartphone.css pour l'affichage sur mobile (largeur maxi 550px).

Ces 3 feuilles de styles sont liées à la page Web par 3 balises <link > dans l'entête HTML incluant l'attribut media correspondant afin d'optimiser les téléchargements de feuilles de styles par le navigateur. Chacune de ces feuilles de styles englobe toutes les définitions dans la directive @media ... {...} correspondante.

NB : on adopte ici une démarche 'classique' Desktop First mais on pourrait très bien démarrer le développement par la définition des styles pour mobiles.

2) Paramétrage général

On peut encore compléter ce jeu de styles à l'aide d'une dernière balise <link> appelant une feuille CSS de 'normalisation' afin de maîtriser les valeurs par défauts des attributs de style.

On peut aussi se contenter d'initialiser certains attributs CSS comme box-sizing (border-box), font-family, font-size (16px), ainsi que les margin et padding de la page, et, pourquoi pas certains attributs généraux comme le max-width des images (100%, car on ne doit jamais agrandir une image au-delà de ses dimensions nominales).

```
/* normalisation */
*, *:before, *:after {
  box-sizing: border-box;
}
html, body {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 16px;
  height: 100%;
  margin: 0; /* toutes les marges seront définies dans les éléments */
  padding: 0;
  background: #eee;
}
img {
  max-width: 100%; /* taille maxi des images : 100% */
  height: auto; /* zoom, pas de déformation */
}
```

Il reste à définir la règle de style pour le body (ou son conteneur principal, ici, wrapper), en termes de largeur (95%) et de marges (8px haut et bas, centrage horizontal).

```
/* enveloppe principale */
#wrapper {
  /* Width à définir 95% */
  /* Margin à définir 8/16 haut et bas + centrage horizontal */
}
```

3) Titre général

Il doit être centré sur son conteneur et le texte encadré d'une bordure ; utiliser les sélecteurs CSS plutôt que des attributs HTML id :

```
/* titre général */
header h1 {
    /* Alignement du texte centrer */
}
header h1 span{
    /* Bordure à définir */
}
```

4) Navigation

Le bloc de navigation occupe toute la largeur disponible ; on peut lui affecter des couleurs de texte, de fond, une bordure et un ombrage...

```
/* Navigation albums */
nav {
    /* Largeur à définir 100% */
    /* Background à définir */
    /* Marge inférieure de 15/16 */
    /* Bordure à définir 1/16 d'épaisseur */
    /* Ombre à définir 10/16 d'épaisseur*/
}
```

Pour la liste des albums, on ajuste marges internes, externes, centrage du contenu.

```
nav ul {
    /* Margin à définir 5/16 haut et bas + centrage horizontal */
    /* Padding à définir 5/16 sur les 4 cotés */
    /* centrer les li dans la ul */
}
```

Pour les éléments , on centre le texte, on les affiche en blocs continus (display :inline-block), on fixe leur largeur (par défaut on souhaite 3 item par ligne donc une largeur des éléments légèrement inférieure à 33%), leurs marges, leur ombrage... et on ajoute le sélecteur adéquat pour affecter la propriété cursor à la valeur pointer au survol de la souris.

```
nav ul li {
    font-weight: bold;
    line-height: 2em;
    display: inline-block;
    background: #eee;
    /* centrer le texte dans la li */
    /* Largeur à définir 3 de front - par défaut 100% / 3 Attention débordement*/
    /* Margin à définir 3/16 */
    /* Ombre à définir 5/16 d'épaisseur*/
}
/* icone de main au survol de la souris */
nav ul li:hover {
    cursor: pointer;
}
```

On définit un style indépendant des éléments HTML (.active) pour mettre en évidence l'album en cours d'affichage (bordure, style de texte, couleur de fond...) ; cette classe de style est affectée par défaut au premier album et le code événementiel JavaScript pourra se charger ensuite de l'affecter là où il faut.

```
/* album actif */
.active {
  font-style: italic;
  border: dotted 2px black;
}
```

5) Miniatures

Le bloc conteneur devra être cadré sur la gauche (float : left ; width : 33%) ; sa hauteur sera imposée (tout comme celle du bloc principal).

Pourquoi pas exprimer cette hauteur en em car cette unité reste 'absolue' dans une même taille de font-size et que cette taille de font-size pourra varier selon les media queries ?

(Ici, on adopte une hauteur de 720 px, soit 45em, ce qui reste confortable pour afficher l'ensemble de la page Web sur un écran standard orientation paysage).

Le bloc de miniatures pourra recevoir un ascenseur vertical en fonction du nombre de miniatures (overflow-y: auto;)

```
/* miniatures */
#gallery {
  float: left;
  width: 33%;
  height: 45em; /* 720 px soit 720/16 */
  overflow-y: auto; /* ascenseur vertical au besoin */
  background: white;
  /* Ombre à définir 10/16 d'épaisseur */
}
```

Les images miniatures sont contenues dans des éléments <a href> ; il faut donc que ces éléments <a> de la galerie de miniatures soient affichés en blocs continus (display : inline-block ;).

Affiner essentiellement leur largeur (légèrement inférieure à 45% car on en souhaite 2 par défaut) et leur largeur maximale (160px pour des miniatures de 150px), ainsi que leur bordure, marge et ombrage.

```
#gallery a {
  display: inline-block;
  margin: 2.5%; /* de la div gallery */
  max-width: 160px; /* les miniatures font 150px * 150px */
  /* Width à définir inférieur à 45% */
  /* Bordure à définir 2/16 d'épaisseur */
  /* Ombre à définir 5/16 d'épaisseur */
}
```

Il reste peu de choses à définir pour les images elles-mêmes : espacements, couleur de fond...

```
#gallery a img {
  margin: 0;
  padding: 2%; /* du a contenant l'image */
  background: white;
}
```

Un sélecteur supplémentaire est nécessaire pour l'animation au survol de la souris (ici on propose une réduction et une rotation simultanées :

```
#gallery a:hover {  
  transition: all 0.3s ease-in-out;  
  transform: scale(0.9) rotate(-5deg);  
}
```

Attention de ne pas 'en faire trop' : réduction de 10% et rotation de 5° sont bien suffisants pour attirer l'attention ; de même une transition douce ne doit pas dépasser ½ seconde

6) Grande image

Le conteneur (élément <main>) sera flottant à droite, d'une largeur de 65% ; sa hauteur est fixée, comme celle des miniatures, à 720px ou 45em, son contenu centré ; reste à affiner couleurs, marges et ombrage...

```
/* conteneur grande image */  
main {  
  float: right;  
  width: 65%;  
  height: 45em; /* 720 px */  
  padding: 0.625em; /* 10/16 */  
  text-align: center; /* pour centrage photo + légende photo */  
  /* Couleur à définir */  
  /* Background à définir */  
  /* Ombre à définir 10/16 d'épaisseur*/  
}
```

La grande image elle-même peut recevoir un léger ombrage ; sa hauteur maximale sera limitée de manière à tenir dans le conteneur (ici, on choisit 580px, soit 36.25em pour que l'ensemble images + titre + commentaire puisse tenir dans les 720 px).

```
/* grande image */  
#grandeImage {  
  max-height: 36.25em; /* limité à 580px pour tenir dans 720 px */  
  /* Ombre à définir 5/16 d'épaisseur*/  
}
```

Le conteneur des textes de titre et légende est centré et sa largeur est ici fixée à 580px ou 36.25em, soit la hauteur maximale des images restera à vérifier le bon comportement avec des images de tailles et proportions différentes.

```
/* title et alt de l'image répétés en h2 et p*/  
#textesImage {  
  max-width: 36.25em; /* limité à 580px */  
  margin: auto;  
}
```

Le texte du commentaire est ici justifié on peut définir une classe de style pour justifier du texte car cette présentation-type reste très générale et réutilisable ; cela nécessite l'emploi d'un attribut class dans le code HTML ou des sélecteurs CSS complexes.

```
.justified {  
  text-align: justify;  
}
```

A ce stade, tester l'ensemble des affichages dans des tailles d'écran 'normales' de PC, voire larges ou très larges et avec des images diverses ; la mise en page doit rester identique (liste des albums 3 par ligne, miniature sur 2 colonnes ou plus, avec ou sans barre de défilement, grande image affichée réduite, centrée et jamais plus grande que sa taille d'origine ; on respecte donc les principes de base du RWD : mise en page fluide et images fluides, occupant bien l'espace.

Quand tout semble correct, convertir toutes les dimensions absolues restantes (marges...) en dimensions relatives (%) et em) et tester à nouveau.

7) Variante pour très grand écran

En cas d'affichage sur écran très large, la largeur disponible augmente considérablement plus vite que sa hauteur ; on peut donc considérer que notre mise en page reste d'actualité car elle favorise l'affichage de la grande image et de ses textes sans déformation ; les miniatures s'affichent en plus grand nombre ce qui facilite la navigation. Pour profiter au mieux du confort d'affichage sur écran très large, on peut éventuellement ajouter un media query modifiant simplement la taille de base de la police de caractères et même inclure ce media query dans la même feuille de styles screen.css.

3/ Styles tablettes et petits écrans

La feuille de styles tablette.css affine les styles de la feuille screen.css pour des écrans modeste (ici, largeur entre 551px et 780px) ; définir la directive @media ... {...} correspondante. Préciser tout d'abord la taille de base de la police de caractères ; un écart de 1 ou 2 pixels suffit pour obtenir de bons résultats.

```
@media screen and (min-width: 551px) and (max-width: 780px ) {  
    html, body {  
        font-size : 15px;  
    }  
}
```

1) Navigation

Pour ne pas occuper trop de place avec la liste des albums, fixer la hauteur maximale (90px par exemple) et ajuster les options de défilement :

```
/* écran petite largeur et tablette */  
nav { /* au besoin passer en scroll sur albums */  
    max-height : 90px;  
    overflow-x : auto;  
    overflow-y: hidden;  
    white-space: nowrap;  
}
```

2) Corps de page (main et gallery)

Modifier légèrement (on souhaite toujours rester 'fluide' !) les proportions de largeur entre miniatures et grande image (27% - 70% par exemple) et réduire légèrement leur hauteur (43em par exemple - noter que la taille du em vient de diminuer).

```
main { /* privilégier la grande image */  
    width: 70%;  
    height: 43em;  
}
```

```
#gallery {  
    width: 27%;  
    height: 43em;  
}
```

Et cela devrait suffire ! L'affichage sur petit écran mixe les 2 feuilles de styles screen.css et tablette.css ; toutes les règles de styles incluses dans tablette.css ont un poids plus fort que celles de screen.css car le media query correspondant est plus précis.

4/ Styles smartphone

La feuille de styles smartphone.css affine les styles de la feuille screen.css pour des écrans mobiles (ici, largeur inférieure à 551px) ; définir la directive @media ... {...} correspondante.

Taille de la police de caractères, un écart de 1 à 2 pixels suffit encore pour obtenir de bons résultats.

```
@media screen and (max-width: 550px) {  
    html, body {  
        font-size : 13px;  
    }  
}
```

1) Navigation

Adapter là règle précédente pour défilement horizontal systématique ; pour gagner de la place, réduire l'affichage de chaque album :

```
nav { /* scroll sur albums */  
    overflow-x: auto;  
    overflow-y: hidden;  
    white-space: nowrap;  
}  
/* réduction de la taille des albums */  
nav ul li {  
    font-size: .9em;  
    line-height: 2em;  
    width: 7.1875em; /* 115/16 */  
}
```

2) Corps de page (main et gallery)

Miniatures et grande image sont affichées l'un en-dessous de l'autre : annuler les flottements (float : none ;), largeurs à 100% et hauteurs en auto, le défilement doit maintenant s'effectuer en horizontal.

```
#gallery {  
    float: none; /* précède main */  
    width: 100%;  
    height: auto; /* plus de hauteur imposée */  
    overflow-x: scroll; /* défilement horizontal */  
    overflow-y: hidden;  
    white-space: nowrap;  
    margin-bottom: 0.9375em; /* idem header */  
}
```

L'affichage <a href>, on réduit à 25% la place utile avec une marge de 2%. CSS fusionne les marges verticales uniquement ; il est donc nécessaire de fusionner 'à la main' les marges horizontales :

```
#gallery a {  
    float: none;  
    width: 25%;  
    margin: 2%; /* marge identique tout autour... */  
}  
#gallery a:nth-child(n+2) { /* pas de fusion auto des marges en horizontal */  
    margin-left: 0;  
}
```

III Extension JavaScript

1/ Afficher dynamiquement la grande image

La fonction JavaScript à écrire doit récupérer les contenus des attributs alt et title de l'image sélectionnée et les injecter dans les éléments HTML de titre et commentaire de l'image ; il faut de plus 'calculer' l'URL de la grande image et l'appliquer à l'élément .

Cette fonction est appelée au clic de chaque image miniature ; elle peut donc passer en paramètre au moins sa référence (this) ; si on veut la rendre réutilisable dans un autre contexte, il est encore possible de passer les références des éléments HTML recevant grande image, titre et commentaire, soit 3 paramètres de plus) ; ici, on se contentera de passer la référence de l'image miniature (), et la fonction JavaScript devra donc 'explorer' le DOM pour accéder aux éléments HTML :

```
<script>
  function affiche(elementImg) {
    var textalt = elementImg.alt;
    var texttitle = elementImg.title;
    var src = elementImg.src;

    var srcBigImg = src.substring(src.lastIndexOf('/') + 1);

    document.getElementById("grandeImage").src = "img/" + srcBigImg;
    document.getElementById("titreGrandeImage").innerHTML = texttitle;
    document.getElementById("texteGrandeImage").innerHTML = textalt;
  }
</script>
```

```
<a href="" onclick="return false;">
  
</a>
```

L'idée ici n'est pas de vraiment comprendre le fonctionnement de la fonction JS Vous n'avez qu'à juste l'intégrer dans votre code HTML avant la fermeture de la balise body. Cependant sans comprendre le code on remarque que getElementById cible un élément HTML dont l'id est ce qui est saisie dans les parenthèses entre guillemet. Vous aurez peut-être besoin d'adapter à votre code.

Tester.

2/ Modifier l'apparence de l'album sélectionné

Chaque élément cliqué devra appeler une fonction JavaScript et lui passer sa référence :

```
<li class="" onclick="active(this);">
```

Cette fonction JavaScript va essentiellement affecter l'attribut HTML class des différents items de la liste des albums ; on choisit ici de supprimer tout d'abord les classes affectées aux éléments puis de définir la classe 'active' pour le seul élément choisi par l'utilisateur.

Bien entendu, il faudrait aussi réinjecter une liste de miniatures dans le bloc gallery et actualiser le bloc principal pour y afficher la première grande image ; tout cela pourrait se faire aisément grâce à une transaction Ajax mais ce n'est pas l'objet de cet exercice et on se limitera ici à affecter les styles CSS des éléments .

Pour l'exemple, on utilisera ici du JavaScript standard et on explorera le DOM 'façon CSS' grâce aux nouvelles méthodes JavaScript :

A insérer dans la balise script

```
function active(elementLi) {  
    var lis = document.querySelectorAll("nav ul li");  
    for (let i = 0; i < lis.length; i++) {  
        lis.item(i).className = '';  
    }  
    elementLi.className = "active";  
}
```

Tester. C'est fini !