





Rapport devoir(mohamed said boufanzi)

Introduction :

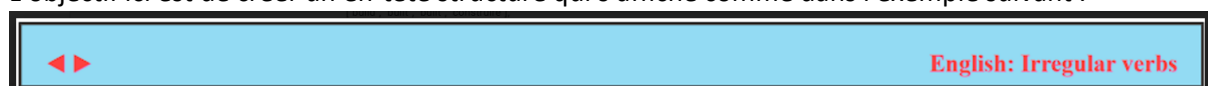
Dans ce rapport, je vais vous emmener a travers les différentes étapes de la création de devoir, de A à Z. La première étape consiste à concevoir le devoir. Cela implique de réfléchir à comment le réaliser et quels outils ou ressources on vas utiliser , puis La création des fichiers qu'en vas utiliser :

	index	25/11/2024 11:13	Chrome HTML Do...	1 KB
	Rapport	25/11/2024 11:12	Microsoft Word D...	12 KB
	scripts	25/11/2024 12:06	JavaScript Source ...	0 KB
	styles	25/11/2024 11:13	CSS Source File	0 KB

nous allons diviser notre projet en trois sections principales :header , main , second. Nous allons créer le contenu de chaque section étape par étape.

1-Conception du Header :

L'objectif ici est de créer un en-tête structuré qui s'affiche comme dans l'exemple suivant :



Donc on a un header dans un tableaux le header il est a droit , on va essayer de traduit te qui ce que en vois au code html et les couleur et les position on css .

```
<table class="head">
<tr>
<td>
<h1 id="Irregular">English:Irregular verbs </h1>
</td>
</tr>
</table>

#Irregular{
color: red;
text-align: right;
}

.head{
width:100%;
height:90px;
border:solid;
border-width:4px;
border-color:black;
background-color: #93dbf3;
margin-bottom:2px;
border-collapse: collapse;
}
```

Ona obtenu sa :



Maintenant, nous allons travailler sur les flèches en haut. Le rôle de ces flèches est d'interchanger entre les deux volets gauche et droit (interchanger les largeurs). Nous avons donc besoin de :

Une image rouge : une image de flèche qui sera notre flèche droite. ► Pour la flèche gauche, nous utiliserons la même image, mais avec une rotation. On va nommer class leftimage

```
.left_image{
  transform: rotate(180deg);
  cursor:pointer
}
```

Et on a besoin de fonction javascript qui va changer les largeurs de deux volet on va les nommer swap.

```
<td>
  <div class="buttons1">
    
    
  </div>
</td>
```

on va nommer notre volet pour l'utiliser dans notre fonctions swap (volet 1 :left_panel for the table , volet2 :right_panel) on va les traiter dans main and second part .Nous utiliserons les transitions CSS pour rendre l'échange fluide ainsi que la propriété CSS width pour modifier la largeur.

```
//exercice a : im trying to create the swaps fonctions at the header as soon you click on the flech it change the width of the two pages (interchanger)
function swopleft() {
  const table = document.getElementById('left_panel');
  const panel = document.getElementById('right_panel');

  table.style.transition = 'width 1s ease';
  panel.style.transition = 'width 1s ease';

  table.style.width = '60%';
  panel.style.width = '37%';
}

function swapright() {
  const table = document.getElementById('left_panel');
  const panel = document.getElementById('right_panel');

  table.style.transition = 'width 1s ease';
  panel.style.transition = 'width 1s ease';

  table.style.width = '37%';
  panel.style.width = '60%';
}
```

Conclusion :



2-Conception du main : (tableau-leftpanel) dans cette partie on va créer notre tableau et notre Button édit – update Delete.

2-1 on va créer l'espace et border où on va lancer notre donner :

```

<div id="left_panel" class="left_panel">
  <table border="2" height="100px" width="100px" id="data_table"
class="words_table">
    </table>
  </div>

```

Et on va créer notre tableau en utilisant javascript, j'ai utilisé le même code que vous avez donné et on a stylé notre tableau en utilisant CSS. Les headers et les couleurs des tableaux on remarque que les couleurs des tableaux et comme un zébra on utilise nth-child.

```

// affiche table de verbs
function displayNames() {
  var table = document.getElementById("data_table");
  var table_content = `
  <tr>
    <th>${verbs[0][0]}</th>
    <th>${verbs[0][1]}</th>
    <th>${verbs[0][2]}</th>
    <th>${verbs[0][3]}</th>
    <th>Actions</th>
  </tr>`;

  verbs.slice(1).forEach((verb, index) => {
    table_content += `
    <tr>
      <td>${verb[0]}</td>
      <td>${verb[1]}</td>
      <td>${verb[2]}</td>
      <td>${verb[3]}</td>
      <td>
        <button onClick='viewVerbDetails(${index + 1})'>Edit</button>
        <button onClick='prepareUpdateVerb(${index + 1})'>Update</button>
        <button onClick='confirmDelete(${index + 1})'>Delete</button>
      </td>
    </tr>`;
  });

  table.innerHTML = table_content;
}

```

```

.left_panel{
  overflow-y:scroll;
  height:570px;
  width:64%;
  border:solid;
  border-width:4px;
  border-color: black;
  float:left;
  display:block;
}

.words_table th {
  background-color: #7fffd4;
}

.words_table tr:nth-child(2n+1) {
  background-color: blanchedalmond;
}

.words_table tr button {
  margin-left:5px;
  margin-right:5px;
  width:60px;
}

.words_table tr th{
  text-align:left;
}

```

Et on va créer les fonctions javascripts/ pour edit -delete -update.

Delete on va utiliser prompt et alerte pour communiquer avec l'utilisateur :

fonction qui confirme avec l'utilisateur et après confirmation il fait la suppression (parfois en click sur des boutons qui sont un peu risquer donc l'effet de confirmer avec l'utilisateur est vraiment important).

```
// supprimer verb
function confirmDelete(index) {
  if (index < 1 || index >= verbs.length) {
    alert("invalid");
    return;
  }

  var x = prompt("delete the verb? (y/n)");
  if (x === 'y') {

    const suprimlettre = verbs[index][0].charAt(0).toLowerCase();

    // supp
    verbs.splice(index, 1);

    alert("Verb deleted");
  } else if (x === 'n') {
    alert("canceled");
  } else {
    alert(" Please type 'y' or 'n'.");
  }
}
}
```

Pour édit : on va afficher une boite dialogue qui vas être au milieu

```
// model dialog for edite verb details
function editmodel(verb) {
  //model container
  const modal = document.createElement('div');
  modal.id = 'view-modal';
  modal.style.cssText = `
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0,0,0,0.1);
    z-index: 1000;
    width: 500px;
  `;
}
```

Et qui vas afficher les donner qui sont stocker dans le tableau pour chaque verbe avec une Button pour fermer cette boite de dialogue .

```
// Model content
model.innerHTML = `
  <h2>Verb Details</h2>
  <div>
    <p><strong>Base form:</strong> ${verb[0]}</p>
    <p><strong>Past tense:</strong> ${verb[1]}</p>
    <p><strong>Past participle:</strong> ${verb[2]}</p>
    <p><strong>Translation:</strong> ${verb[3]}</p>
    <p><strong>Synonyms:</strong> ${verb[4]} || 'N/A'</p>
    <p><strong>Example sentence:</strong> ${verb[5]} || 'N/A'</p>
    <p><strong>Related images:</strong> ${verb[6]} || 'N/A'</p>
  </div>
  <div style="margin-top: 15px;">
    <button onclick="closeViewmodel()">Close</button>
  </div>
`;
}
```

Pour update : même chose une boite ou en communique avec client mais ici il va donner des information et on vas l modifier dans le tableaux , donc on va utiliser plusieurs fonction js + deux Button cette fois une pour annuler et une autre pour confirmer :

```
// Create a model dialog for updating verb details
function updatemodel(verb) {
  // Create a model container
  const model = document.createElement('div');
  model.id = 'update-model';
  model.style.cssText = `
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0,0,0,0.1);
    z-index: 1000;
    width: 500px;
  `;
}
```

Et le contenu de cette boite et input pour que les utilisateurs entre les information .

```
function updatemodel(verb) {
  // model content
  model.innerHTML = `
    <h2>Update Verb Details</h2>
    <form id="update-verb-form">
      <div>
        <label>Base Form:</label>
        <input type="text" name="baseForm" value="${verb[0]}" required>
      </div>
      <div>
        <label>Past Tense:</label>
        <input type="text" name="pastTense" value="${verb[1]}" required>
      </div>
      <div>
        <label>Past Participle:</label>
        <input type="text" name="pastParticiple" value="${verb[2]}" required>
      </div>
      <div>
        <label>Translation:</label>
        <input type="text" name="translation" value="${verb[3]}" required>
      </div>
      <div>
        <label>Synonyms:</label>
        <input type="text" name="synonyms" value="${verb[4]} || ''">
      </div>
      <div>
        <label>Example Sentence:</label>
        <textarea name="example">${verb[5]} || ''</textarea>
      </div>
      <div>
        <label>Related Images (URL):</label>
        <input type="text" name="images" value="${verb[6]} || ''">
      </div>
      <div style="margin-top: 15px;">
        <button type="button" onclick="saveverbupdate()">Save Update</button>
        <button type="button" onclick="closeupdatemodel()">Cancel</button>
      </div>
    </form>
  `;
}
```

```
// ajouter overlay
const overlay = document.createElement('div');
overlay.id = 'model-over';
overlay.style.cssText = `
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0,0,0,0.5);
  z-index: 999;
`;

document.body.appendChild(overlay);
document.body.appendChild(model);
```

Un fonction prépare update (saveverbupdate()) pour confirmer et sauvegarder les donnes . et lorsque on update le verbe en reçois un alerte pour verbe update. Et une autre (closeup model()) pour cancel (removechilde.)

```
function closeupdatemodel() {
  const model = document.getElementById('update-model');
  const overlay = document.getElementById('model-over');

  if (model) document.body.removeChild(model);
  if (overlay) document.body.removeChild(overlay);
}
```

```
// updating
function saveverbupdate() {
  // Ensure we have a valid index
  if (currentVerbIndex === null) {
    alert("invalid");
    return;
  }

  // Get form values
  const form = document.getElementById('update-verb-form');

  // Update the verb in the array with new values
  verbs[currentVerbIndex] = [
    form.baseForm.value,
    form.pastTense.value,
    form.pastParticiple.value,
    form.translation.value,
    form.synonyms.value,
    form.example.value,
    form.images.value
  ];

  // fermer model et refresher
  closeupdatemodel();
  displayNames();
  alert("verb updated ");

  // on refresh l index
  currentVerbIndex = null;
}
```

a cette partie on aura notre volet 1 parfait comme l'image on obtenir sa :

Base form	Past tense	Past participle	Translation	Actions
abide	abode	abode	demeurer	<button>Edit</button> <button>Update</button> <button>Delete</button>
awake	awoke	awoken	(se) réveiller, aussi awake/awoke/awoke	<button>Edit</button> <button>Update</button> <button>Delete</button>
be	was/were	been	être	
bear	bore	borne	porter/supporter/soutenir	
beat	beat	beaten	battre	
become	became	become	become	
beget	begat	begotten	engendrer, aussi beget begot/begotten	
begin	began	begun	commencer	
bend	bent	bent	se courber, etc.	
bereave	bereft	bereft	déposséder/priver	
bring	brought	brought	apporter	

Update Verb Details

Base Form:

Past Tense:

Past Participle:

Translation:

Synonyms:

Example Sentence:

Related Images (URL):

Save Update Cancel

3-Conception du second :

(rightpanel) on vas la diviser on 4 , tout section on va lui donner class et id pour modifier leur css et on vas crée dans chaque une, une fonction correspond a notre exercice.

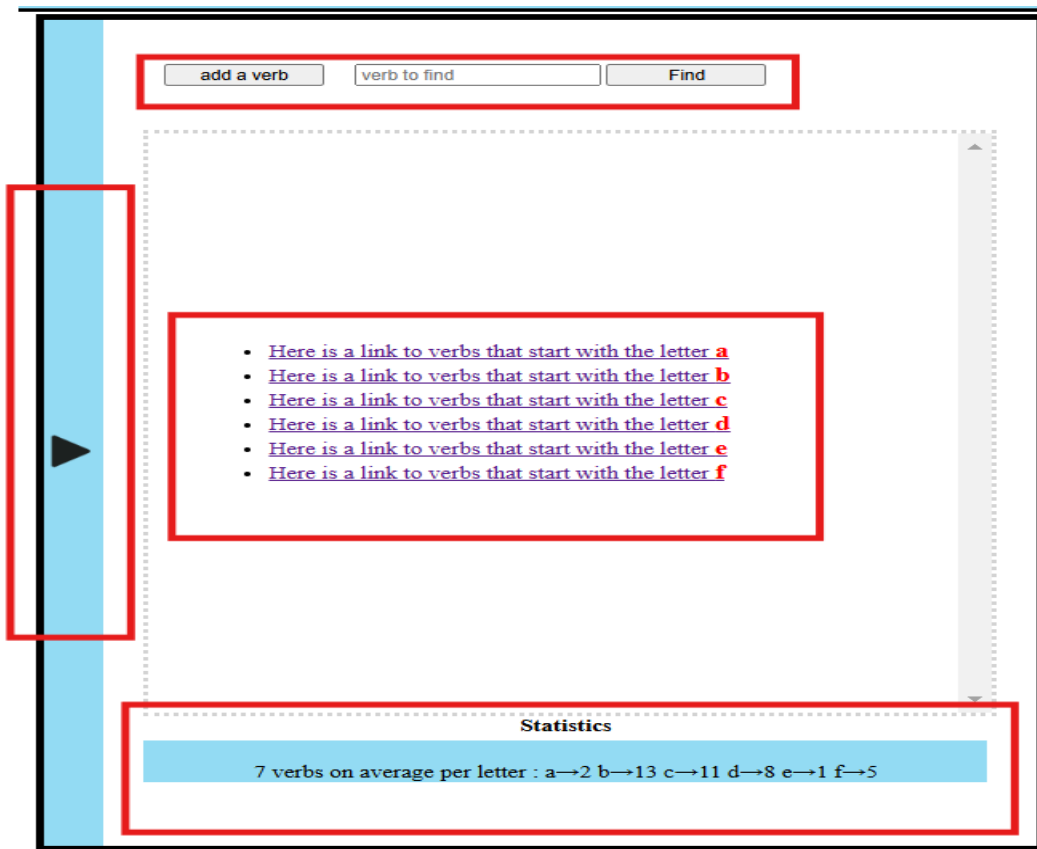
(blue_bar) ou La flèche noire qui affiche et cash la volet droit (right panel) .

Pour cest trois il sont dons la même section mais une top une midle et une en bas.

(right bar top) Contient add verb et find .

(Link_div) ou on trouve les liens.

(Right_bar_bottom) Contient les statistiques.



On traduit tous ce qu'on a dit en un code html :

```
<div id="right_panel" class="right_panel">
  <div class="blue_bar">
    
    
  </div>
  <div id="right_bar_main_bar" class="right_bar_main_bar">
    <div class="right_bar_top">
      <button onClick="addVerb()">add a verb</button>
      <input type="text" placeholder="verb to find" name="input" id="search_input" />
      <button onClick="search()">Find</button>
    </div>
    <div id="links_div" class="links_div">
      <ul onLoad="display_letters()" id="link_list" class="link_list">
      </ul>
    </div>
    <div class="right_bar_bottom">
      <p>Statistics</p>
      <div onClick="getListOfLetter()" id="statistic">
      </div>
    </div>
  </div>
</div>
```

On va traiter chaque section seule :

Pour blue_bar :

Css :

```
.blue_bar {
  float:left;
  height:100%;
  width:30px;
  background-color: #93dbf3;
  position:relative;
}
```

Pour les fonctions js :

On va utiliser deux fonction un pour masquer notre volet gauche et une autre pour l'afficher, notre boutons sont des images flesh , iconb.png et on va les nommer .

```
.blue_bar_image{
  height:25px;
  width:25px;
  float:middle;
  position:relative;
  top:50%;
  cursor:pointer;
}

.right_image{
  cursor:pointer;
}

.hide_blue_bar_image{
  transform:rotate(180deg);
  position:relative;
  top:50%;
  height:25px;
  width:25px;
  display:none;
  cursor:pointer;
}
```

```
//exercice c;
function hide() {
  const wordTableWrapper = document.getElementById('left_panel');
  const rightPanel = document.getElementById('right_panel');
  const rightBarMainBar = document.getElementById('right_bar_main_bar');
  const blueBarImage = document.getElementById('blue_bar_image');
  const hideBlueBarImage = document.getElementById('hide_blue_bar_image');

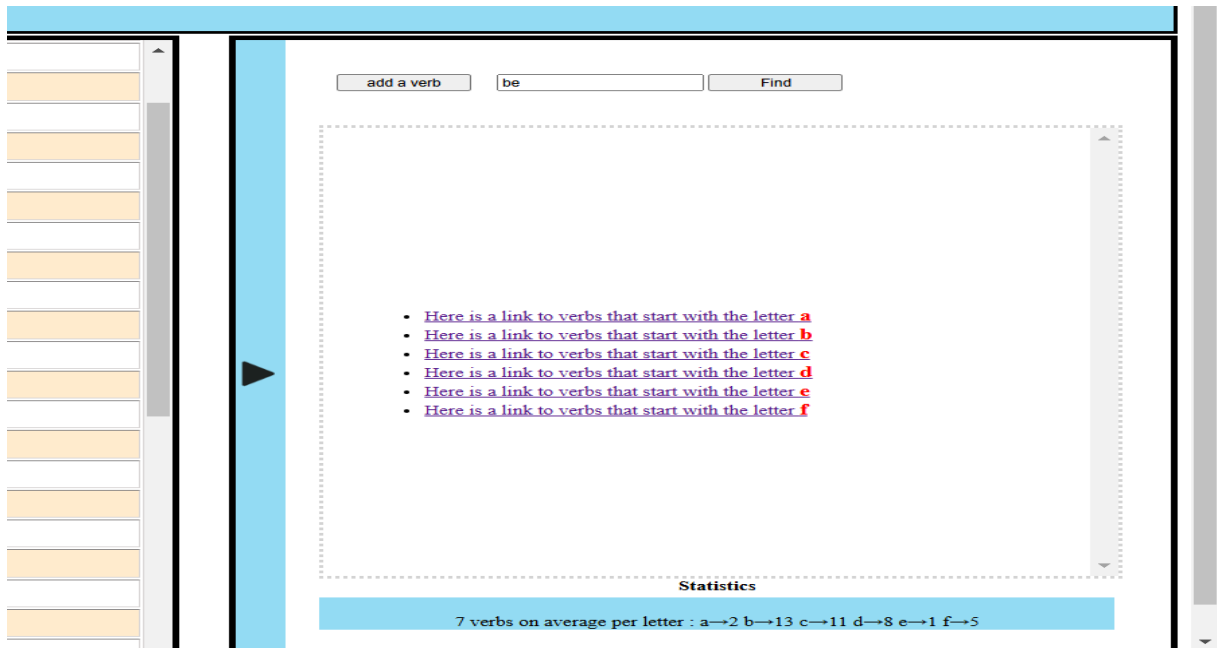
  //Ajouter des transitions
  wordTableWrapper.style.transition = 'width 0.3s ease';
  rightPanel.style.transition = 'width 0.3s ease';

  // Agrandit le panneau de gauche et réduit le panneau de droite
  wordTableWrapper.style.width = '95%';
  rightPanel.style.width = '3%';

  // Masquer le contenu principal du panneau de droite mais garder la barre bleue visible
  rightBarMainBar.style.display = 'none';

  // Changer la visibilité des boutons bascule
  blueBarImage.style.display = 'none';
  hideBlueBarImage.style.display = 'block';
}
```

Cette fonction il va donner comme resultat



```
function show() {
  const wordTableWrapper = document.getElementById('left_panel');
  const rightPanel = document.getElementById('right_panel');
  const rightBarMainBar = document.getElementById('right_bar_main_bar');
  const blueBarImage = document.getElementById('blue_bar_image');
  const hideBlueBarImage = document.getElementById('hide_blue_bar_image');

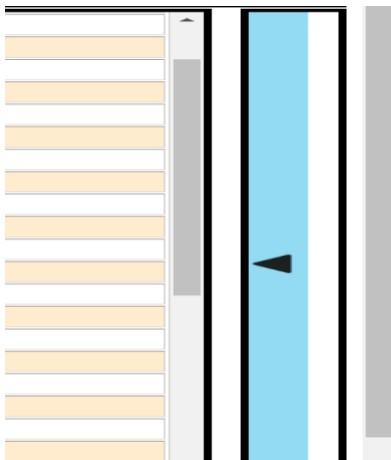
  //Ajouter des transitions
  wordTableWrapper.style.transition = 'width 0.3s ease';
  rightPanel.style.transition = 'width 0.3s ease';

  // largeurs d'origine
  wordTableWrapper.style.width = '60%';
  rightPanel.style.width = '37%';

  // Afficher le contenu principal du panneau de droite
  rightBarMainBar.style.display = 'block';

  // Changer la visibilité des boutons bascule
  blueBarImage.style.display = 'block';
  hideBlueBarImage.style.display = 'none';
}
```

Cette fonction il vas donner comme resultat .



Pour `right_bar_top` : il va contenir deux main fonction,

adverb()+ un model ou en vas ajouter les donnees , et annuler

```
// Function pour ajouter un verb
function addVerb() {
  // creation d'un model
  const model = document.createElement('div');
  model.style.cssText = `
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background: white;
    padding: 20px;
    border: 2px solid black;
    z-index: 1000;
  `;

  model.innerHTML = `
    <h3>Add New Verb</h3>
    <div>
      <label>Base Form: </label>
      <input type="text" id="newBaseForm" required>
    </div>
    <div>
      <label>Past Tense: </label>
      <input type="text" id="newPastTense">
    </div>
    <div>
      <label>Past Participle: </label>
      <input type="text" id="newPastParticiple">
    </div>
    <div>
      <label>Translation: </label>
      <input type="text" id="newTranslation">
    </div>
    <button onclick="submitNewVerb()">Add</button>
    <button onclick="closeAddmodel()">Cancel</button>
  `;
}
```

```
// applique le nouveau verb
function submitNewVerb() {
  const baseForm = document.getElementById('newBaseForm').value;
  const pastTense = document.getElementById('newPastTense').value;
  const pastParticiple = document.getElementById('newPastParticiple').value;
  const translation = document.getElementById('newTranslation').value;

  // Valider
  if (!baseForm) {
    alert('Base form is required');
    return;
  }

  // Creation de un nouveau tableau
  const newVerb = [baseForm, pastTense, pastParticiple, translation];

  //alphabetique ordre
  let insertIndex = 1;
  for (let i = 1; i < verbs.length; i++) {
    if (baseForm.toLowerCase() < verbs[i][0].toLowerCase()) {
      insertIndex = i;
      break;
    }
    insertIndex = i + 1;
  }

  //lajouter
  verbs.splice(insertIndex, 0, newVerb);

  // display
  displayNames();
  display_letters();
}
```

```
// les case incomplete on le donne un rouge border
const row = document.querySelector('#data_table tr:nth-child(${insertIndex + 1})');
if (!pastTense || !pastParticiple || !translation) {
  row.style.border = '2px solid red';
}

closeAddmodel();
}
```

```
// Function pour fermer le model
function closeAddmodel() {
  const model = document.querySelector('div[style*="position: fixed"]');
  if (model) {
    document.body.removeChild(model);
  }
}
```

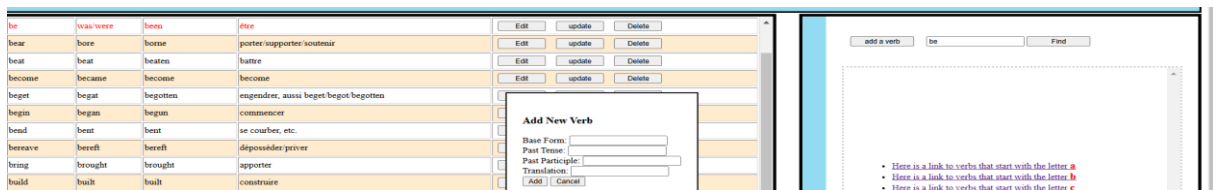
et search() cette fonction a rôle de trouver notre verbe et de l'afficher en haut en rouge.

```
// Function pour chercher sur notre verb
function search() {
  const searchInput = document.getElementById('search_input').value.toLowerCase();
  const table = document.getElementById('data_table');
  const rows = table.getElementsByTagName('tr');

  // Reset
  for (let i = 1; i < rows.length; i++) {
    rows[i].style.color = '';
  }

  // trouver le verb
  for (let i = 1; i < rows.length; i++) {
    const verb = rows[i].cells[0].textContent.toLowerCase();
    if (verb === searchInput) {
      // colorer le ligne de verbe en rouge et le scroller en haut .
      rows[i].style.color = 'red';
      rows[i].scrollIntoView({ behavior: 'smooth' });
      break;
    }
  }
}
```

Donc cette section il va donner comme résultat sa :



Pour les liens : on a un lien qui vous Transfer a un emplacement ou notre verbe est positionner et List de lettre qui en a colorer avec le rouge et on augmenter le font size avec css donc on a besoin de deux fonction

```
.link_list {
  margin: 35% 0 0 20px;
}

.link_list li a span {
  color: red;
  font-weight: bold;
  font-size: large;
}
```

```
function display_letters() {
  const linkList = document.getElementById('link_list');
  const letters = new Set();

  // les premier lettre de chaque verbe
  verbs.slice(1).forEach(verb => {
    letters.add(verb[0].charAt(0).toLowerCase());
  });

  // organiser les letters alphabetically
  const sortedLetters = Array.from(letters).sort();

  // creations des liens
  let linkContent = '';
  sortedLetters.forEach(letter => {
    linkContent += `
    <li>
      <a href="#" onclick="scrollToLetter('${letter}')">
        Here is a link to verbs that start with the letter <span>${letter}</span>
      </a>
    </li>`;
  });

  linkList.innerHTML = linkContent;
}
```

Donc cette fonction (display letters) crée notre lien et l'organise d'après l'ordre alphabétique qui collecter grâce a le premier lettre de verbe.

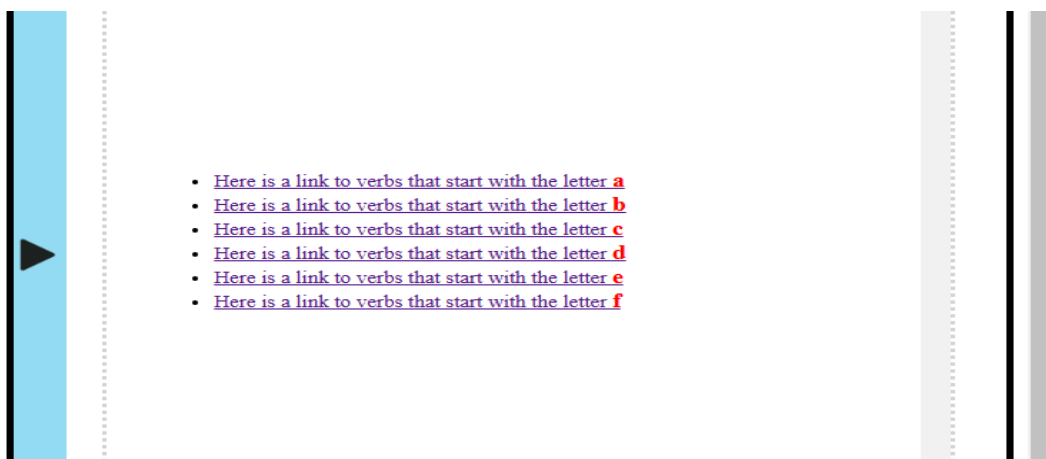
```
// fonction pour que on deplace a chaque premier verb qui a cette lettre
function scrollToLetter(letter) {
  const table = document.getElementById('data_table');
  const rows = table.getElementsByTagName('tr');

  // inicialiser
  for (let i = 1; i < rows.length; i++) {
    rows[i].style.color = '';
  }

  // trouver le premier verbe que se trouve avec cette lettre
  for (let i = 1; i < rows.length; i++) {
    const firstLetter = rows[i].cells[0].textContent.charAt(0).toLowerCase();
    if (firstLetter === letter.toLowerCase()) {
      // Highlight the row
      rows[i].style.color = 'red';
      // Scroll to the row
      rows[i].scrollIntoView({ behavior: 'smooth' });
      break;
    }
  }
}
```

Cette fonction scroller letter il cherche sur le premier verbe avec cette lettre et il se déplace a a cette lettre et le colore ne rouge,

Donc cette section il va donner comme résultat sa :



Pour la section right bar Bottom : on vas utiliser une fonction js pour conter le nombre des verbe avec un lettre on vas l'appeler update statistiques + on vas customiser notre section bleu background changer font weight etc.

```
.right_bar_bottom{
  display:block;
  width:100%;
  margin: 0px 0 0 20px;
}

.right_bar_bottom p{
  text-align: center;
  width:100%;
  height:5px;
  margin-top:0px;
  font-weight:bold;
  display:block;
}

.right_bar_bottom div {
  width:100%;
  background-color: #93dbf3;
  padding-top:15px;
  text-align:center;
}
```

```
// Fonction pour modify statistics
function updateStatistics() {
  const statisticsDiv = document.getElementById('statisc');
  const letterCounts = {};

  // Comptez les verbes pour chaque lettre
  verbs.slice(1).forEach(verb => {
    const firstLetter = verb[0].charAt(0).toLowerCase();
    letterCounts[firstLetter] = (letterCounts[firstLetter] || 0) + 1;
  });

  // Calculer la moyenn
  const totalVerbs = Object.values(letterCounts).reduce((a, b) => a + b, 0);
  const uniqueLetters = Object.keys(letterCounts).length;
  const average = uniqueLetters ? Math.round(totalVerbs / uniqueLetters) : 0;

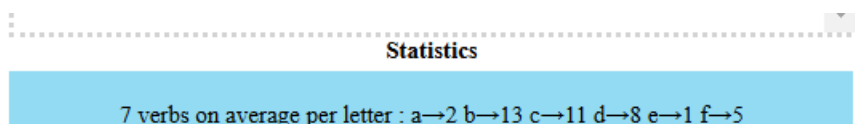
  //Cr  er une cha  ne de statistiques
  let statsText = `${average} verbs on average per letter : `;

  // Ajouter un nombre de lettres individuel
  statsText += Object.entries(letterCounts)
    .sort((a, b) => a[0].localeCompare(b[0])) // Sort alphabetically
    .map(([letter, count]) => `${letter}→${count}`)
    .join(' ');

  statisticsDiv.textContent = statsText;
}
```

Cette fonction il compte les verbe de chaque lettre puis il calcule le moyenne et en fin il cr  e les chaine de statistique.

Donc cette section il va donner comme r  sultat sa :



Statistics

7 verbs on average per letter : a→2 b→13 c→11 d→8 e→1 f→5

Remarque :

On doit l'appel a chaque cr  ation d'un nouveau lien (nouveau verb)

```
// Update statistics apres updating les link |
updateStatistics();
```

Les trois dernier section on été dans un div

```
<div id="right_bar_main_bar" class="right_bar_main_bar">
```

On va la styler avec css

```
.right_bar_main_bar{  
  height:90%;  
  width:85%;  
  margin-left:30px;  
  display:block;ssc  
}
```

Conclusion :

En somme, ce projet illustre de manière détaillée les différentes étapes de la création d'une interface web interactive et bien structurée, en utilisant HTML, CSS, et JavaScript.

Autre détails et ressource :

Extension : findcolor , j'ai l'utiliser pour détecter les couleurs exact dans les images.

Jai commenter mon code en langue français .

Tout autre contenu est en anglais .