

Fondamentaux du Web

HTML, CSS, JavaScript

Lab 9 — Les modes de jeu

Drapeaux, Langues, Voisins

Licence 1 DUT1-Informatique — 2025–2026

Informations pratiques

Où sommes-nous ? Jàng Afrig charge les vrais pays et propose un mode Capitales. Mais nos données contiennent bien plus : drapeaux, langues, voisins ! Aujourd'hui, on transforme le quiz en un jeu à **quatre modes**, sans réécrire le moteur — juste en variant les questions.

Avant de commencer :

- Le Lab 8 fonctionne (tag v0.8) : les pays se chargent depuis l'API.
- Vous avez dans chaque pays : **nom**, **capitale**, **drapeau**, **langues**, **voisins**.

Validation de fin : les quatre modes (Capitales, Drapeaux, Langues, Voisins) fonctionnent, avec une barre de progression. Tag v0.9 poussé + rapport.

Objectifs du Lab 9

À la fin, vous saurez :

1. Proposer plusieurs modes de jeu depuis l'écran d'accueil.
2. **Paramétrer** la génération d'une question selon le mode choisi.
3. Afficher une **image** (le drapeau) dans une question.
4. Construire des questions sur les langues et le nombre de voisins.
5. Ajouter une **barre de progression** visuelle.

Table des matières

Rappel express du Lab 8	2
1 Choisir un mode depuis l'accueil	2
2 Une question selon le mode	3
2.1 Mode Capitales (rappel) et Drapeaux	3
2.2 Mode Langues	4
2.3 Mode Voisins (réponse numérique)	4
3 Afficher l'image quand il faut	5

4	La barre de progression (CSS)	6
5	Exercices d'application	6
6	Aide-mémoire du Lab 9	7
7	Sauvegarder mon travail (routine Git)	7
8	Résumé et transition	7

Rappel express du Lab 8

Chaque pays est maintenant un objet riche : `{ cca3, nom, capitale, drapeau, langues, voisins }`. On va exploiter ces nouveaux champs.

1 Choisir un mode depuis l'accueil

On remplace l'unique bouton « Mode Capitales » par **quatre boutons**. Chaque bouton porte un attribut `data-mode` qui dit lequel c'est. Dans l'écran `#accueil` de `index.html` :

```
<section id="accueil" class="cachee">
  <p>Choisis ton mode de jeu :</p>
  <label for="pseudo">Ton pseudo :</label>
  <input type="text" id="pseudo" placeholder="Ex : Aminata">

  <div id="modes">
    <button class="mode" data-mode="capitale">Capitales</button>
    <button class="mode" data-mode="drapeau">Drapeaux</button>
    <button class="mode" data-mode="langue">Langues</button>
    <button class="mode" data-mode="voisin">Voisins</button>
  </div>
</section>
```

Dans `script.js`, on les écoute tous, et on lit le mode dans `event.target.dataset.mode` :

```
let modeActuel = "capitale";
const boutonsMode = document.querySelectorAll(".mode");

boutonsMode.forEach((bouton) => {
  bouton.addEventListener("click", (event) => {
    const mode = event.target.dataset.mode; // lit l'attribut data-mode
    demarrerQuiz(mode);
  });
});
```

Une nouveauté à prendre telle quelle : `data-mode`

Un attribut HTML qui commence par `data-` sert à **ranger une info** sur un élément. On le relit en JavaScript avec `.dataset` : `data-mode` devient `element.dataset.mode`. C'est la façon propre d'attacher « ce bouton lance le mode drapeau » directement dans le HTML.

On adapte `demarrerQuiz` pour qu'il **reçoive le mode** :

```
function demarrerQuiz(mode) {
  modeActuel = mode;
  pseudo = champPseudo.value;
  if (pseudo === "") { pseudo = "Joueur"; }
  score = 0;
  indexQuestion = 0;
  questions = genererQuestions(10, mode);
  afficherEcran(ecranQuiz);
  afficherQuestion();
}

function genererQuestions(nombre, mode) {
  const liste = [];
  for (let i = 0; i < nombre; i++) {
    liste.push(genererQuestion(mode));
  }
  return liste;
}
```

2 Une question selon le mode

Le cœur du Lab : une seule fonction `genererQuestion(mode)` qui fabrique la bonne question. D'abord, un **petit outil** qui construit quatre propositions de texte (la bonne + trois distracteurs pris dans un champ donné) :

```
// 4 propositions : la bonne réponse + 3 autres prises dans `champ`
function quatrePropositions(bonneReponse, champ) {
  const props = [bonneReponse];
  while (props.length < 4) {
    const autre = choisirAuHasard(pays);
    const valeur = autre[champ]; // ex. autre.capitale ou autre.nom
    if (valeur && !props.includes(valeur)) {
      props.push(valeur);
    }
  }
  return melanger(props);
}
```

`autre[champ]` utilise l'accès par crochets (Lab 6) : si `champ` vaut "nom", alors `autre[champ]` est `autre.nom`. Cela rend l'outil réutilisable pour tous les modes texte.

2.1 Mode Capitales (rappel) et Drapeaux

```
function genererQuestion(mode) {
  const bonPays = choisirAuHasard(pays);

  if (mode === "capitale") {
    return {
      enonce: `Quelle est la capitale du pays : ${bonPays.nom} ?`,
      bonneReponse: bonPays.capitale,
      propositions: quatrePropositions(bonPays.capitale, "capitale"),
      image: null
    };
  }

  if (mode === "drapeau") {
```

```

return {
  enonce: "À quel pays appartient ce drapeau ?",
  bonneReponse: bonPays.nom,
  propositions: quatrePropositions(bonPays.nom, "nom"),
  image: bonPays.drapeau // l'adresse de l'image du drapeau
};
}
// ... (langue et voisin : ci-dessous)
}

```

2.2 Mode Langues

On choisit un pays qui a au moins une langue, on en tire une, et on demande qui la parle :

```

if (mode === "langue") {
  let p = choisirAuHasard(pays);
  while (!p.langues) { p = choisirAuHasard(pays); } // s'assurer qu'il a des langues
  const listeLangues = Object.values(p.langues); // ex. ["French", "Wolof"]
  const langue = choisirAuHasard(listeLangues);
  return {
    enonce: `Dans quel pays parle-t-on le ${langue} ?`,
    bonneReponse: p.nom,
    propositions: quatrePropositions(p.nom, "nom"),
    image: null
  };
}
}

```

Object.values, à prendre tel quel

Notre champ `langues` est un objet comme `{ fra: "French", wol: "Wolof" }`. `Object.values(...)` en extrait la liste des valeurs : `["French", "Wolof"]`. On peut alors en choisir une au hasard.

Une simplification assumée

Une langue peut être parlée dans plusieurs pays (le français, par exemple). Nos distracteurs étant tirés au hasard, l'un d'eux pourrait, rarement, parler aussi cette langue. À notre niveau, on l'accepte : l'important est de faire fonctionner le mode. On affinera dans une vraie version.

2.3 Mode Voisins (réponse numérique)

Ici la bonne réponse est un **nombre** (le nombre de voisins). On le transforme en texte, et on fabrique des distracteurs numériques proches :

```

if (mode === "voisin") {
  const nb = bonPays.voisins ? bonPays.voisins.length : 0;
  return {
    enonce: `Combien de pays voisins a ${bonPays.nom} ?`,
    bonneReponse: String(nb),
    propositions: quatreNombres(nb),
    image: null
  };
}
}

```

```
// 4 nombres : le bon + 3 voisins numériques (jamais négatifs)
function quatreNombres(bonNombre) {
  const props = [String(bonNombre)];
  while (props.length < 4) {
    const variation = bonNombre + Math.floor(Math.random() * 7) - 3; // +/- 3
    if (variation >= 0 && !props.includes(String(variation))) {
      props.push(String(variation));
    }
  }
  return melanger(props);
}
```

String(nb) convertit le nombre en texte, car nos boutons affichent et comparent du texte (Lab 7).

3 Afficher l'image quand il faut

Le mode Drapeaux a besoin d'une image. Ajoutez-la dans l'écran #quiz de index.html (cachée par défaut) :

```
<section id="quiz" class="cachee">
  <div id="piste-progression"><div id="barre-progression"></div></div>
  <p id="progression">Question 1 / 10</p>
  <img id="image-question" class="cachee" alt="Drapeau à identifier">
  <h2 id="enonce">Énoncé</h2>
  <div id="reponses">
    <button class="reponse">A</button>
    <button class="reponse">B</button>
    <button class="reponse">C</button>
    <button class="reponse">D</button>
  </div>
</section>
```

Puis on met à jour afficherQuestion pour montrer ou cacher l'image, et pour faire avancer la barre :

```
const elImage = document.querySelector("#image-question");
const barre = document.querySelector("#barre-progression");

function afficherQuestion() {
  const q = questions[indexQuestion];
  elProgression.textContent = `Question ${indexQuestion + 1} / ${questions.length}`;
  elEnonce.textContent = q.enonce;

  if (q.image) {
    elImage.src = q.image; // on met l'adresse de l'image
    elImage.classList.remove("cachee");
  } else {
    elImage.classList.add("cachee"); // pas d'image pour les autres modes
  }

  boutonsReponse.forEach((bouton, i) => {
    bouton.textContent = q.propositions[i];
  });

  // faire avancer la barre de progression
  const pourcentage = (indexQuestion / questions.length) * 100;
  barre.style.width = pourcentage + "%";
}
```

```
}
```

4 La barre de progression (CSS)

Une barre qui se remplit au fil des questions. Ajoutez à `style.css` :

```
#piste-progression {
  background: #eeeeee;
  border-radius: 6px;
  height: 10px;
  margin: 1rem 0;
}
#barre-progression {
  background: #c0392b; /* brique de la palette */
  height: 100%;
  width: 0;
  border-radius: 6px;
  transition: width 0.3s ease; /* l'animation douce du Lab 4 */
}
#image-question {
  max-width: 220px;
  display: block;
  margin: 1rem auto;
}
```

`element.style.width`, à prendre tel quel

En JavaScript, `barre.style.width = "40%"` change *directement* la largeur de l'élément. Combiné à la **transition** CSS, la barre se remplit en douceur. C'est le pont JS ↔ CSS : le code décide, le style anime.

Vérification

Lancez chaque mode l'un après l'autre. En mode Drapeaux, l'image s'affiche ; dans les autres, elle disparaît. La barre avance à chaque question. **Quatre jeux pour le prix d'un moteur !**

5 Exercices d'application

Exercice 1 — Mode « Superficie »

Ajoutez un cinquième mode : « Quel est le plus grand de ces pays ? ». *Indices* : (1) ajoutez `area` à la liste `fields=` de l'URL et au `map` de `chargerPays` (champ `superficie`). (2) Tirez quatre pays, la bonne réponse est celui de plus grande `superficie`.

Exercice 2 — Compteur d'erreurs

Ajoutez une variable `erreurs` (avec `let`), incrémentée dans le `else` de `repondre`, et affichez-la sur l'écran Résultat à côté du score.

6 Aide-mémoire du Lab 9

Besoin	Code
Ranger une info dans le HTML	<code>data-mode="drapeau"</code>
La relire en JS	<code>element.dataset.mode</code>
Accès par crochets	<code>objet[champ]</code>
Valeurs d'un objet	<code>Object.values(objet)</code>
Nombre → texte	<code>String(nombre)</code>
Afficher une image	<code>img.src = adresse</code>
Changer une dimension	<code>element.style.width = "40%"</code>

7 Sauvegarder mon travail (routine Git)

Routine Git de fin de Lab — tag v0.9

```
git status
git add .
git commit -m "Lab 9 : modes Drapeaux, Langues, Voisins + progression"
git tag v0.9
git push
git push --tags
```

Vérifiez sur GitHub que le commit *et* le tag v0.9 apparaissent.

Rapport à produire

Déposez `rapport-lab-9-NOM-Prenom.pdf` dans `rapports/`, avec quatre sections :

1. **Ce que j'ai fait** : les quatre modes, l'affichage du drapeau, la barre de progression.
2. **Difficultés rencontrées**.
3. **Comment je les ai résolues**.
4. **Lien vers mon repo** avec le tag v0.9.

8 Résumé et transition

Avec un seul moteur de quiz — une fonction `genererQuestion` qui prend le mode en paramètre — vous offrez quatre expériences de jeu. Vous savez désormais paramétrer un comportement, afficher des images, et animer une barre de progression.

Au **Lab 10**, on finit l'application : sauvegarder les meilleurs scores entre les sessions (`localStorage`), ajouter un mode chrono, et soigner l'accessibilité. Ce sera la dernière touche avant la mise en ligne.

Le jeu est complet. Au Lab 10, on le rend mémorable et accessible.

Dr. El Hadji Bassirou TOURÉ — École Supérieure Polytechnique — UCAD
Licence 1 DUT1-Informatique — Fondamentaux du Web — 2025–2026