

Fondamentaux du Web

HTML, CSS, JavaScript

Lab 10 — Sauvegarde et finitions

localStorage, chrono, accessibilité, polish

Licence 1 DUT1-Informatique — 2025–2026

Où en sommes-nous ?

Au Lab 9, Jàng Afrig propose **quatre modes de jeu** (Capitales, Drapeaux, Langues, Voisins) avec les vraies données de l'API. L'application est riche, mais il lui manque les *finitions* d'un vrai produit : se souvenir des scores, un mode chronométré, et le soin du détail (accessibilité, confort).

Avant de commencer : le Lab 9 est terminé (tag v0.9), les quatre modes fonctionnent.

Validation de fin : les scores sont sauvegardés et survivent à la fermeture du navigateur, le mode Chrono fonctionne, l'application est soignée. Tag v0.14 poussé + rapport.

Objectifs du Lab 10

À la fin, vous saurez :

1. Sauvegarder des données dans le navigateur avec `localStorage`.
2. Convertir objets et tableaux en texte avec `JSON.stringify` / `JSON.parse`.
3. Enregistrer et afficher un classement des meilleurs scores.
4. Effacer des données proprement (avec confirmation).
5. Créer un **mode Chrono** avec `setInterval` / `clearInterval`.
6. Rendre l'application plus **accessible** (textes alternatifs, navigation au clavier).

Table des matières

Rappel express du Lab 9	1
1 localStorage : la mémoire du navigateur	1
2 Sauvegarder un score	2
3 La page « Mes Scores » (top 5)	2
4 Effacer les scores (avec confirmation)	3
5 Le mode Chrono	3
6 La barre de temps (CSS)	4

7	Accessibilité : penser à tout le monde	4
8	Navigation au clavier	5
9	Polish et idées bonus (facultatif)	5
10	Aide-mémoire du Lab 10	5
11	Sauvegarder mon travail (routine Git)	5
12	Résumé et transition	6

Rappel express du Lab 9

Vous avez quatre modes de jeu, une fonction `genererQuestion(mode)`, et un score à la fin de chaque partie. Aujourd'hui, on garde ce score en mémoire et on peaufine.

1 localStorage : la mémoire du navigateur

Jusqu'ici, dès qu'on recharge la page, tout est oublié : le score disparaît. `localStorage` est un petit espace de rangement, propre à votre navigateur, qui **garde les données même après fermeture**. *Analogie* : un cahier de brouillon que le navigateur range dans un tiroir, et qu'il retrouve à chaque visite.

```
localStorage.setItem("pseudo", "Fatou"); // écrire
const p = localStorage.getItem("pseudo"); // lire -> "Fatou"
localStorage.removeItem("pseudo");       // effacer
```

`localStorage` — le tiroir du navigateur

"pseudo"	→	"Fatou"
"scores"	→	"[{...},{...}]"

(une clé, une valeur — en texte)

Survit à la fermeture de l'onglet et au redémarrage de l'ordinateur.

localStorage ne range que du *texte*

On ne peut y mettre que des chaînes de caractères. Pour ranger un **tableau** ou un **objet**, il faut d'abord le transformer en texte avec `JSON.stringify`, et le retransformer en le relisant avec `JSON.parse`.

```
const scores = [{ pseudo: "Fatou", mode: "capitale", score: 8 }];

localStorage.setItem("scores", JSON.stringify(scores)); // objet -> texte
const lus = JSON.parse(localStorage.getItem("scores")); // texte -> objet
```

2 Sauvegarder un score

À la fin de chaque partie, on ajoute le score à la liste déjà rangée. On lit l'ancienne liste (ou un tableau vide la toute première fois), on y ajoute, on réenregistre.

```
function sauverScore(pseudo, mode, score, total) {
  const ancien = localStorage.getItem("scores");
  const scores = ancien ? JSON.parse(ancien) : []; // [] si rien encore

  scores.push({ pseudo: pseudo, mode: mode, score: score, total: total });
  localStorage.setItem("scores", JSON.stringify(scores));
}
```

On appelle `sauverScore(...)` dans `terminerQuiz()`, juste après avoir calculé le score. Le `ancien ? ... : []` est le ternaire du CM3 : s'il n'y a rien encore, on part d'une liste vide.

3 La page « Mes Scores » (top 5)

On affiche les cinq meilleurs scores. On lit la liste, on la **trie** du plus grand au plus petit, on garde les cinq premiers, et on les affiche.

```
function afficherMeilleurs() {
  const ancien = localStorage.getItem("scores");
  const scores = ancien ? JSON.parse(ancien) : [];

  scores.sort((a, b) => b.score - a.score); // trie (à prendre tel quel)
  const top5 = scores.slice(0, 5);          // garde les 5 premiers

  const liste = document.querySelector("#liste-scores");
  liste.textContent = ""; // on vide avant de remplir
  top5.forEach((s) => {
    const item = document.createElement("li");
    item.textContent = `${s.pseudo} --- ${s.mode} : ${s.score}/${s.total}`;
    liste.appendChild(item);
  });
}
```

Deux lignes à prendre telles quelles

`scores.sort((a, b) => b.score - a.score)` range les scores *du plus grand au plus petit*. `scores.slice(0, 5)` renvoie un nouveau tableau avec seulement les éléments d'index 0 à 4. Recopiez-les telles quelles : on en reparlera en L2.

4 Effacer les scores (avec confirmation)

Un bouton « Effacer mes scores ». Comme c'est irréversible, on demande confirmation avec `confirm(...)`, qui renvoie `true` si l'utilisateur clique « OK ».

```
function effacerScores() {
  const ok = confirm("Effacer définitivement tous tes scores ?");
  if (ok) {
    localStorage.removeItem("scores");
    afficherMeilleurs(); // la liste se rafraîchit (vide)
  }
}
```

5 Le mode Chrono

Un compte à rebours ajoute du piment. `setInterval(fonction, 1000)` exécute une fonction toutes les 1000 millisecondes (1 seconde). `clearInterval(...)` l'arrête. *Analogie* : on remonte un minuteur de cuisine; il sonne chaque seconde jusqu'à zéro, et on peut l'arrêter à tout moment.

```
let tempsRestant = 30;
let chrono = null; // l'identifiant du minuteur

function demarrerChrono() {
  tempsRestant = 30;
  elTemps.textContent = tempsRestant;

  chrono = setInterval(() => {
    tempsRestant = tempsRestant - 1;
    elTemps.textContent = tempsRestant;

    if (tempsRestant <= 0) {
      arreterChrono();
      terminerQuiz(); // le temps est écoulé
    }
  }, 1000); // toutes les 1 seconde
}

function arreterChrono() {
  clearInterval(chrono);
}
```

Toujours arrêter son minuteur

Si on oublie `clearInterval`, le minuteur continue de tourner en arrière-plan, même après la fin du quiz. On l'arrête donc dès qu'on n'en a plus besoin (fin du temps, ou réponse à la dernière question).

6 La barre de temps (CSS)

Pour visualiser le temps, une barre qui rétrécit et passe au **rouge** dans les 10 dernières secondes. En CSS, on prépare la barre et une classe « urgente » :

```
#barre-temps {
  height: 12px;
  background: #2a9d8f; /* vert calme */
  width: 100%;
  transition: width 1s linear;
}
#barre-temps.urgent { background: #e63946; } /* rouge */
```

Puis, dans le tick du chrono, on ajuste la largeur et on ajoute la classe au bon moment :

```
const pourcent = (tempsRestant / 30) * 100;
barreTemps.style.width = pourcent + "%"; // rétrécit la barre

if (tempsRestant <= 10) {
  barreTemps.classList.add("urgent"); // passe au rouge
}
```

`element.style.width` modifie directement le style d'un élément depuis JavaScript — pratique pour une valeur qui change en continu.

7 Accessibilité : penser à tout le monde

Une bonne application est utilisable par **tous**, y compris les personnes qui naviguent avec un lecteur d'écran (qui lit la page à voix haute) ou uniquement au clavier. Deux gestes simples et importants :

- **Texte alternatif sur les images** (`alt`) : pour le mode Drapeaux, le lecteur d'écran doit pouvoir décrire l'image.
- **Étiquette sur les boutons-icônes** (`aria-label`) : un bouton qui n'affiche qu'un symbole a besoin d'un nom lisible.

```

<button aria-label="Rejouer la partie">⌂</button>
```

En mode Drapeaux, on remplit donc le `alt` en JavaScript au moment d'afficher le drapeau :

```
imageDrapeau.src = bonPays.flags.svg;
imageDrapeau.alt = "Drapeau à identifier";
```

8 Navigation au clavier

Confort supplémentaire : répondre avec les touches **1, 2, 3, 4** plutôt qu'à la souris. On écoute l'événement `keydown` sur toute la page; `event.key` contient la touche pressée (un texte).

```
document.addEventListener("keydown", (event) => {
  if (event.key === "1") { repondre(boutonsReponse[0].textContent); }
  if (event.key === "2") { repondre(boutonsReponse[1].textContent); }
  if (event.key === "3") { repondre(boutonsReponse[2].textContent); }
  if (event.key === "4") { repondre(boutonsReponse[3].textContent); }
});
```

9 Polish et idées bonus (facultatif)

Quelques finitions à tenter

1. Afficher le **meilleur score** directement sur l'écran d'accueil.
2. Un petit message d'encouragement selon le score (« Excellent ! », « Continue ! »).
3. Vérifier que les messages d'erreur réseau (Lab 8) sont clairs et rassurants.
4. Soigner les transitions (réutiliser celles du Lab 4) pour des changements d'écran fluides.

10 Aide-mémoire du Lab 10

Besoin	Code
Écrire / lire une donnée	<code>localStorage.setItem(cle, texte)</code> / <code>getItem(cle)</code>
Effacer une donnée	<code>localStorage.removeItem(cle)</code>
Objet → texte / texte → objet	<code>JSON.stringify(obj)</code> / <code>JSON.parse(txt)</code>
Demander confirmation	<code>confirm("...")</code> (renvoie true/false)
Répéter chaque seconde	<code>setInterval(fonction, 1000)</code>
Arrêter le minuteur	<code>clearInterval(chrono)</code>
Changer un style en direct	<code>element.style.width = "50%"</code>
Touche pressée	<code>event.key</code> (dans un écouteur keydown)

11 Sauvegarder mon travail (routine Git)

Routine Git de fin de Lab — tag v0.14

```
git status
git add .
git commit -m "Lab 10 : localStorage, chrono et accessibilite"
git tag v0.14
git push
git push --tags
```

On passe directement à v0.14 pour garder le tag aligné sur le numéro du Lab. Vérifiez le commit *et* le tag sur GitHub.

Rapport à produire

Déposez `rapport-lab-10-NOM-Prenom.pdf` dans `rapports/`, avec quatre sections :

1. **Ce que j'ai fait** : sauvegarde des scores, page des meilleurs, mode Chrono, accessibilité.
2. **Difficultés rencontrées** (messages d'erreur copiés).
3. **Comment je les ai résolues**.
4. **Lien vers mon repo** avec le tag v0.14.

12 Résumé et transition

Jàng Afrig est maintenant une **application complète** : quatre modes de jeu, de vraies données, des scores mémorisés, un mode chronométré, et le souci de tous ses utilisateurs. C'est un vrai petit produit, dont vous pouvez être fier.

Il ne reste qu'une étape, et non des moindres : le **mettre en ligne**. À la séance 15, on déploiera Jàng Afrig sur **GitHub Pages**. Vous obtiendrez une **adresse web publique**, accessible depuis n'importe quel téléphone — celui d'un ami, d'un parent. Et on posera le tag final : v1.0.

L'application est prête. À la S15, le monde entier pourra y jouer.

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