```
document.addEventListener('DOMContentLoaded', function() {
  // Sélectionner les éléments nécessaires
  var explorateurCheckbox = document.getElementById('explorateur');
  var lastCoordsSpan = document.getElementById('lastCoords');
  var coordValueElement = document.getElementById('coordValue');
  var coordField = document.getElementById('coords_colonie');
  var dateRapportField = document.getElementById('date_rapport'); // Champ "date du rapport"
  var form = document.getElementById('rapportForm');
  var extraFields = document.getElementById('extraFields');
  var toggleButton = document.getElementById('toggleButton');

  if (!form) {
    console.error('L\'élément avec l\'ID "rapportForm" n\'a pas été trouvé.');
    return;
  }

  function updateCoords() {
    fetch('explorateur.php')
      .then(response => response.text())  // Lire la réponse en tant que texte
      .then(text => {
        console.log('Réponse brute:', text); // Affichez la réponse brute
        try {
          const data = JSON.parse(text); // Essayez de convertir le texte en JSON
          console.log('Données reçues:', data);
          const lastCoords = data.last_system || 'Aucune coordonnée trouvée';

          // Mettre à jour l'affichage des coordonnées
```

```
            coordValueElement.textContent = lastCoords;
            coordField.value = lastCoords;

            // Sauvegarder la coordonnée dans le stockage local
            localStorage.setItem('lastExploredCoord', lastCoords);
          } catch (e) {
            console.error('Erreur lors de l\'analyse JSON:', e);
          }
        })
        .catch(error => console.error('Erreur lors de la récupération des coordonnées :', error));
  }

  // Exécuter la mise à jour des coordonnées au chargement
  updateCoords();

  // Fonction pour masquer les champs complémentaires
  function masquerChampsComplementaires() {
    extraFields.style.display = 'none';
    toggleButton.textContent = 'Déployer le formulaire';
  }

  // Fonction pour afficher les champs complémentaires
  function afficherChampsComplementaires() {
    extraFields.style.display = 'block';
    toggleButton.textContent = 'Réduire le formulaire';
  }
```

```
  // Fonction pour basculer entre l'affichage et le masquage des champs complémentaires
  function toggleChampsComplementaires() {
    if (extraFields.style.display === 'none' || extraFields.style.display === '') {
      afficherChampsComplementaires();
    } else {
      masquerChampsComplementaires();
    }
  }

  // Ajouter un écouteur d'événements au bouton de déploiement du formulaire
  if (toggleButton) {
    toggleButton.addEventListener('click', toggleChampsComplementaires);
  }

  // Gérer l'affichage de la coordonnée lorsque l'option Explorateur est cochée
  if (explorateurCheckbox) {
    // Initialiser l'état de la case à cocher depuis le stockage local
    explorateurCheckbox.checked = localStorage.getItem('explorateurChecked') === 'true';

    // Afficher ou masquer les coordonnées basées sur l'état de la case à cocher
    if (explorateurCheckbox.checked) {
      lastCoordsSpan.style.display = 'inline';
      var storedCoord = localStorage.getItem('lastExploredCoord');
      if (storedCoord) {
        coordValueElement.textContent = storedCoord; // Afficher la coordonnée stockée
```

```
            coordField.value = storedCoord; // Assurer que le champ de coordonnée est mis à jour
        } else {
            updateCoords(); // Récupérer la coordonnée de la base de données
        }
    } else {
        lastCoordsSpan.style.display = 'none';
    }

    // Gérer le changement d'état de la case à cocher Explorateur
    explorateurCheckbox.addEventListener('change', function() {
        if (this.checked) {
            lastCoordsSpan.style.display = 'inline';
            updateCoords(); // Mettre à jour les coordonnées
        } else {
            lastCoordsSpan.style.display = 'none';
        }
        // Sauvegarder l'état de la case dans le stockage local
        localStorage.setItem('explorateurChecked', this.checked);
    });
}

// Gestion de la soumission du formulaire
if (form) {
    form.addEventListener('submit', function(event) {
        // Vérifier si le champ "date du rapport" est vide
        if (dateRapportField && dateRapportField.value === '') {
            // Si vide, insérer la date actuelle
```

```
            var today = new Date();
            var day = String(today.getDate()).padStart(2, '0');
            var month = String(today.getMonth() + 1).padStart(2, '0'); // Les mois commencent à 0
            var year = today.getFullYear();

            // Formater la date au format AAAA-MM-JJ (format SQL)
            var currentDate = year + '-' + month + '-' + day;

            // Mettre à jour le champ "date du rapport" avec la date du jour
            dateRapportField.value = currentDate;
        }

        // Permettre au formulaire d'être soumis après la modification
    });
}

// Initialiser l'affichage des champs complémentaires au chargement de la page
masquerChampsComplementaires(); // On masque les champs supplémentaires par défaut

// Mise à jour des coordonnées au chargement de la page, même si explorateur est coché
if (explorateurCheckbox.checked) {
    updateCoords(); // Assurer que la coordonnée est à jour dès le chargement
}

// Fonction pour vérifier si le joueur existe
```

```
function checkPlayerExists(nomJoueur, callback) {
    var xhr = new XMLHttpRequest();
    xhr.open('POST', 'check_player_exists.php', true);
    xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xhr.onreadystatechange = function () {
        if (xhr.readyState === XMLHttpRequest.DONE) {
            if (xhr.status === 200) {
                try {
                    var response = JSON.parse(xhr.responseText);
                    if (callback) {
                        callback(response.exists); // Passer uniquement `exists` à la fonction de
rappel
                    }
                } catch (e) {
                    console.error("Erreur lors du parsing de la réponse JSON", e);
                }
            } else {
                console.error("Erreur de requête :", xhr.status, xhr.statusText);
            }
        }
    };
    xhr.send('nom_joueur=' + encodeURIComponent(nomJoueur));
}

// Fonction pour formater les nombres avec des espaces pour la lisibilité
function formatNumberWithSpaces(number) {
    return number.toString().replace(/\B(?=(\d{3})+(?!\d))/g, ' ');
}
```

```
// Fonction pour extraire les informations du rapport et remplir les champs
function extractAndFillFields() {
  var rapportTexte = document.getElementById('info').value;
  var infos = extraireInformationsRapport(rapportTexte);

  // Log pour vérifier les informations extraites
  console.log("Informations extraites :", infos);

  // Stockage des informations dans sessionStorage dans l'ordre défini
  if (infos.nom_joueur) {
    sessionStorage.setItem('nom_joueur', infos.nom_joueur);
  }

  if (infos.nom_colonie) {
    sessionStorage.setItem('nom_colonie', infos.nom_colonie);
  }
  if (infos.coordonnees) {
    sessionStorage.setItem('coords_colonie', infos.coordonnees);
  }
  if (infos.date_rapport) {
    sessionStorage.setItem('date_rapport', infos.date_rapport);
  }
  if (infos.ressources.fer !== null) {
    sessionStorage.setItem('fer', infos.ressources.fer);
    // Debug: Affichage des valeurs avant envoi
    console.log("Fer avant formatage :", infos.ressources.fer);
```

```
        console.log("Fer après formatage :", formatNumberWithSpaces(infos.ressources.fer));
    }
    if (infos.ressources.or !== null) {
        sessionStorage.setItem('or', infos.ressources.or);
        // Debug: Affichage des valeurs avant envoi
        console.log("Or avant formatage :", infos.ressources.or);
        console.log("Or après formatage :", formatNumberWithSpaces(infos.ressources.or));
    }
    if (infos.ressources.cristal !== null) {
        sessionStorage.setItem('cristal', infos.ressources.cristal);
        // Debug: Affichage des valeurs avant envoi
        console.log("Cristal avant formatage :", infos.ressources.cristal);
        console.log("Cristal après formatage :", formatNumberWithSpaces(infos.ressources.cristal));
    }
    if (infos.ressources.hydrogene !== null) {
        sessionStorage.setItem('hydrogene', infos.ressources.hydrogene);
        // Debug: Affichage des valeurs avant envoi
        console.log("Hydrogène avant formatage :", infos.ressources.hydrogene);
        console.log("Hydrogène après formatage :", formatNumberWithSpaces(infos.ressources.hydrogene));
    }
  }

  // Vérifier si le joueur existe lorsque le texte change dans la textarea
  document.getElementById('info').addEventListener('input', function() {
    var rapportTexte = this.value;
```

```
    var matchNomJoueur = rapportTexte.match(/du joueur\s+([^\n]+)/);


    if (matchNomJoueur) {
      var nomJoueur = matchNomJoueur[1].trim();
      var nomJoueurField = document.getElementById('nom_joueur');
      if (nomJoueurField) {
        nomJoueurField.value = nomJoueur;
      }
      checkPlayerExists(nomJoueur, function(existe) {
        joueurConnu = existe; // Met à jour la variable en fonction de la réponse


        if (joueurConnu) {
          masquerChampsComplementaires(); // Masquer les champs si le joueur est connu
        } else {
          afficherChampsComplementaires(); // Afficher les champs si le joueur est inconnu
        }
      });
    } else {
      joueurConnu = false; // Assumer que le joueur est nouveau si le nom n'est pas trouvé
      masquerChampsComplementaires(); // Masquer les champs si aucun nom de joueur n'est trouvé
      var nomJoueurField = document.getElementById('nom_joueur');
      if (nomJoueurField) {
        nomJoueurField.value = ''; // Effacer le champ nom_joueur
      }
    }
```

```
  });

  // Fonction pour vérifier les coordonnées avant la soumission du formulaire
  function verifierCoordonnees(event) {
    event.preventDefault();

    var coordsElement =
document.querySelector('[name="colonies[0][coords_colonie]"]');
    var nomColonieElement =
document.querySelector('[name="colonies[0][nom_colonie]"]');
    var dateRapportElement = document.getElementById('date_rapport');

    if (coordsElement && nomColonieElement && dateRapportElement) {
      var coords = coordsElement.value;
      var nomColonie = nomColonieElement.value;
      var dateRapport = dateRapportElement.value;

      var xhr = new XMLHttpRequest();
      xhr.open("GET", "checkDuplicate.php?coords_colonie=" +
encodeURIComponent(coords) +
                  "&nom_colonie=" + encodeURIComponent(nomColonie) +
                  "&date_rapport=" + encodeURIComponent(dateRapport), true);
      xhr.onreadystatechange = function () {
        if (xhr.readyState === XMLHttpRequest.DONE) {
          if (xhr.status === 200) {
            console.log('Réponse brute du serveur:', xhr.responseText); // Afficher la
réponse brute
            try {
              var response = JSON.parse(xhr.responseText);
```

```
                if (response.duplicate) {
                    // Si un doublon est détecté, rediriger vers choixAdresse.html pour les modifications
                    window.location.href = 'choix.html?coords=' + encodeURIComponent(coords) +
                                           '&nom_colonie=' + encodeURIComponent(nomColonie) +
                                           '&date_rapport=' + encodeURIComponent(dateRapport);
                } else {
                    // Si aucun doublon n'est détecté, soumettre le formulaire normalement
                    document.getElementById('rapportForm').submit();
                }
            } catch (e) {
                console.error("Erreur lors du parsing de la réponse JSON", e);
            }
        } else {
            console.error("Erreur de requête :", xhr.status, xhr.statusText);
        }
      }
    };
    xhr.send();
  } else {
    console.error('Les champs requis sont introuvables.');
  }
}

// Fonction pour extraire les informations du rapport
function extraireInformationsRapport(rapportTexte) {
  var regexDate = /^(\d{2})\/(\d{2})\/(\d{4}) (\d{2}):(\d{2}):(\d{2})/;
  var regexNomJoueur = /du joueur\s+([^\n]+)/;
```

```
var regexNomColonie = /planète\s+([^\[]+)/;
var regexCoordonnees = /\[(\d+:\d+:\d+)\]/;

// Expressions régulières pour les ressources
var regexFer = /Fer:\s*([\d.,]+)/;
var regexOr = /Or:\s*([\d.,]+)/;
var regexCristal = /Cristal:\s*([\d.,]+)/;
var regexHydrogene = /Hydrogène:\s*([\d.,]+)/;

// Expressions régulières pour les militaires
var regexSoldats = /Soldats:\s*([\d.,]+)/;
var regexColonels = /Colonels:\s*([\d.,]+)/;
var regexTechniciens = /Techniciens:\s*([\d.,]+)/;
var regexExperts = /Experts:\s*([\d.,]+)/;
var regexEspions = /Espions:\s*([\d.,]+)/;
var regexKamikazes = /Kamikazes:\s*([\d.,]+)/;
var regexMarines = /Marines:\s*([\d.,]+)/;
var regexUniteElite = /Unité_Élite:\s*([\d.,]+)/;
var regexBioSoldat = /Bio-Soldat:\s*([\d.,]+)/;
var regexAgentSecret = /Agent_Secret:\s*([\d.,]+)/;
var regexSoldatDroide = /Soldat_Droïde:\s*([\d.,]+)/;
var regexAndroideCombat = /Androïde_Combat:\s*([\d.,]+)/;
var regexNanosoldat = /Nanosoldat:\s*([\d.,]+)/;
var regexSniper = /Sniper:\s*([\d.,]+)/;

// Initialiser l'objet informations
var informations = {
   date_rapport: null,
```

```
nom_joueur: null,
nom_colonie: null,
coordonnees: null,
ressources: {
  fer: null,
  or: null,
  cristal: null,
  hydrogene: null
},
militaires: {
  soldats: null,
  colonels: null,
  techniciens: null,
  experts: null,
  espions: null,
  kamikazes: null,
  marines: null,
  unite_elite: null,
  bio_soldat: null,
  agent_secret: null,
  soldat_droide: null,
  androide_combat: null,
  nanosoldat: null,
  sniper: null
},
vaisseaux: null,
defenses: null,
bombes: null,
```

```
    batiments_ressources: null,
    batiments_militaires: null,
    batiments_scientifiques: null,
    laboratoire: null,
    zone51: null,
    artefacts: null
};

// Extraire les valeurs de date, joueur, colonie, et coordonnées
var matchDate = rapportTexte.match(regexDate);
var matchNomJoueur = rapportTexte.match(regexNomJoueur);
var matchNomColonie = rapportTexte.match(regexNomColonie);
var matchCoordonnees = rapportTexte.match(regexCoordonnees);

if (matchDate) {
    informations.date_rapport = `${matchDate[3]}/${matchDate[2]}/${matchDate[1]}`;
}

if (matchNomJoueur) {
    informations.nom_joueur = matchNomJoueur[1].trim();
}

if (matchNomColonie) {
    informations.nom_colonie = matchNomColonie[1].trim();
}

if (matchCoordonnees) {
    informations.coordonnees = matchCoordonnees[1];
```

```
  }

  // Fonction pour extraire les ressources
  var parseResource = function(match, type) {
    if (match) {
      var value = match[1].replace(/\./g, '').replace(/,/g, '');
      return parseInt(value, 10);
    }
    return null;
  };

  informations.ressources.fer = parseResource(rapportTexte.match(regexFer), 'Fer');
  informations.ressources.or = parseResource(rapportTexte.match(regexOr), 'Or');
  informations.ressources.cristal = parseResource(rapportTexte.match(regexCristal),
'Cristal');
  informations.ressources.hydrogene =
parseResource(rapportTexte.match(regexHydrogene), 'Hydrogène');

  // Fonction pour extraire et formater les informations militaires
  var parseMilitaryInfo = function(regex, key) {
    var match = rapportTexte.match(regex);
    if (match) {
      return formatNumberWithSpaces(parseInt(match[1].replace(/\./g, ''), 10));
    }
    return null;
  };

  informations.militaires.soldats = parseMilitaryInfo(regexSoldats, 'soldats');
  informations.militaires.colonels = parseMilitaryInfo(regexColonels, 'colonels');
```

```
    informations.militaires.techniciens = parseMilitaryInfo(regexTechniciens,
'techniciens');
    informations.militaires.experts = parseMilitaryInfo(regexExperts, 'experts');
    informations.militaires.espions = parseMilitaryInfo(regexEspions, 'espions');
    informations.militaires.kamikazes = parseMilitaryInfo(regexKamikazes, 'kamikazes');
    informations.militaires.marines = parseMilitaryInfo(regexMarines, 'marines');
    informations.militaires.unite_elite = parseMilitaryInfo(regexUniteElite, 'unite_elite');
    informations.militaires.bio_soldat = parseMilitaryInfo(regexBioSoldat, 'bio_soldat');
    informations.militaires.agent_secret = parseMilitaryInfo(regexAgentSecret,
'agent_secret');
    informations.militaires.soldat_droide = parseMilitaryInfo(regexSoldatDroide,
'soldat_droide');
    informations.militaires.androide_combat = parseMilitaryInfo(regexAndroideCombat,
'androide_combat');
    informations.militaires.nanosoldat = parseMilitaryInfo(regexNanosoldat,
'nanosoldat');
    informations.militaires.sniper = parseMilitaryInfo(regexSniper, 'sniper');

    // Log des valeurs extraites pour le debug
    console.log("Informations extraites : ", informations);

    return informations;
}

// Fonction pour formater les nombres avec des espaces
function formatNumberWithSpaces(value) {
    return value.toLocaleString('fr-FR'); // Format avec espaces pour les nombres
}

    function extractAndFillFields() {
```

```
    var rapportTexte = document.getElementById('info').value;
    var infos = extraireInformationsRapport(rapportTexte);

    console.log('Nom du joueur extrait :', infos.nom_joueur);

    if (infos.nom_joueur) {
      document.getElementById('nom_joueur').value = infos.nom_joueur;
      // Déclencher l'événement blur pour mettre à jour les champs complémentaires
      document.getElementById('nom_joueur').dispatchEvent(new Event('blur'));
    }

    if (infos.nom_colonie) {
      document.querySelector('[name="colonies[0][nom_colonie]"]').value =
infos.nom_colonie;
    }

    if (infos.coordonnees) {
      document.querySelector('[name="colonies[0][coords_colonie]"]').value =
infos.coordonnees;
    }

    if (infos.date_rapport) {
      document.getElementById('date_rapport').value = infos.date_rapport;
    }

    if (infos.ressources) {
      document.getElementById('ressources').value = JSON.stringify(infos.ressources);
    }
```

```
    if (infos.fer) {
        document.getElementById('fer').value = infos.fer;
    }

    if (infos.or) {
        document.getElementById('or').value = infos.or;
    }

    if (infos.cristal) {
        document.getElementById('cristal').value = infos.cristal;
    }

    if (infos.hydrogene) {
        document.getElementById('hydrogene').value = infos.hydrogene;
    }

    if (infos.vaisseaux) {
        document.getElementById('vaisseaux').value = infos.vaisseaux.join('\n');
    }

    if (infos.militaires) {
        document.getElementById('militaires').value = Array.isArray(infos.militaires) ?
infos.militaires.join('\n') : infos.militaires;
    }

    if (infos.batiments_ressources) {
        document.getElementById('batiments_ressources').value =
infos.batiments_ressources.join('\n');
    }
```

```
    if (infos.batiments_militaires) {
      document.getElementById('batiments_militaires').value =
infos.batiments_militaires.join('\n');
    }

    if (infos.batiments_scientifiques) {
      document.getElementById('batiments_scientifiques').value =
infos.batiments_scientifiques.join('\n');
    }

    if (infos.laboratoire) {
      document.getElementById('laboratoire').value = infos.laboratoire.join('\n');
    }

    if (infos.zone51) {
      document.getElementById('zone51').value = infos.zone51.join('\n');
    }

    if (infos.artefacts) {
      document.getElementById('artefacts').value = infos.artefacts.join('\n');
    }

    extraireEtRemplirDefenses(rapportTexte);
    extraireEtRemplirBombes(rapportTexte);
  }

    function extraireEtRemplirDefenses(rapportTexte) {
    var regexDefenses = {
```

```
    tour_combat: /Tour_de_Combat:\s*([\d.]+)/,
    canon_laser: /Canon_Laser:\s*([\d.]+)/,
    grand_canon_laser: /Grand_Canon_Laser:\s*([\d.]+)/,
    rayon_tracteur: /Rayon_Tracteur:\s*([\d.]+)/,
    lanceur_missiles: /Lanceur_de_Missiles:\s*([\d.]+)/,
    satellite_ions: /Satellite_à_Ions:\s*([\d.]+)/,
    batterie_electromagnetique: /Batterie_Electromagnétique:\s*([\d.]+)/,
    canon_plasma: /Canon_à_Plasma:\s*([\d.]+)/,
    canon_electromagnetique: /Canon_Electromagnétique:\s*([\d.]+)/,
    silos_missiles_hem: /Silos_à_Missiles_HEM:\s*([\d.]+)/,
    complexe_defense_orbital: /Complexe_de_Défense_Orbital:\s*([\d.]+)/,
    missile_interception: /Missile_d'Interception:\s*([\d.]+)/
  };

  for (var defense in regexDefenses) {
    var match = rapportTexte.match(regexDefenses[defense]);
    if (match) {
      document.getElementById(defense).value = match[1];
    }
  }
}

  function extraireEtRemplirBombes(rapportTexte) {
  var regexBombes = {
    bombe_nucleaire: /Bombe_Nucléaire:\s*([\d.]+)/,
    bombe_plasma: /Bombe_Plasma:\s*([\d.]+)/,
    bombe_electromagnetique: /Bombe_Électromagnétique:\s*([\d.]+)/,
    bombe_antimatiere: /Bombe_Antimatière:\s*([\d.]+)/
```

```
    };

    for (var bombe in regexBombes) {
        var match = rapportTexte.match(regexBombes[bombe]);
        if (match) {
            document.getElementById(bombe).value = match[1];
        }
    }
}

    var submitButton = document.querySelector('#rapportForm button[type="submit"]');
    if (submitButton) {
        submitButton.addEventListener('click', verifierCoordonnees);
    } else {
        console.error('Le bouton de soumission est introuvable.');
    }

    // Ajouter un événement pour réinitialiser le formulaire si nécessaire
    var resetButton = document.querySelector('#resetButton');
    if (resetButton) {
        resetButton.addEventListener('click', resetForm);
    } else {
        console.error('Le bouton de réinitialisation est introuvable.');
    }

    // Extraire et remplir les champs lorsque le rapport est fourni
    document.getElementById('info').addEventListener('input', extractAndFillFields);
```

```
    // Fonction pour réinitialiser le formulaire
    function resetForm() {
      var form = document.getElementById('rapportForm');
      if (form) {
        form.reset();
        masquerChampsComplementaires(); // Masquer les champs complémentaires lors de la réinitialisation
      }
    }
  });

  document.addEventListener('DOMContentLoaded', function() {
    console.log('Essayez de trouver le formulaire...');
    const form = document.getElementById('rapportForm');
    console.log('Formulaire trouvé:', form);

    form.addEventListener('submit', function(event) {
      event.preventDefault(); // Empêche le rechargement de la page

      var coordsElement = document.querySelector('[name="colonies[0][coords_colonie]"]');
      var nomColonieElement = document.querySelector('[name="colonies[0][nom_colonie]"]');
      var dateRapportElement = document.getElementById('date_rapport');

      if (coordsElement && nomColonieElement && dateRapportElement) {
        var coords = coordsElement.value;
        var nomColonie = nomColonieElement.value;
        var dateRapport = dateRapportElement.value;
```

```
        console.log("Coordonnées:", coords);
        console.log("Nom de la colonie:", nomColonie);
        console.log("Date du rapport:", dateRapport);

        // Vérifier les doublons via une requête AJAX
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "checkDuplicate.php?coords_colonie=" +
encodeURIComponent(coords) +
                "&nom_colonie=" + encodeURIComponent(nomColonie) +
                "&date_rapport=" + encodeURIComponent(dateRapport), true);

        xhr.onreadystatechange = function () {
          if (xhr.readyState === XMLHttpRequest.DONE) {
            console.log("Réponse reçue :", xhr.responseText); // Affiche la réponse
complète
            if (xhr.status === 200) {
              try {
                var response = JSON.parse(xhr.responseText);
                console.log("Parsed response:", response); // Affiche la réponse JSON

                if (response.duplicate) {
                  console.log("Doublon détecté, redirection vers choix.html");
                  const params = new URLSearchParams({
                    coords: coords,
                    nom_colonie: nomColonie,
                    date_rapport: dateRapport
                  });
                  window.location.href = 'choix.html?' + params.toString();
```

```
                } else {
                    console.log("Aucun doublon, envoi du formulaire");
                    const formData = new FormData(form);

                    // Envoi du formulaire à AnalyseRapport.php
                    fetch('AnalyseRapport.php', {
                        method: 'POST',
                        body: formData
                    })
                    .then(response => response.json())
                    .then(data => {
                        // Gérer la réponse de AnalyseRapport.php
                        if (data.doublon) {
                            console.log("Doublon détecté dans AnalyseRapport, redirection vers choix.html");
                            window.location.href = 'choix.html?coords=' + encodeURIComponent(coords) +
                                                   '&nom_colonie=' + encodeURIComponent(nomColonie) +
                                                   '&date_rapport=' + encodeURIComponent(dateRapport);
                        } else {
                            // Soumission réussie, afficher un message
                            alert('Rapport soumis avec succès !');
                        }
                    })
                    .catch(error => {
                        console.error('Erreur lors de la soumission :', error);
                    });
```

```
                }
            } catch (e) {
                console.error("Erreur lors du parsing de la réponse JSON", e);
            }
        } else {
            console.error("Erreur de requête :", xhr.status, xhr.statusText);
        }
      }
    };
    xhr.send();
  } else {
    console.error('Les champs requis sont introuvables.');
  }
});

// Gérer l'affichage des champs supplémentaires
document.getElementById('toggleButton').addEventListener('click', function() {
  var extraFields = document.getElementById('extraFields');
  if (extraFields.style.display === 'none') {
    extraFields.style.display = 'block';
    this.textContent = 'Masquer le formulaire';
  } else {
    extraFields.style.display = 'none';
    this.textContent = 'Déployer le formulaire';
  }
});

// Réinitialiser le formulaire
```

```
    document.getElementById('resetButton').addEventListener('click', function() {
        document.getElementById('rapportForm').reset();
    });
});
```