

# Jalon 1 : Phase critique

L'objectif est à la fois d'afficher un calendrier, mais aussi de construire un moteur capable de naviguer avec précision à travers l'histoire des réformes calendaires.

Voici le détail technique de ce jalon :

## 1. Structure de l'Architecture (HTML5/CSS3)

L'objectif est de créer un squelette robuste et sémantique avant d'y injecter de l'interactivité.

- **Définition du Layout :** Mise en place du header pour le titre et le sous-titre, ainsi que d'un panneau latéral dédié à la recherche d'année.
- **Conteneur du Calendrier :** Création de la grille principale qui accueillera les jours générés dynamiquement.
- **Design Responsive :** Utilisation de classes CSS pour assurer que l'interface s'adapte aux mobiles, notamment en masquant certains textes longs au profit d'icônes sur petits écrans.

## 2. Moteur de Calcul Temporel (JavaScript)

C'est ici que l'application définit sa "vérité" historique.

- **Algorithme des Années Bissextilles :** Implémentation d'une logique différenciée selon l'époque :
  - *Période Julienne (-46 à 1582) :* Une année est bissextile tous les 4 ans.
  - *Période Grégorienne (Post-1582) :* Une année est bissextile si elle est divisible par 4, mais pas par 100, sauf si elle est divisible par 400.
- **Gestion des Bornes Temporelles :** Configuration de l'intervalle de fonctionnement allant de l'an **-46** (début du calendrier julien) jusqu'à l'an **9999**.

## 3. Logique de la Réforme Grégorienne

Le défi majeur de ce jalon est de gérer visuellement la suppression de jours lors du passage du calendrier julien au grégorien en octobre 1582.

- **Détection des jours sautés :** Crédit à la fonction `isDaySkipped(year, month, day)`.
- **Traitements Visuels :** Les jours "fantômes" (sautés) reçoivent une classe CSS spécifique `skipped-day` qui applique un effet barré (`line-through`) et une opacité réduite.

- **Information Contextuelle** : Affichage automatique du type de calendrier actif (Julien, Grégorien ou Transition) selon l'année saisie par l'utilisateur.

#### 4. État Global de l'Application

Initialisation des variables de contrôle dans script.js pour maintenir la cohérence de l'affichage :

- currentDate : Date système actuelle pour le marquage du jour courant.
  - selectedYear : L'année actuellement visualisée par l'utilisateur.
  - currentTheme : Initialisé par défaut sur "light" ou récupéré depuis le stockage local.
- 

## Jalon 2 : Spécialisation

Le **Jalon 2** marque le passage d'un calendrier générique à un outil spécialisé dans la liturgie chrétienne. C'est ici que l'intelligence métier est injectée pour transformer des dates en moments de célébration:

### 1. Constitution de la Base de Données des Saints

L'application doit disposer d'une référence fixe pour chaque jour de l'année.

- **Tableau saintsOfTheYear** : Création d'un objet JavaScript contenant 366 entrées (clé: "MM-JJ", valeur: "Nom du Saint").
- **Exemples d'entrées** : Intégration de dates clés comme le 1er janvier pour Sainte Marie ou le 2 février pour la Chandeleur.
- **Fonction d'accès** : Développement de getSaintOfDay(month, day) pour extraire le saint correspondant à n'importe quelle cellule du calendrier.

### 2. Algorithmes des Fêtes Mobiles

Contrairement aux saints, certaines fêtes changent de date chaque année (fêtes liées au cycle de Pâques).

- **Calcul de Pâques** : Implémentation de l'algorithme (souvent basé sur le cycle lunaire et l'équinoxe de printemps) pour déterminer la date de Pâques pour l'année sélectionnée.
- **Déduction des fêtes liées** : Utilisation de la date de Pâques comme point de référence pour calculer automatiquement l'Ascension (Pâques + 39 jours) ou la Pentecôte (Pâques + 49 jours).
- **Stockage temporaire** : Les dates calculées sont injectées dans la variable d'état currentHolidays lors du changement d'année.

### **3. Logique de Rendu Visuel (Rendering Engine)**

La fonction renderCalendar() devient le cœur de l'affichage en fusionnant dates, saints et fêtes.

- **Boucle de génération** : Le script parcourt tous les jours du mois sélectionné et crée un élément HTML .calendar-day pour chacun.
- **Typage des célébrations** : Attribution de classes CSS spécifiques selon l'importance de l'événement détecté:
  - .holiday-major : Pour les fêtes religieuses majeures.
  - .holiday-mobile : Pour les fêtes dont la date varie.
- **Identification du jour courant** : Application de la classe isToday si la date générée correspond à la date système de l'utilisateur.

### **4. Interface de Consultation (Modales)**

Une simple cellule de calendrier ne suffisant pas à afficher toutes les informations, un système de détail est mis en place.

- **Déclenchement au clic** : Chaque jour devient cliquable pour ouvrir la #holidayModal.
  - **Contenu riche** : La modale est remplie dynamiquement avec le titre de la fête (#modalTitle), sa date formatée (#modalDate) et une description textuelle (#modalDescription).
  - **Action externe** : Intégration du bouton #addToGoogleCalendar pour permettre l'exportation de l'événement vers l'agenda de l'utilisateur
- 

## **Jalon 3 : UX/UI**

Transformation de l'outil de calcul en une application web moderne, ergonomique et personnalisable.

Quatre piliers de développement du jalon :

### **1. Système de Thèmes Dynamiques (Mode Clair/Sombre)**

L'application intègre une gestion avancée de l'apparence pour s'adapter aux préférences de l'utilisateur :

- **Logique de bascule (toggleTheme)** : Implémentation d'une fonction qui change les variables CSS globales (couleurs de fond, de texte et de bordures) sans recharger la page.

- **Persistante des données** : Utilisation du localStorage pour mémoriser le choix du thème (clair ou sombre) afin que l'utilisateur retrouve sa configuration lors de sa prochaine visite.
- **Interactivité visuelle** : Changement dynamique de l'icône de contrôle (passage d'une lune à un soleil) et application de transitions douces sur les éléments de l'interface comme les panneaux et les boutons.

## 2. Ergonomie et Navigation Avancée

L'objectif est de rendre la navigation intuitive, peu importe l'année ou l'appareil utilisé:

- **Recherche d'année arbitraire** : Mise en place d'un champ numérique (#yearInput) permettant d'accéder instantanément à n'importe quelle année entre -46 et 9999.
- **Interface Responsive** : Optimisation du design pour les petits écrans, notamment en masquant le texte des boutons de don pour ne conserver que l'icône, libérant ainsi de l'espace visuel.
- **États visuels interactifs** : Gestion des états de survol (hover) et des états désactivés pour les boutons de navigation, avec des effets d'agrandissement (scale-105) et d'ombres portées.

## 3. Widget de Temps Réel

Pour renforcer l'aspect "utilitaire quotidien" du calendrier, un module d'horloge est ajouté :

- **Horloge fixe** : Intégration d'un widget discret positionné en bas à droite de l'écran (#LiveTime).
- **Mise à jour dynamique** : Utilisation d'un script pour afficher l'heure en temps réel, offrant une information immédiate à l'utilisateur sans quitter l'interface du calendrier.

## 4. Feedback Utilisateur et Modales Spécialisées

L'interaction est enrichie par des retours visuels clairs :

- **Notifications "Toast"** : Développement d'un système de messages temporaires apparaissant en haut de l'écran pour confirmer des actions ou donner des informations rapides.
- **Modales de soutien et d'information** : Création de fenêtres dédiées pour le système de don (#donateModal) et les mentions légales, utilisant des animations d'entrée fluides (modal-enter).

## Jalon 4 : Raffinement

Transformer l'application en un produit fini, accessible à l'international et doté d'une personnalité unique grâce à des détails interactifs.

### 1. Système d'Internationalisation (i18n)

L'application passe d'une interface statique à un système multilingue dynamique.

- **Moteur de traduction** : Implémentation de la fonction `t(key)` qui récupère les chaînes de caractères depuis un dictionnaire chargé en mémoire.
- **Chargement asynchrone** : Utilisation de `fetch` pour charger les fichiers JSON (`fr.json`, `en.json`, `es.json`, etc.) de manière asynchrone au démarrage de l'application via `loadAllTranslations()`.
- **Sélecteur de langue** : Mise en place d'un menu déroulant interactif permettant de changer la langue de l'interface (`currentLang`) sans rechargement de la page.
- **Gestion des erreurs (Fallback)** : Si une traduction est manquante dans la langue sélectionnée, le système bascule automatiquement sur la version française par défaut.

### 2. Notifications et Système "Toast"

Pour améliorer la communication avec l'utilisateur, un système de retour visuel est intégré.

- **Composant showNotification** : Création dynamique d'éléments `div` avec la classe `.notification-toast`.
- **Animations CSS** : Utilisation de transitions `transform` pour faire apparaître le message depuis le haut de l'écran et le faire disparaître après 3 secondes.

### 3. "Easter Eggs" et Animations Festives

Cette étape ajoute une couche de ludisme à l'expérience utilisateur lors de dates symboliques.

- **Détection de dates spéciales** : La fonction `checkEasterEgg(year, month, day)` surveille les dates affichées.
- **Déclencheurs (Triggers) :**
  - **Noël** : Active l'événement "christmas" le 25 décembre.
  - **Anniversaires spécifiques** : Des animations comme des feux d'artifice ("fireworks"), des ballons ("balloons") ou des haltères ("dumbbells") se

déclenchent pour les anniversaires des contributeurs du projet (ex: Antoine Bianconi, Alexandre Guillier).

- **Particules temporaires** : La fonction triggerEasterEgg génère des éléments visuels éphémères qui s'auto-détruisent après quelques secondes.

#### 4. Conformité, Don et Finalisation du Code

Le projet est préparé pour sa mise en ligne publique.

- **Modale de Don** : Intégration du bouton donate-btn avec une modale configurable (showDonateModal) dont le lien de redirection varie selon la langue sélectionnée.
- **Mentions Légales & RGPD** : Création d'une structure modale dynamique (showLegalNotice) pour afficher les informations de conformité basées sur les traductions JSON.
- **Licence et Open Source** : Application de la licence MIT et préparation du dépôt pour les contributions externes (instructions de clone, de fork et de pull request).