CPE Lyon - 3ETI - 2025-2026

Techniques et Langages du Web

Projet “Agence de voyages”

Le sujet ci-dessous présente le travail attendu lors des sept séances de projet. Certains termes pourront vous paraître obscurs lors des premières séances ; ils deviendront clairs au fur et à mesure que nous avancerons dans le cours. Il est cependant fortement conseillé de lire le sujet dans son intégralité. Les seuls langages autorisés sont ceux vus en cours. En particulier, l’utilisation de frameworks (Boostrap, Angular...) n’est pas permise.

I Cahier des charges

Le but de ce projet est de réaliser un site web qui vous a été commandé par une **agence de voyages**. La MOA (maîtrise d’ouvrage) a exprimé un certain nombre d’exigences fonctionnelles *minimales* (libre à vous de faire preuve d’imagination et d’ajouter des fonctionnalités !). Portez-y la plus grande attention, car **c’est le respect de ces exigences qui déterminera en grande partie votre note** (le client est roi !) :

1. Le site doit comporter **au minimum** les pages suivantes :
   - une ***page principale*** qui présente les différentes destinations en photo, **disposées selon une grille**
   - une page de ***Réservation***
   - une page de ***Contenu du panier***
   - une page de ***Confirmation de commande***
   - une page ***A propos & Contact***

2. Le choix d’une destination sur la page principale (**en cliquant sur une photo**) doit amener à la page de réservation. **Il ne doit bien sûr y avoir qu’une seule page de réservation, commune à toutes les destinations proposées !** Celle-ci reprend l’intitulé du voyage sélectionné, et contient un formulaire de réservation. Lorsque ce formulaire a été **complété**, un bouton de validation ajoute la réservation le voyage au panier.

3. Le formulaire de réservation doit comporter les éléments suivants :
   - date de départ*
   - date de retour*
   - nombre d’adultes*
   - nombre d’enfants (-12 ans)*
   - une case à cocher “Petit-déjeuner ?”
   - une zone affichant le prix calculé
   - un bouton de soumission et un bouton de remise à zéro
   - d’autres éléments que vous jugeriez utiles

   ⚠ Les éléments marqués d’une * doivent obligatoirement être remplis par l’utilsateur avant soumission du formulaire.

4. Le prix est calculé **automatiquement** en fonction de la durée du séjour, du nombre d’adultes, du nombres d’enfants et du petit déjeuner ; un enfant paie 40% du prix d’un adulte, quel que soit le séjour choisi. Un petit déjeuner ajoute un supplément de 15€ par personne et par jour. Evidemment, la date de retour doit obligatoirement être postérieure à la date de départ. **Les enfants ne peuvent voyager**

sans être accompagnés d'un adulte. Toute modification dans le formulaire conduit à un recalcul automatique du prix.

5. De base, le modèle de données des séjours est très simple : chaque séjour est caractérisé par un titre et un prix par adulte et par jour. Vous pourrez cependant enrichir librement ce modèle si vous le souhaitez (en ajoutant le prix du vol en avion, des prestations optionnelles comme des visites guidées, etc.).

6. La page **Panier** permet de visualiser le ou les voyages réservés (oui, on s'adresse à une clientèle de luxe !) depuis la page de réservation. Elle comporte également un **formulaire de finalisation de la commande**, avec les coordonnées de l'utilisateur (identité, adresse postale, adresse mail...).

7. La page de contact doit comporter les éléments suivants :
   - adresse (postale) de l'agence de voyages
   - numéro de téléphone ; un clic sur ce numéro doit ouvrir automatiquement l'application associée à la composition d'appels téléphoniques si l'utilisateur en dispose (par exemple Skype sur PC, ou le téléphone sur smartphone)
   - bouton d'envoi d'un mail ; l'objet du mail doit être prérempli avec « Demande de renseignements »et le corps du message doit commencer par « Bonjour, je souhaiterais obtenir des renseignements sur ».

8. Sur toutes les pages on doit retrouver :
   - un menu de navigation
   - un pied de page

   Naturellement, vous veillerez à **ne pas dupliquer** de code entre toutes les pages !

9. Sur chaque photo de destination figurant sur la page principale, la température actuelle de ladite destination doit être affichée. Pour cela, vous utiliserez l'API proposée par le site web **OpenWeatherMap**.

10. **Filtrage** : sur la page d'accueil de votre site, proposant les différentes destination, l'utilisateur doit disposer de **filtres** (prix min / max ; disponibilité en fonction de la date ; animaux acceptés ou non ; petit déjeuner proposé, etc.). **Les destinations ne correspondant pas aux filtres doivent être automatiquement masquées, sans clic sur un bouton de validation des filtres**.

    On n'exige pas de fonction de **tri** (par exemple par prix croissant).

Eléments optionnels :

11. En dehors des éléments précédents, qui sont imposés, l'agence de voyages vous laisse le champ libre pour le style. Il est évident qu'une page HTML nue, sans style, serait largement insuffisante. Laissez-donc libre cours à votre imagination, habillez votre site. Vous pouvez ajouter des effets (par exemple au survol des destinations), mais conservez un site agréable à visiter. Nous n'attendons pas un site de qualité professionnelle ; même si l'aspect esthétique du site sera pris en compte, **nous serons surtout attentifs sur le respect du cahier des charges, ainsi que sur la qualité et la clarté de votre code** !

12. **Authentification** : l'utilisateur peut s'authentifier sur le site, à l'aide d'un formulaire simple (nom d'utilisateur et mot de passe).
    Comme vous ne disposez pas encore de base de données ni de serveur, vous simulerez une petite base de données d'utilisateurs (3 ou 4 comptes suffiront) permettant d'accepter ou rejeter une connexion.

13. **Historique de commandes** : le site peut être doté d'une fonctionnalité "Historique des commandes" permettant à un utilisateur de consulter ses commandes passées.

14. Le site ne doit pas être nécessairement *responsive*, c'est-à-dire s'adapter à la taille de l'écran sur lequel il est consulté (en particulier sur les écrans de petite taille type smartphone). Cependant, une attention toute particulière sera portée au respect des normes d'*accessibilité*.

II Organisation du projet

Vous travaillerez en **binôme**. Le projet démarre le **9 octobre** et est à rendre pour le **vendredi 20 décembre à 23h59** (heure de fermeture automatique du dépôt). **Aucun rendu ne sera possible après cette date**, et le code qui sera évalué sera celui présent sur le dépôt à ce moment-là.

Chaque binôme dispose d'un dépôt sur Github (ce qui implique que chaque membre dispose d'un compte GitHub, personnel ou CPE). Vous devez commencer par créer une équipe *via* le lien de votre groupe disponible sur la page e-campus du cours ; **pour des raisons pratiques, cette équipe doit impérativement être désignée par les deux noms de famille des membres du binôme, sous la forme `NOM1-NOM2`**.

L'idée est donc de mettre l'accent sur le travail d'équipe, la collaboration, et le partage de code à l'aide de **Git** : chaque membre du binôme travaille de son côté sur une fonctionnalité précise **en accord avec l'autre membre**, puis **pousse** son code sur le dépôt partagé pour la mise en commun (*fusion*).

Attention à bien gérer votre temps et votre collaboration, ce sont deux composantes fondamentales de la gestion et de la réussite d'un projet !

III Préparation de l'environnement de développement

Pour le développement de votre site, nous vous conseillons d'utiliser l'éditeur de texte **Visual Studio Code** de Microsoft (ne pas confondre avec *Visual Studio*, qui est un environnement de développement beaucoup plus complet), disponible gratuitement, multi-plateformes et qui fournit une interface graphique pour GitHub. VS Code dispose d'un *Marketplace* dans lequel vous pourrez trouver des centaines d'extensions pour différents langages de programmation. Nous vous conseillons d'installer les extensions suivantes :

- `French Language Pack for VS Code`, si vous souhaitez disposer de l'interface en français ;
- `W3C Web Validator`, qui vous indique les différentes erreurs contenues dans votre code HTML ;
- `Live Server` ; cette extension vous sera utile lorsque utiliserez des fonctionnalités qui nécessitent un serveur. Pour l'utiliser, ouvrez avec VS Code un **dossier** contenant un fichier HTML, et un bouton `Go Live` apparaît dans la barre d'état (de VS Code). Un clic sur ce bouton ouvre une fenêtre de navigateur, qui s'actualisera automatiquement dès que vous modifierez **et enregistrerez** un fichier du dossier ;
- `Git Graph`, `Git History` et `Git Lens` qui facilitent grandement l'utilisation de Git avec VS Code.

Si vous souhaitez utiliser votre propre machine, il vous faudra installer également le logiciel `Git` (https://git-scm.com/) si vous ne l'avez pas déjà.

Les postes Linux de l'école sont tous déjà équipés de ces logiciels (sauf éventuellement certaines des extensions pour Visual Studio Code présentées ci-dessus).

Enfin, il est également vivement conseillé d'installer l'extension `Web Developer` pour Chrome ou Firefox, qui propose de nombreux outils d'aide au développement Web, comme des *validateurs* de code.

IV Démarrage du projet

Commencez par créer un *jeton d'accès personnel* (Personnal Access Token) sur GitHub, comme indiqué sur cette page. Il vous suffira ensuite de *clôner votre dépôt GitHub* (troisième icône de la barre latérale, dans VS Code).

V Etapes du projet

Cette section vous donne le travail attendu à la fin de chaque séance, ce qui doit vous permettre d'évaluer votre avancement et éventuellement votre retard.

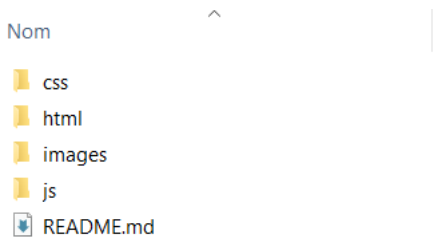
Séance 1 : HTML - Conception, structure et contenu du site (1/2)

Pour cette première séance, commencez par mettre en application la méthodologie de conception présentée en cours. En particulier, identifiez les différentes tâches et répartissez-vous le travail équitablement dans le binôme. Une fois que vous aurez une bonne compréhension du sujet et du résultat attendu, passez au prototypage (soit papier, soit avec des outils en ligne comme `https://wireflow.co/`, `https://mockflow.com/`, `draw.io`...).

Une fois ce travail réalisé, rassemblez les ressources (photos, icones...) ; vous pourrez trouver des photos gratuites sur `pixabay.com` ou `unsplash.com`, et des icones sur `thenounproject.com` ou `fontawesome.com`.
Rappel : n'utilisez que des illustrations libres de droit !

Inutile de partir sur un site proposant 50 destinations ! 6 ou 8 destinations suffiront largement pour couvrir les besoins du projet.

Créez ensuite l'arborescence de votre projet : cela fait partie des bonnes pratiques de créer des dossiers séparés pour les fichiers HTML, les feuilles de style CSS, les scripts JavaScript, les ressources (images, icônes, polices de caractère…) :



Enfin seulement, passez au code HTML des quatres pages imposées, et faites en sorte que la navigation soit fonctionnelle. Certains éléments (comme le contenu du panier) seront générés plus tard, dynamiquement ; ce n'est pas grave, laissez-les vides pour l'instant ! De même, les en-tête et pied de page seront eux aussi ajoutés dynamiquement par la suite.

Tant que nous n'avons pas abordé la manière de stocker des données, saisissez vos textes "en dur", c'est-à-dire directement dans le code HTML. Il s'agit évidemment d'une pratique déconseillée, et nous verrons au fur et à mesure des cours de meilleures méthodes.

Important : consacrez cette séance à décrire la *structure* de vos pages, et pas encore leur mise en forme (ce qui peut-être très frustrant !!!), que l'on abordera plus tard.

A l'issue de cette séance

- Vous devez avoir une maquette de votre site, sous forme de "fil de fer" (c'est-à-dire, comment on page de chaque page à une autre) ;
- Votre projet doit êtr correctement organisé (un dossier `HTML`, un dossier `CSS`, etc.) ;
- Vous avez rassemblé toutes les ressources graphiques de votre site (photos, polices de caractères, etc.) ;
- vous veillerez à ce que les balises sémantiques les plus adéquates sont utilisées (par exemple pour les sections, les adresses, les champs de formulaire...) ;
- la navigation sur le site doit être opérationnelle (le passage correct d'une page à une autre).

Séance 2 : HTML (2/2) + CSS / Mise en forme, style

Durant cette séance, l'objectif est de mettre en forme vos pages afin de leur donner leur apparence finale. C'est donc durant cette séance que vous allez pouvoir placer les différents blocs sur les pages, définir les styles, polices de caractères, fonds d'écran, boutons... utilisés sur votre site. Bien sûr, il est important de vous coordonner au sein du binôme pour que toutes vos pages aient un style uniforme pour proposer une expérience utilisateur optimale.

En particulier, c'est lors de cette séance que vous devez mettre en place la grille des destinations présentées sur la page d'accueil (à l'aide de l'API *CSS Grid*).

N'hésitez pas à jeter un coup d'œil à ces ressources :

- vous trouverez sur `https://www.w3schools.com/howto/default.asp` de très nombreux tutoriels très simples (Section "How-to")
- les sites `https://html-css-js.com` et `https://grid.layoutit.com/` proposent de nombreux outils intéressants, comme des générateurs automatiques d'effets, de tableaux ou de dispositions (layouts)
- *Google Fonts* propose de nombreuses polices de caractères que vous pouvez utilisez facilement sur vos sites (voir `https://www.w3schools.com/howto/howto_google_fonts.asp`)

A l'issue de cette séance

- le code HTML de toutes vos pages doit être terminé ;
- tous les avertissements et erreurs signalés par le validateur doivent être corrigés ;
- vous devez avoir déterminé le style de votre site (polices de caractères, apparence des boutons, couleurs...) ;
- les destinations sur la page d'accueil doivent être disposées selon une grille à l'aide d'un `grid layout` comme présenté en cours ;
- les plus rapides peuvent également essayer de rendre leur site *responsive* pour qu'il soit consultable sur un smartphone.

Séance 3 : JavaScript (1/4) - Calculs, `getXXXByXXX`, jQuery

Vous allez enfin pouvoir ajouter un peu de dynamisme à votre site !

Web dynamique

Quand on parle de *pages dynamiques*, on ne parle pas d'"animations" sur la page, mais plutôt de pages web dont le contenu est généré automatiquement, et pour lequel la mise en page s'adapte automatiquement.

Normalement, cette étape est réalisée côté "backend", sur un *serveur web* ; cette partie n'étant pas au programme de première année, nous la réaliserons exceptionnellement côté "'frontend" ou "client", c'est-à-dire dans le navigateur de l'utilisateur.

Tout d'abord, travaillez sur la page de **réservation** : son contenu doit s'adapter en fonction de la destination choisie sur la page d'accueil (voir ci-dessous).

Comment savoir sur quelle destination l'utilisateur a cliqué ?

Pour retrouver, depuis la page de réservation, le séjour sur lequel le client a cliqué sur la page d'accueil, vous utiliserez la méthode présentée en cours :

- **sur la page principale**, pour chaque séjour, rajoutez dans l'adresse du lien vers la page de réservation un identifiant du séjour, sous la forme `id=identifiant` précédé par le caractère '**?**'. L'identifiant doit bien sûr correspondre à l'index correspondant dans votre tableau de destinations.
  Exemple : `<a href="reservation.html?id=7">`
- **sur la page Réservation**, vous pourrez alors récupérer l'identifiant contenu dans l'URL de la façon suivante :
  `let sejour_id = new URLSearchParams(window.location.search).get("id").`

Ensuite, passez au formulaire : la validation de celui-ci doit mener à la **page de contenu du panier** (page que vous alimenterez dynamiquement par la suite), mais seulement si tous les éléments du formulaire ont été correctement saisis (c'est-à-dire que la date de départ ne doit pas être postérieure à la date d'arrivée, qu'on ne peut pas choisir une date de départ antérieure à la date du jour, etc.). Sur la page de réservation, on doit également trouver **le prix total du voyage**, calculé **automatiquement** en fonction des durée et options choisies.

Enfin, à l'aide de **jQuery**, chargez dynamiquement, sur chacune de vos pages, l'en-tête et le pied de page.

A l'issue de cette séance

- la page de réservation doit identifier la destination sélectionnée sur la page d'accueil ;
- le formulaire de réservation doit être opérationnel (validation de la saisie, redirection vers la page panier…) ;
- le calcul du prix total du voyage doit tenir compte des différentes options, être automatique et fonctionnel ;
- toutes vos pages doivent charger dynamiquement l'en-tête et le pied de page.

Séance 4 : JavaScript (2/4) - Classes, objets, templates

Jusqu'ici, tous les contenus de votre site ont été écrits « en dur » dans les fichiers HTML. C'est évidemment une très mauvaise pratique car :

- les fichiers HTML devraient être réservés essentiellement à la description de la *structure* de votre site ;
- cela engendre des fichiers HTML longs et difficiles à maintenir ;
- à chaque fois que vous voudrez / devrez modifier le prix ou une illustration d'un séjour, vous devrez modifier le fichier HTML et le republier ;
- pour chaque destination, la structure du code HTML est la même que pour toutes les autres ! Seul le *contenu* change. On a donc du code répété, dupliqué ; c'est évidemment une très mauvaise chose, car si on décide de modifier la structure HTML pour *une* destination, il faut répéter ces modifications pour *toutes* les destinations (ce qui est synonyme de perte de temps, de manque d'efficacité, et source d'erreur (on peut oublier une modification à un endroit par exemple)).

L'idée est donc de rendre votre site web *dynamique* : le contenu des pages doit être lu et ajouté *dynamiquement*, quel que soit le nombre de destinations proposées. Dans l'idéal, vous devriez stocker toutes les informations relatives à vos destinations dans une **base de données**. L'utilisation d'une base de données nécessitant des connaissances qui dépassent le cadre de ce cours, nous allons, **dans un premier temps**, *simuler* une base de données en utilisant des tableaux d'objets JavaScript. Au fur et à mesure de l'avancée dans le cours, nous mettrons en place d'autres solutions.

Pour éviter la duplication du code HTML, vous alimenterez dynamiquement vos pages à l'aide de *templates*, comme vu en cours.

A l'issue de cette séance

- vous devez disposer d'une **classe** (au sens JavaScript !) `Destination`, avec les attributs caractérisant une destination, et les méthodes nécessaires ;
- d'un tableau d'instances / objets de cette classe `Destination` ;
- sur la page d'accueil, les destinations doivent être désormais créées *via* l'utilisation de `templates` HTML ;
- un clic sur une destination de la page d'accueil doit mener à la page de réservation, automatiquement adaptée pour la destination en question (le nom de la destination doit apparaître, et le prix de la réservation doit correspondre à **cette** destination).

Séance 5 : JavaScript (3/4) - DOM, filtres

L'objectif de cette séance est de mettre à profit vos connaissances sur le modèle DOM pour mettre en place différents filtres sur la page d'accueil ; à chaque modification des filtres, les destinations ne correspondant pas aux filtres doivent être masquées dynamiquement (i.e. sans avoir besoin de recharger la page). Les filtres peuvent être, par exemple :

- une fourchette de prix,
- si les enfants sont aurorisés,
- si les animaux sont autorisés,
- etc.

Rappel : aucun **tri** n'est exigé !

Par ailleurs, vous utiliserez également le DOM pour personnaliser la page de réservation en fonction de la destination sur laquelle l'utilisateur a cliquée (depuis la séance 4, vous disposez de toutes les informations nécessaires : d'une part, l'identifiant de la destination, récupérée depuis la *query string* ; d'autre part toutes les informations de cette destination, qui se trouvent dans l'instance correspondante de la classe JavaScript `Destination`).

A l'issue de cette séance

- vous devez disposer de différents filtres permettant de masquer dynamiquement (i.e. sans rechargement de la page) les destinations ne correspondant pas aux filtres sélectionnés. Un bouton permet de réinitialiser les filtres et de faire réapparaître toutes les destinations ;
- la page de réservation doit être personnalisée dynamiquement en fonction de la destination cliquée sur la page d'accueil.

Séance 6 : JavaScript (4/4) - Stockage, Panier, JSON, API

Cette séance comporte trois objectifs :

1. de conserver une trace des réservations effectuées précédemment, afin de pouvoir les afficher lorsque l'utilisateur en effectue une nouvelle ou clique sur le panier ;
2. le requêtage d'une API externe (`openweathermap`), dans le but de récupérer les températures en temps réel des destinations proposées, et de les afficher sur la page d'accueil ;
3. l'externalisation de toutes les données concernant les destinations proposées ; autrement dit, toutes les données des destinations proposées doivent être insérées dans un fichier JSON (qui fait office de base de données), et ne plus apparaître directement dans vos fichiers JavaScript. L'idée est alors de commencer par lire ce fichier pour alimenter les pages du site.

Pour cela, toutes les informations concernant la réservation (dates, nombre d'adultes et d'enfants, prise du petit-déjeuner ou non, etc.) doivent être gérées à l'aide d'un objet JavaScript ; cet objet JavaScript est

ensuite converti en chaîne de caractères au format JSON (cf. méthode `JSON.stringify` vue en cours) ; enfin, cette chaîne peut être conservée dans le *stockage local* du navigateur (cf. API Storage). Des boutons ou icônes doivent permettre à l'utilisateur de retirer un élément du panier (ce qui a pour effet de supprimer l'objet JSON correspondant du *stockage local*).

Pour l'appel d'API externes, ou la lecture d'un fichier JSON, le principe est identique : vous devrez utiliser la méthode `fetch` vue en cours.

⚠ **Rappel** : l'utilisation simple de la méthode `fetch` produit une *promesse* JavaScript et conduit à du code asynchrone (autrement dit, si vous effectuez plusieurs requêtes pour obtenir les différentes températures, **rien ne garantit que les réponses arriveront dans le même ordre !** Pour contourner cette difficulté, pensez à utiliser les mots-clés `async` et `await` présentés en cours.

A l'issue de cette séance

Ca y est ! Votre site doit être complètement opérationnel !