



**POLYTECHNIQUE
MONTRÉAL**

LE GÉNIE
EN PREMIÈRE CLASSE

LOG4420 – Conception de sites web dynam. et transact.

Travail pratique 3

Chargé de laboratoire:

Antoine Béland

Automne 2017

Département de génie informatique et de génie logiciel

1 Objectifs

Le but de ce travail pratique est de vous familiariser avec le langage JavaScript et avec la bibliothèque [jQuery](#).

Plus particulièrement, vous aurez à mettre en place plusieurs interactions sur différentes pages du site web afin que celui-ci devienne dynamique et fonctionnel. À la fin de ce travail, le site web sera utilisable, mais aucune mesure de sécurité ne sera présente puisque l'ensemble des opérations seront réalisées du côté client.

2 Introduction

Lors des deux premiers travaux pratiques, vous avez mis en place la structure et la mise en forme du site web à réaliser. Cependant, comme vous l'avez sans doute remarqué, celui-ci était complètement statique. En effet, il n'était pas possible d'interagir avec les formulaires ou d'afficher une liste de produits correspondant au critère de tri et à la catégorie sélectionnés, par exemple.

Afin d'être en mesure d'ajouter des interactions du côté client, le langage JavaScript doit être utilisé. En ce sens, ce langage permet entre autres de manipuler les éléments du [DOM](#) (*Document Object Model*), d'avoir accès aux API HTML (p. ex. *Geolocation*, *Local Storage*, etc.) et de réaliser des requêtes asynchrones ([AJAX](#)) pour récupérer des ressources ou pour communiquer avec des services web.

3 Travail à réaliser

À partir du code que vous avez produit lors des travaux pratiques précédents, vous aurez à ajouter la couche logique du site web grâce au langage JavaScript. Dans le cas où votre travail pratique 2 n'aurait pas été bien réussi, vous pouvez utiliser le corrigé du TP2 qui se trouve sur [Moodle](#) afin de partir du bon pied pour ce troisième travail.



Avertissement

Afin de promouvoir la compréhension du langage JavaScript, seules les bibliothèques [jQuery](#) et [jQuery Validation](#) sont permises pour ce travail pratique.

Avant de débiter, assurez-vous d'avoir récupéré l'archive associée à ce travail pratique sur Moodle. Cette archive contient des fichiers supplémentaires que vous devez inclure dans votre projet. Pour ce faire, copiez l'ensemble des fichiers de l'archive dans votre dossier de projet.

3.1 Prise en charge de Node.js

Au cours de ce travail pratique ainsi que des prochains, vous aurez à utiliser [Node.js](#). Il s'agit d'un environnement multiplateforme qui permet l'exécution d'applications JavaScript. Node.js est extrêmement populaire en raison de sa légèreté et de son efficacité. En effet, l'environnement est basé sur un modèle d'événements asynchrones permettant d'éviter les blocages et les attentes (p. ex. entrées/sorties) tout en étant conçu pour créer des applications extensibles (*scalables*)¹.

L'environnement Node.js est déjà installé sur les postes du laboratoire à Polytechnique. Vous pouvez consulter ce [lien](#) si vous souhaitez l'installer sur votre machine.

Node.js est également livré avec le gestionnaire de paquets [npm](#) (*Node Package Manager*). Ce gestionnaire de paquet est extrêmement utile puisqu'il répertorie plusieurs centaines de milliers de paquets pouvant être utilisés pour le développement d'applications de toutes sortes. Par ailleurs, ce gestionnaire de paquets peut facilement être utilisé via le terminal. En effet, les commandes sont invoquées en utilisant le mot `npm` suivi du nom de la commande. Vous utiliserez principalement les commandes [install](#), [uninstall](#) et [start](#).

Toutes les applications Node.js qui utilisent le gestionnaire de paquets npm doivent posséder le fichier [package.json](#) à leurs racines. Ce fichier indique plusieurs informations telles que le nom, les auteurs ainsi que les dépendances qui sont utilisés par l'application.

En ce qui concerne ce travail pratique, le fichier `package.json` vous est fourni avec toutes les dépendances nécessaires pour le bon fonctionnement de l'application web. En d'autres mots, vous n'avez pas besoin d'installer de nouvelles dépendances pour ce travail. Cependant, vous devez installer les dépendances nécessaires en tapant la commande suivante à la racine de votre projet :

```
npm install
```

1. Pour en savoir plus sur les particularités de Node.js, vous pouvez consulter ce [lien](#).

Cette commande installera toutes les dépendances dans un nouveau dossier appelé « `node_modules` » à la racine de votre projet.

3.2 Mise en place d'un serveur web

La première étape à réaliser pour ce travail est de mettre en place un serveur web. Puisque plusieurs requêtes asynchrones devront être effectuées pour récupérer la liste des produits (`data/products.json`), un serveur web doit obligatoirement être mis en place puisqu'il n'est pas possible d'effectuer des requêtes de ce type vers des fichiers locaux (`file:///`). En ce sens, seules les requêtes HTTP sont autorisées pour récupérer ce genre de fichiers. Pour en savoir plus, vous pouvez consulter cette [réponse](#) sur Stack Overflow.

Une solution simple est d'utiliser le paquet [gulp-connect](#) qui est disponible sur npm. En effet, ce serveur web demande très peu de configuration et est très facile à utiliser. Ce paquet a déjà été inclus dans les dépendances de l'application web et a déjà été configuré. Pour démarrer le serveur, vous devez taper la commande suivante :

```
npm start
```

Une fois la commande entrée dans la console, le serveur web démarrera à l'adresse suivante : <http://localhost:8000>.

3.3 Éléments à réaliser

Maintenant que le serveur web est fonctionnel, la prochaine étape est d'implémenter les éléments nécessaires pour que le site soit utilisable. Avant de débiter, assurez-vous d'inclure la bibliothèque jQuery. En ce sens, vous devez inclure la ligne suivante sur toutes les pages du site web :

```
<script type="text/javascript" src="./assets/js/jquery-3.2.1.min.js"></script>
```

L'utilisation de jQuery est fortement recommandée pour ce travail pratique. En effet, cette bibliothèque simplifie la manipulation des éléments du DOM tout en fournissant des méthodes utiles pour la réalisation de requêtes AJAX.

Vous aurez besoin de sauvegarder les données du panier d'achats et celles des commandes afin de pouvoir les utiliser de page en page. Pour ce faire, vous devez utiliser l'API [Local Storage](#) de HTML5. Plusieurs sites offrent des tutoriels simples pour l'utilisation du *Local*

Storage, dont [celui-ci](#). Vous pouvez visualiser l'état courant du *Local Storage* avec l'outil de débogage de votre navigateur web (raccourci F12) sous l'onglet « Application » (pour Google Chrome).

Les sous-sections qui suivent décrivent les éléments à réaliser pour les différentes parties du site web. Il est à noter que le ou les fichiers qui contiendront votre code JavaScript doivent se trouver dans le dossier `assets/scripts`. Par ailleurs, pour vous aider dans l'organisation de votre code JavaScript, l'annexe A de ce document expose plusieurs éléments qui pourraient vous être utiles.



Notez bien

Assurez-vous de respecter les identifiants (`id="..."`) et les classes (`class="..."`) qui vous sont imposés. En effet, les éléments qui doivent posséder des identifiants particuliers seront utilisés par les tests automatisés qui devront être exécutés sur le site web (voir la section §EXÉCUTION DE TESTS D'ACCEPTATION AUTOMATISÉS). Par ailleurs, le format monétaire à utiliser pour les prix doit respecter les contraintes suivantes : le séparateur décimal est la virgule (« , ») et deux nombres décimaux doivent être présents pour chacun des prix.

3.3.1 Entête

1. Lorsqu'aucun élément est présent dans le panier d'achats, le badge indiquant le nombre de produits se trouvant dans le panier (`.shopping-cart > .count`) ne doit pas être visible. La figure 2a se trouvant en annexe B illustre ce cas.
2. Lorsqu'un ou plusieurs produits sont présents dans le panier, le badge indiquant le nombre d'items qui se trouvent dans le panier doit être visible et celui-ci doit indiquer le bon nombre de produits en tout temps, et ce, sur toutes les pages du site web. La figure 2b (voir annexe B) montre l'affichage attendu lorsqu'un item se trouve dans le panier.

3.3.2 Page des produits (`products.html`)

1. La liste des produits (`id="products-list"`) doit correspondre aux produits listés dans le fichier « `products.json` ». Pour ce faire, vous devez effectuer une requête AJAX pour récupérer le fichier JSON et vous devez par la suite ajouter les informations récupérées à la liste. Assurez-vous que la liste est triée par prix croissant, c'est-à-dire le critère « Prix (bas-haut) » (critère par défaut). L'affichage attendu est illustré à la figure 3 (voir annexe B).

2. Le groupe de boutons associé aux catégories des produits (`id="product-categories"`) doit être fonctionnel. Ainsi, lorsqu'un bouton du groupe est cliqué, la catégorie associée à ce bouton doit être appliquée à la liste de produits. En ce sens, seuls les produits liés à la catégorie sélectionnée devront être visibles dans la liste des produits. Également, lorsqu'un bouton est cliqué, ce même bouton doit devenir sélectionné (`class="selected"`). Toutes ces opérations doivent être réalisées sans rafraîchissement de page. La figure 4 se trouvant en annexe B illustre d'ailleurs l'affichage souhaité lorsque la catégorie « Appareil photo » est sélectionnée.
3. Le groupe de boutons liés au classement des produits (`id="product-criteria"`) devrait avoir un comportement similaire au groupe de boutons relié aux catégories de produits. En effet, lorsqu'un bouton du groupe est cliqué, la liste de produits affichée devra être triée selon le critère de tri sélectionné. De plus, lorsqu'un bouton est cliqué, celui-ci doit devenir sélectionné (`class="selected"`). Toutes ces opérations doivent se faire sans rafraîchissement de page. La figure 5 (voir annexe B) montre l'affichage attendu lorsque la catégorie « Consoles » et le critère de tri « Nom (A-Z) » sont sélectionnés.
4. Le texte associé au nombre de produits qui sont actuellement affichés dans la liste de produits (`id="products-count"`) doit être mis à jour à chaque fois qu'une nouvelle catégorie est sélectionnée. En ce sens, ce texte doit indiquer en tout temps le bon compte de produits.
5. Lorsqu'un produit de la liste est cliqué, le lien pour accéder à ce produit devrait être de la forme suivante : « `./product.html?id=#productId` ». Cette forme correspond à un paramètre qui est passé avec l'URL. Dans ce cas-ci, le paramètre est « `id` », et celui-ci est égal à l'identifiant du produit (`#productId`).

3.3.3 Page d'un produit (`product.html`)

1. À partir du paramètre spécifié par l'URL nommé « `id` », la page d'un produit devrait afficher les informations associées au produit ayant le même numéro d'identifiant qui a été spécifié. Ainsi, le nom du produit (`id="product-name"`), l'image (`id="product-image"`), la description (`id="product-desc"`), les caractéristiques (`id="product-features"`) ainsi que le prix (`id="product-price"`) devront être mis à jour afin d'afficher les bonnes informations. Pour en savoir plus concernant la récupération d'un paramètre d'URL avec JavaScript, veuillez consulter ce [lien](#). La figure 6 (voir annexe B) montre l'affichage attendu pour le produit « Console Wii ».

2. Dans le cas où un identifiant invalide a été spécifié via le paramètre «`id`» de l'URL, le contenu principal de la page doit être vide. Uniquement un titre doit être présent et ce dernier doit indiquer «Page non trouvée!». La figure 7 se trouvant en annexe B illustre l'affichage souhaité lorsqu'un identifiant invalide est spécifié.
3. Le formulaire permettant d'ajouter le produit au panier (`id="add-to-cart-form"`) doit être fonctionnel. Ainsi, il doit être possible de spécifier la quantité à ajouter au panier pour le produit courant. De plus, lorsque le bouton «Ajouter» est cliqué, les informations du formulaire doivent être utilisées pour ajouter le produit au panier. Cela doit se faire sans aucun rafraîchissement de page.
4. Lorsque le produit est ajouté au panier, une boîte de dialogue (`id="dialog"`) doit faire son apparition en bas de la page afin de confirmer que le produit a été ajouté au panier. Cette notification doit être affichée pendant cinq secondes. Également, il faut que la quantité associée aux nombres d'items dans le panier d'achats (`.shopping-cart > .count`) soit mise à jour dès qu'un produit est ajouté. La figure 8 (voir annexe B) montre l'affichage souhaité lorsqu'un produit est ajouté au panier.

3.3.4 Page du panier d'achats (`shopping-cart.html`)

1. Lorsqu'aucun élément ne se trouve dans le panier d'achats, seuls le titre de la page ainsi qu'un paragraphe indiquant «Aucun produit dans le panier.» doivent être présents. La figure 9 qui se trouve en annexe B montre l'affichage attendu lorsque le panier est vide.
2. Lorsqu'un ou plusieurs produits se trouvent dans le panier, ceux-ci doivent être listés en ordre alphabétique dans le tableau. Également, le prix de chacun des produits (`class="price"`) ainsi que le total (`id="total-amount"`) doivent être cohérents en fonction de la quantité de chacun des items. La figure 10 (voir annexe B) montre l'affichage souhaité lorsque le panier contient des items.
3. Le bouton «X» (`class="remove-item-button"`) qui est associé à un item particulier permet de supprimer l'item qui lui est associé dans le panier d'achats. Lorsqu'un bouton «X» est cliqué, une fenêtre de type `confirm` doit faire son apparition pour indiquer à l'utilisateur que l'item associé s'apprête à être supprimé (p. ex. «Voulez-vous supprimer le produit du panier?»). Lorsque l'utilisateur confirme la suppression de l'item, ce dernier est supprimé du panier. Ainsi, la ligne du tableau correspondant à cet item doit être supprimée de la page et le total associé aux items dans le panier doit être mis à jour. De plus, dans le cas où cette suppression ferait en sorte que le panier devient

vide, la page doit indiquer qu'il n'y a plus de produits. Toutes ces opérations doivent se faire sans rafraîchissement de page. La figure 11 qui se trouve en annexe B illustre un exemple de boîte de dialogue qui doit être affichée avant la suppression d'un item dans le panier.

4. Le bouton « - » (`class="remove-quantity-button"`) permet de diminuer d'un la quantité associée à un item particulier du panier. Lorsque la quantité d'un item est d'un, le bouton lié à ce même item doit être désactivé (voir la propriété `disabled`). Lorsque la quantité associée à un item est supérieure à un, le bouton de cet item doit être actif. Lorsque le bouton est cliqué, la quantité de l'item (`class="quantity"`) doit être décrémentée d'un. De plus, le prix ainsi que le total doivent être mis à jour pour refléter les changements. Cela doit se faire sans rafraîchissement de page.
5. Le bouton « + » (`class="add-quantity-button"`) sert à augmenter d'un la quantité pour item du panier d'achats. Lorsque le bouton est cliqué, la quantité de l'item (`class="quantity"`) doit être ajustée. Également, le prix et le total doivent être actualisés. Le tout doit être fait sans rafraîchissement de page.
6. Lorsque le bouton qui permet de vider le panier (`id="remove-all-items-button"`) est cliqué, une fenêtre de confirmation (`confirm`) doit faire son apparition pour indiquer à l'utilisateur que tous les items du panier s'apprêtent à être supprimés (p. ex. « Voulez-vous supprimer tous les produits du panier ? »). Une fois que l'utilisateur a confirmé la suppression des éléments, les items doivent être supprimés et la page doit indiquer qu'aucun élément n'est présent dans le panier. Cela doit se faire sans rafraîchissement de page.

3.3.5 Page de commande (`order.html`)

Il est obligatoire d'utiliser la bibliothèque jQuery.validate pour effectuer la validation du formulaire. En ce sens, assurez-vous d'inclure les lignes de code suivantes sur la page de confirmation afin que la bibliothèque soit fonctionnelle :

```
<script type="text/javascript" src="./assets/js/jquery.validate.min.js"></script>
<script type="text/javascript" src="./assets/js/additional-methods.min.js"></script>
<script type="text/javascript" src="./assets/js/messages_fr.js"></script>
```

1. Le formulaire de commande (`id="order-form"`) peut être envoyé uniquement lorsque tous les champs sont complets et validés.

2. Les champs du formulaire doivent avoir les contraintes suivantes :
 - Le prénom (`id="first-name"`) et le nom (`id="last-name"`) sont obligatoires et doivent avoir une longueur minimale de deux caractères ;
 - L'adresse courriel (`id="email"`) est obligatoire et doit être valide (p. ex. antoine.-beland@polymtl.ca) ;
 - Le numéro de téléphone (`id="phone"`) est obligatoire et doit être un numéro de téléphone canadien valide (`phoneUs`) ;
 - Le numéro de carte de crédit (`id="credit-card"`) est obligatoire et doit être valide (p. ex. 4111 1111 1111 1111 [numéro de carte Visa]) ;
 - La date d'expiration de la carte de crédit (`id="credit-card-expiry"`) est obligatoire et doit respecter le format mm/aa. Les valeurs possibles pour le mois (mm) sont 01 à 12. Celles possibles pour l'année (aa) sont 00 à 99. Puisqu'aucune méthode n'existe pour effectuer une validation d'une carte de crédit dans le format demandé avec `jQuery.validate`, vous devez écrire une [méthode de validation personnalisée](#) pour valider cette date. Il est conseillé d'utiliser une [expression régulière](#) (*RegExp*) pour réaliser cette validation. Le message d'erreur à inscrire si la date est invalide est : « La date d'expiration de votre carte de crédit est invalide. ».
3. Lorsque le formulaire est soumis et qu'un ou plusieurs champs ne sont pas valides, un message d'erreur doit être affiché en dessous de chacun des champs invalides. Ces messages d'erreurs doivent provenir de la bibliothèque `jQuery.validate`. De plus, cela doit se faire sans rafraîchissement de page. La figure 12 qui se trouve en annexe B montre un formulaire soumis contenant plusieurs erreurs.
4. Lorsque le formulaire est soumis et que toutes les informations des champs sont valides, les items présents dans le panier d'achats doivent être retirés et une commande doit être créée. Pour les besoins de ce travail pratique, assurez-vous de conserver uniquement le nom (prénom et nom) du client ainsi que le numéro de commande (numéro unique correspondant au nombre de commandes déjà effectuées [p. ex. la deuxième commande qui a été effectuée sera la commande #2]).

3.3.6 Page de confirmation (`confirmation.html`)

1. Le numéro de commande (`id="confirmation-number"`) ainsi que le nom du client (`id="name"`) doivent être affichés sur la page. La figure 13 (voir annexe B) montre un exemple de page de confirmation qui est attendue.

3.4 Exécution de tests d'acceptation automatisés

Afin de valider l'ensemble des éléments à réaliser pour ce travail pratique, des tests d'acceptation automatisés (*End-to-End Tests*) vous ont été fournis afin que vous puissiez vérifier votre travail. Tous les tests ont été réalisés avec [Nightwatch.js](#). Il est à noter que ces tests serviront à l'évaluation de votre travail. En ce sens, il est important que l'ensemble des tests fonctionnent **avant** que vous remettiez votre travail pratique.

Avant d'exécuter les tests, assurez-vous que le serveur web est en fonction. Pour exécuter les tests, veuillez entrer la commande suivante dans un terminal à la racine du projet :

```
npm run e2e
```

L'environnement de test est actuellement configuré pour fonctionner dans le laboratoire de Polytechnique (système d'exploitation Linux). Si vous souhaitez exécuter les tests sur un autre environnement, vous pouvez jeter un coup d'œil à ce [lien](#) (voir fichier «`nightwatch.json`» dans le dossier «`tests`»).



Avertissement

Il est important de n'apporter **aucune** modification aux fichiers dans le dossier `tests` (sauf «`nightwatch.json`»). En effet, ces fichiers contiennent tous les tests automatisés qui sont exécutés sur le site web et se doivent de ne pas être modifiés. Assurez-vous donc de respecter les identifiants et les classes imposés afin que l'ensemble des tests puissent fonctionner.



Conseils pour la réalisation du travail pratique

1. Utiliser le [mode strict](#) dans vos scripts pour faciliter le débogage de votre code.
 2. Jetez un coup d'œil à l'annexe A pour vous donner des idées en ce qui a trait à l'organisation de votre code.
 3. Utilisez le [débogueur](#) du navigateur pour vous aider à identifier les erreurs.
 4. N'attendez pas à la dernière minute pour commencer le laboratoire! Le débogage en JavaScript peut parfois s'avérer long et ardu.
 5. Soyez consistant dans vos conventions de codage (voir [guide de codage de Mark Otto](#)).
-

4 Remise

Voici les consignes à suivre pour la remise de ce travail pratique :

1. Vous devez placer le code de votre projet dans un dossier compressé au format ZIP nommé « TP3_matricule1_matricule2.zip ». Assurez-vous d'exclure les dossiers « node_modules » et « tests » avant de remettre votre travail.
2. Vous devrez également créer un fichier nommé « temps.txt » à l'intérieur du dossier de votre projet. Vous indiquerez le temps passé au total pour ce travail.
3. Le travail pratique doit être remis avant **23h55**, le **2 novembre 2017** sur Moodle.

Aucun retard ne sera accepté pour la remise de ce travail. En cas de retard, le travail se verra attribuer la note de zéro. Également, si les consignes 1 et 2 concernant la remise ne sont pas respectées, une pénalité de -5% est applicable.

Le navigateur web **Google Chrome** sera utilisé pour tester votre site web.

5 Évaluation

Globalement, vous serez évalué sur le respect des exigences des éléments à réaliser. Plus précisément, le barème de correction est le suivant :

Exigences	Points
Respect des exigences pour les éléments à réaliser	
Entête	1
Page des produits	3,5
Page d'un produit	2,5
Page du panier d'achats	4
Page de commande	3,5
Page de confirmation	0,5
Code JavaScript	
Structure du code	3
Qualité et clarté du code	2
Total	20

L'évaluation se fera en grande partie grâce aux tests d'acceptation automatisés qui vous sont fournis. En effet, les tests automatisés permettront de valider les exigences pour les éléments à réaliser.

Ce travail pratique a une pondération de **9%** sur la note du cours.

6 Questions

Si vous avez des interrogations concernant ce travail pratique, vous pouvez poser vos questions sur le canal [#tp3](#) sur Slack. N'hésitez pas à poser vos questions sur ce canal afin qu'elles puissent également profiter aux autres étudiants. De plus, vous pouvez rejoindre le chargé de laboratoire sur Slack ([@antoinebeland](#)) si vous souhaitez lui poser une question en privé.

Annexes

A Organiser son code en JavaScript

L'organisation du code en JavaScript peut parfois être difficile, particulièrement pour ceux qui débutent avec ce langage. En effet, contrairement aux langages orientés objet traditionnels, JavaScript possède une syntaxe très permissive et n'impose pas une structure particulière pour son fonctionnement. C'est donc au programmeur de se discipliner pour réaliser du code clair, maintenable et réutilisable. Afin de vous aider à prendre en charge ce langage, cette annexe expose différentes astuces qui vous permettront de mieux organiser votre code en JavaScript.

A.1 Séparer la logique des données

Le premier aspect à considérer est la séparation de la logique permettant la manipulation des vues de celle qui manipule les données. En ce sens, il est important de définir plusieurs modules afin de s'assurer que chacun d'entre eux possède une seule responsabilité principale.

Plusieurs modèles existent afin de séparer les différents éléments d'une application (p. ex. Modèle MVC [Modèle-vue-contrôleur]). Un modèle possible qui permet la séparation des différentes responsabilités est d'ailleurs illustré à la figure 1. Ce modèle expose trois composants principaux : les services, les contrôleurs et les vues. Les services sont responsables de récupérer les données et de les organiser afin qu'elles puissent être utilisées par les contrôleurs. Les contrôleurs, à partir des données fournies par les services, mettent à jour les vues et récupèrent les données en entrée (formulaires, etc.) afin de les traiter. Les vues, elles, correspondent à l'interface graphique de l'application web.

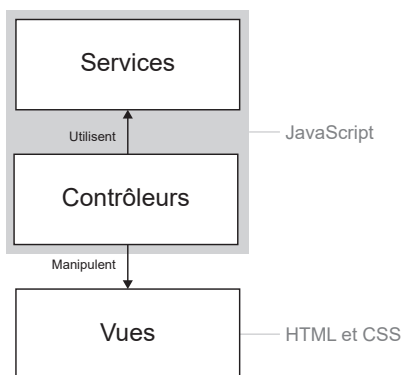


FIGURE 1 – Modèle possible pour la séparation des responsabilités

A.2 Structurer votre code en composants

À partir des différents éléments qui ont été identifiés, la deuxième étape est de créer des composants en JavaScript. Il existe plusieurs manières de définir des entités en JavaScript. Cette section s'intéressera particulièrement à deux manières simples de créer des composants.

La façon la plus simple de créer une entité en JavaScript est de définir un **objet littéral**. Cet objet permet la définition de variables et de fonctions ayant une visibilité publique. Par souci de clarté, un exemple de code est illustré ci-dessous. Pour en savoir plus sur ce type de définition, vous pouvez consulter ce [lien](#).

<pre>// ***** Objet littéral ***** var example = { variable: 1, function1: function() { // ... }, function2: function() { // ... } };</pre>	<pre>// ***** Exemples d'utilisation ***** example.variable; // Sortie: 1 example.function1(); // Exécution de la fonction 1 example.function2(); // Exécution de la fonction 2</pre>
---	--

Une autre façon de structurer son code en JavaScript est d'utiliser le **patron module**. Ce patron permet la définition de variables et de fonctions ayant une visibilité privée ou publique. Un exemple de code est d'ailleurs illustré ci-dessous pour mettre en évidence un cas d'utilisation de ce patron. Pour en savoir plus sur le patron module, vous pouvez consulter ce [tutoriel](#).

<pre>// ***** Patron module ***** var example = (function() { // Variable privée var privateVariable = 1; // Fonction privée function _privateFunction() { /* ... */ } // Fonctions publiques return { publicFunction1: function() { _privateFunction(); privateVariable = 2; }, publicFunction2: function() { return privateVariable; } }; })();</pre>	<pre>// ***** Exemples d'utilisation ***** example.privateVariable; // undefined (variable privée) example._privateFunction(); // Erreur! Fonction non définie (privée) example.publicFunction1(); // Exécution de la fonction 1 example.publicFunction2(); // Sortie: 2</pre>
---	--

A.3 Ressources supplémentaires

Il existe plusieurs ressources que vous pouvez consulter pour vous aider à structurer votre code. Voici donc quelques liens qui pourraient vous être utiles pour la réalisation de ce travail pratique :

Organisation du code avec jQuery	https://learn.jquery.com/code-organization/concepts/
Utilisation d'espaces de noms	http://falola.developpez.com/tutoriels/javascript/namespace/

B Captures d'écran



(a) Panier vide



(b) Panier avec un item

FIGURE 2 – Badge indiquant le nombre de produits dans le panier

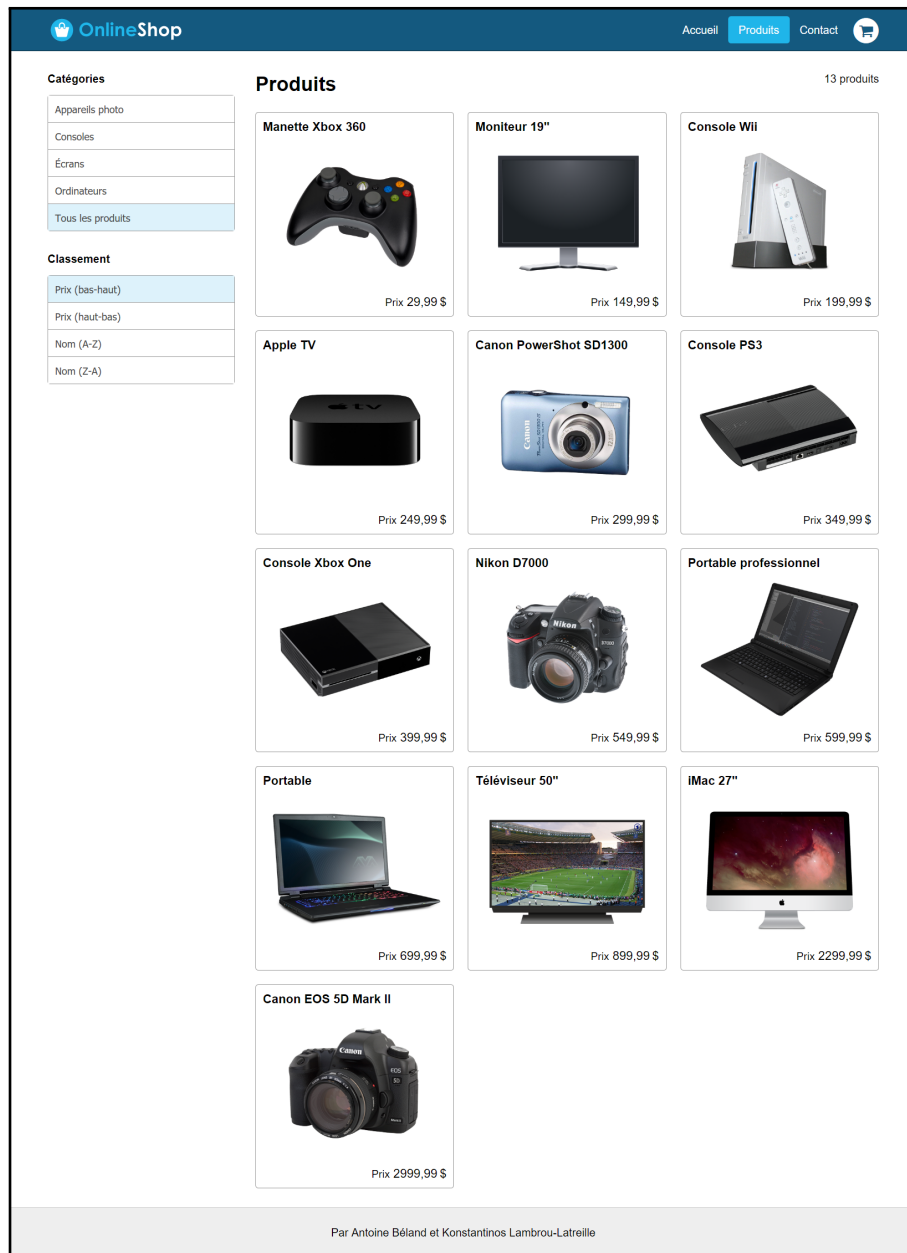


FIGURE 3 – Affichage par défaut de la page des produits

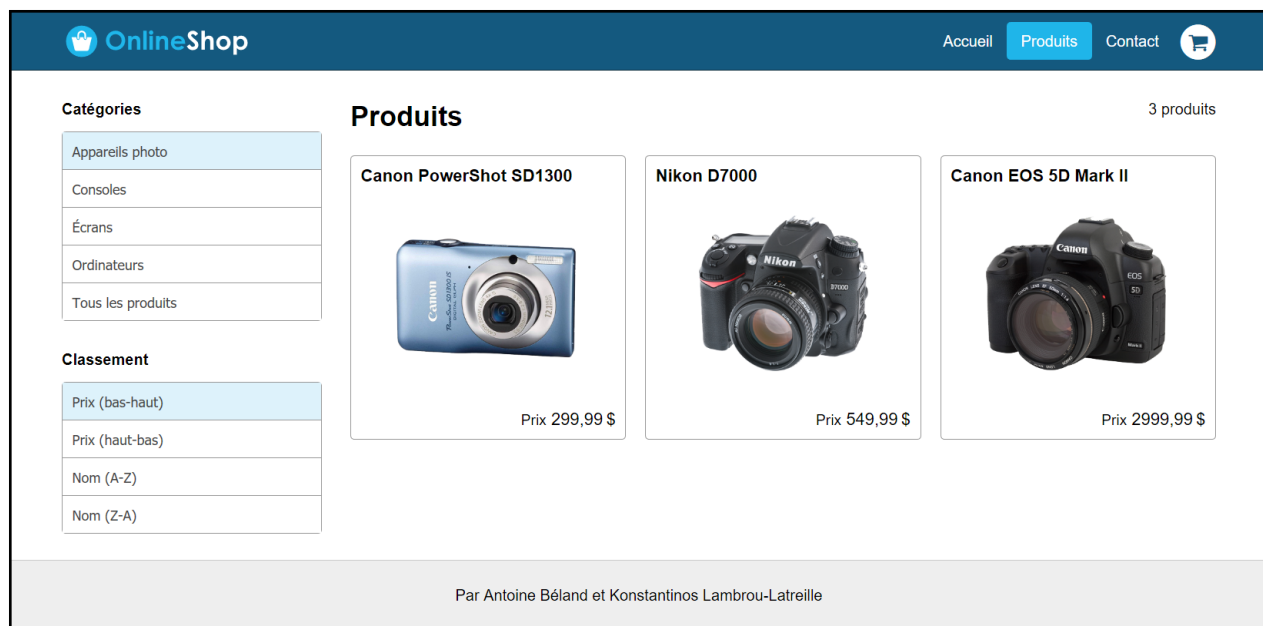


FIGURE 4 – Affichage de la catégorie « Appareil photo »

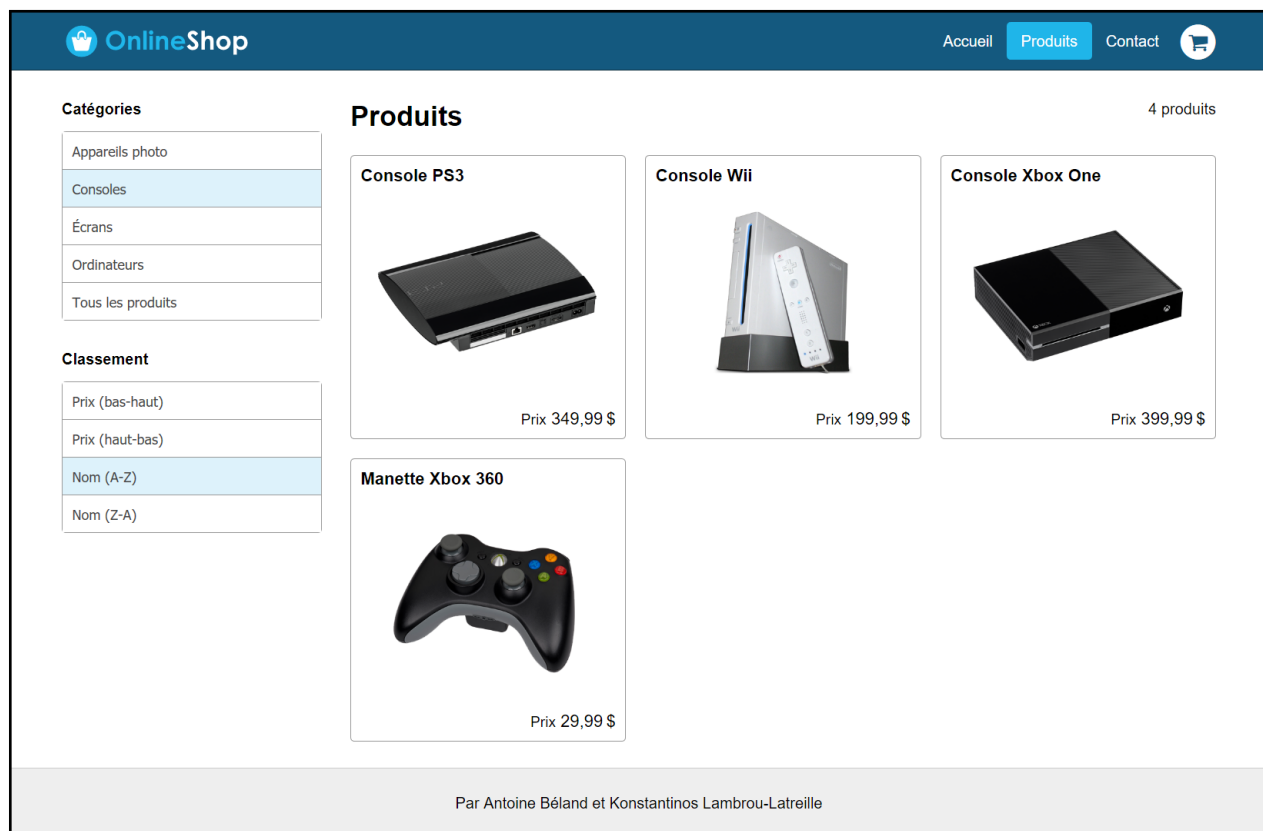


FIGURE 5 – Affichage de la catégorie « Consoles » avec le critère de tri « Nom (A-Z) »


OnlineShop

Accueil
Produits
Contact


Console Wii



Description

La Wii est une console de jeux vidéo de salon du fabricant japonais Nintendo. Console de la septième génération, tout comme la Xbox 360 et la PlayStation 3 avec lesquelles elle est en rivalité, elle est la console de salon la plus vendue de sa génération et a comme particularité d'utiliser un accéléromètre capable de détecter la position, l'orientation et les mouvements dans l'espace de la manette. ([Wikipédia](#))

Caractéristiques

- Microprocesseur Unit Broadway Power PC
- Processeur graphique (AMD) ATI Hollywood cadencé à 243 MHz
- Mémoire de 24 Mo de 1TSRAM + 64 Mo de 1TSRAM de MoSys Technology
- Lecteur DVD Panasonic Matsushita Electronic
- 2 Prises USB 2.0, Sortie Vidéo/Audio YUV et Sortie RVB


Prix: 199,99 \$


Quantité:  



Par Antoine Béland et Konstantinos Lambrou-Latreille

FIGURE 6 – Affichage attendu pour la page d'un produit


OnlineShop

Accueil
Produits
Contact


Page non trouvée!

Par Antoine Béland et Konstantinos Lambrou-Latreille

FIGURE 7 – Affichage désiré lorsqu'un identifiant de produit invalide est spécifié



[Accueil](#)
[Produits](#)
[Contact](#)



Console Wii



Description

La Wii est une console de jeux vidéo de salon du fabricant japonais Nintendo. Console de la septième génération, tout comme la Xbox 360 et la PlayStation 3 avec lesquelles elle est en rivalité, elle est la console de salon la plus vendue de sa génération et a comme particularité d'utiliser un accéléromètre capable de détecter la position, l'orientation et les mouvements dans l'espace de la manette. ([Wikipédia](#))

Caractéristiques


- Microprocesseur Unit Broadway Power PC
- Processeur graphique (AMD) ATI Hollywood cadencé à 243 MHz
- Mémoire de 24 Mo de 1TSRAM + 64 Mo de 1TSRAM de MoSys Technology
- Lecteur DVD Panasonic Matsushita Electronic
- 2 Prises USB 2.0, Sortie Vidéo/Audio YUV et Sortie RVB

Le produit a été ajouté au panier.


Quantité:   [Ajouter](#)

Par Antoine Béland et Konstantinos Lambrou-Latreille

FIGURE 8 – Affichage souhaité lorsqu'un produit est ajouté dans le panier



[Accueil](#)
[Produits](#)
[Contact](#)



Panier

Aucun produit dans le panier.

Par Antoine Béland et Konstantinos Lambrou-Latreille

FIGURE 9 – Affichage attendu lorsque le panier est vide

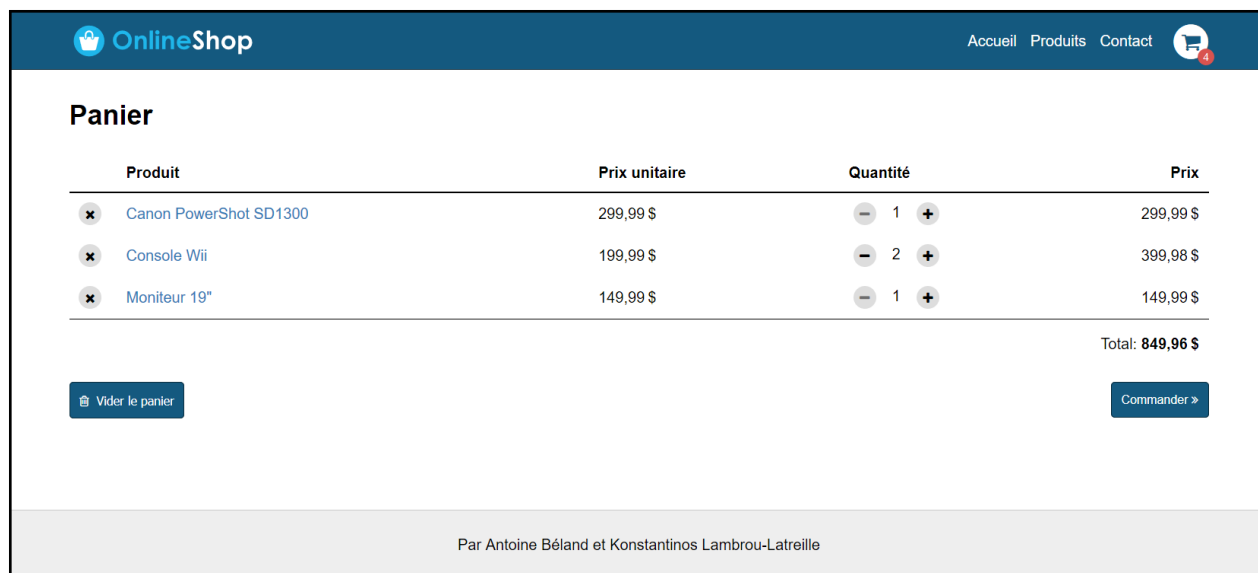


FIGURE 10 – Affichage souhaité lorsque des produits se trouvent dans le panier

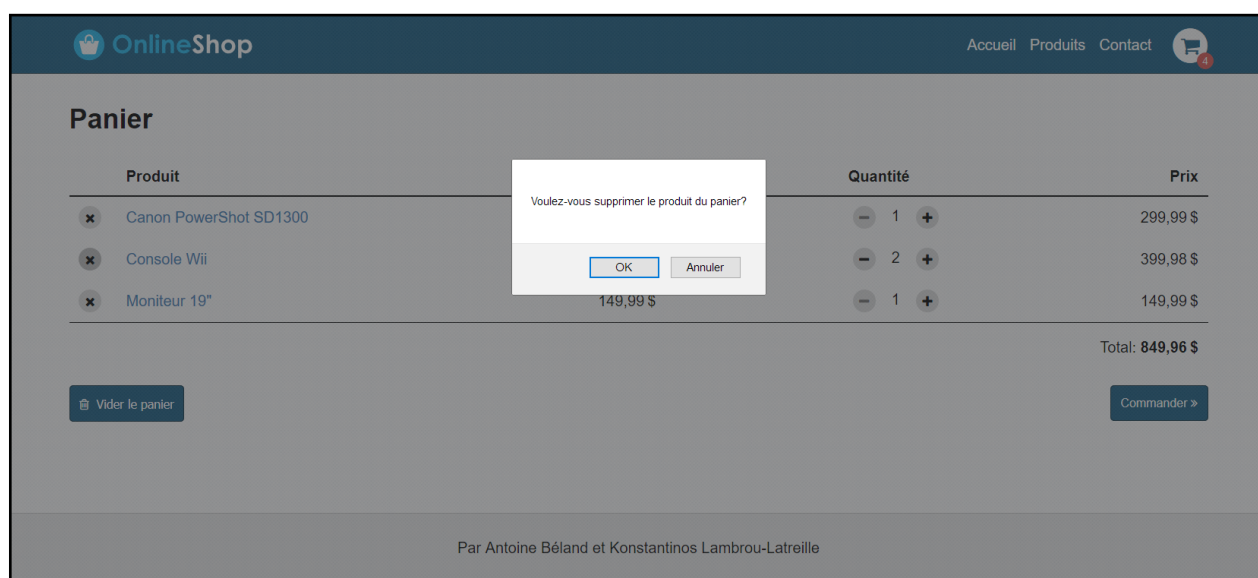


FIGURE 11 – Exemple de boîte de dialogue avant la suppression d'un item dans le panier

OnlineShop Accueil Produits Contact

Commande

Contact

Prénom: Antoine

Nom: B
Veuillez fournir au moins 2 caractères.

Adresse courriel: Ce champ est obligatoire.

Téléphone: Ce champ est obligatoire.

Paieement

Numéro de carte de crédit: Ce champ est obligatoire.

Expiration (mm/aa): 12
La date d'expiration de votre carte de crédit est invalide.

Payer >

Par Antoine Béland et Konstantinos Lambrou-Latreille

FIGURE 12 – Formulaire de la page de commande contenant des erreurs

OnlineShop Accueil Produits Contact

Votre commande est confirmée Antoine Béland!

Votre numéro de confirmation est le 00001.

Par Antoine Béland et Konstantinos Lambrou-Latreille

FIGURE 13 – Page de confirmation lorsque le formulaire soumis est valide