DOCUMENTATION DU CODE DU CHATBOT

VARIABLES GLOBALES

COMPTHIST (EVENTS.JS)

Résumé : Indice de position dans l'historique.
Commentaire : La position -1 représente le fait qu'aucune précédente entrée n'est affichée dans la barre.

DEMOMODE (EVENTS.JS)

Résumé : Indique si appuyer sur les flèches « Haut » et « Bas » nous fait nous déplacer dans l'historique ou dans le corpus d'exemples prévu pour les démonstrations.
Commentaire : True pour mode démo, false pour mode historique.

CTRL|ONE|TWO|THREE|FOUR (EVENTS.JS)

Résumé : Chaque variable représente si la touche du clavier du même nom est pressée ou non.
Commentaire : Variables utilisées pour les raccourcis multi-touches (ex : Ctrl+1 pour remplacer F1).

PREDIC (PREDIC.JS)

Résumé : Objet javascript contenant la structure de données utilisée pour l'auto complétion et la prédiction de texte et quelques fonctions associées à cette dernière.
Commentaire : La structure est un JSON constitué de trois branches : une première contenant tous les mots de la structure, le dictionnaire, une deuxième contenant les scores et une dernières contenant les suivants.

PREDIC1|PREDIC2|PREDIC3 (PREDIC.JS)

Résumé : Ces variables contiennent les valeurs affichées dans les boutons de prédiction associés.
Commentaire : RAS

CORPUS (PREDIC.JS)

Résumé : Tableau contenant quelques phrases de base pour remplir la structure d'auto complétion à sa première utilisation.
Commentaire : Il s'agit des questions isolées pour diriger mon stage.

PREVIEW (PREVIEW.JS)

Résumé : Variable contenant l'élément du DOM représentant la fenêtre de prévisualisation.
Commentaire : RAS

ALADIN (ALADIN.JS)

Résumé : Dernier widget Aladin-Lite affiché dans la zone de chat.
Commentaire : RAS

LASTLOOKED (ALADIN.JS)

Résumé : Nom du dernier objet astronomique affiché dans le widget Aladin-Lite.
Commentaire : Utile pour recentrer le widget avant d'en ajouter un autre.

OTYPE|CATALOG_NAME|CATALOG_DESC|MEAS|OID (CATALOGUES.JS)

Résumé : Paramètres retournés par le traitement du langage naturel lors d'une demande de catalogues.
Commentaire : Otype → Type de l'objet, Catalog_name → Nom du catalogue recherché, Catalog_Desc → Last ou Most Popular, Meas → Mesure(s) contenue(s) dans le catalogue recherché, Oid → Nom de l'objet mentionné dans le catalogue recherché.

RADIUSVIZIER (VIZIER.JS)

Résumé : Rayon utilisé pour la recherche sur VizieR.
Commentaire : Par défaut vaut 0.5.

AROUNDOTYPE (AROUND.JS)

Résumé : Type des objets recherchés lors du déclenchement de l'intention de recherche autour d'un objet précis.
Commentaire : RAS

DEMO (DEMO.JS)

Résumé : Tableau qui contient les exemples proposés lors du mode démo.
Commentaire : RAS

LASTERROR (ERROR.JS)

Résumé : Dernière erreur Javascript levée durant l'exécution du code.
Commentaire : RAS

LASTREQ (HISTORY.JS)

Résumé : Historique des derniers messages de l'utilisateur sous forme de tableau.
Commentaire : RAS

BDD (PARSER.JS)

Résumé : Services du CDS
Commentaire : Simbad, VizieR et Aladin.

SESSIONID (MAIN.JS)

Résumé : ID de session de l'utilisateur, pour pouvoir profiter d'une expérience personnelle.
Commentaire : RAS.

INTERFACE

Se trouvent dans ce dossier la totalité (en principe) des fonctions communiquant avec l'interface du chatbot. On trouvera dans cette partie les fonctions relatives à l'affichage en général. L'interface comporte deux chatbos l'un concernant l'interrogation des services du CDS et l'autre, un prototype (auquel on fait référence par chatbot FAQ)

DISPLAY.JS

Contient les fonctions liées à l'affichage sur la zone de chat et dans les bulles

DISPLAYUSERMESSAGE(MESSAGE)

Résumé : Affiche un message dans une bulle de l'utilisateur (bulles bleues).	
Entrée : {String} Message à afficher	**Sortie :** Aucune
Commentaire : RAS	

DISPLAYCHATBOTMESSAGE(MESSAGE)

Résumé : Affiche un message dans une bulle du chatbot (bulles noires).	
Entrée : {String} Message à afficher	**Sortie :** Aucune
Commentaire : RAS	

DISPLAYCHATBOTFAQMESSAGE(MESSAGE)

Résumé : Affiche un message dans une bulle du chatbot FAQ (bulles noires).	
Entrée : {String} Message à afficher	**Sortie :** Aucune
Commentaire : RAS	

DISPLAYWAITING()

Résumé : Affiche l'animation avec la bulle de chargement.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

DISPLAYPHOTOMETRIC(OID)

Résumé : Affiche dans une bulle le graphique « Photometric Points » présent sur le portail du CDS.	
Entrée : {String} Nom de l'objet sur lequel porte le graphique.	**Sortie :** Aucune
Commentaire : Peut mettre un certain temps à s'afficher.	

DESTROYWAIT()

Résumé : Retire la bulle de chargement de l'espace de chat.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

REFRESHSCROLLING()

Résumé : Descend le plus possible la barre de scroll du côté droit de la zone de chat.	
Entrée : Aucune	**Sortie :** Aucune

Commentaire : Attention aux contenus qui mettent un certain temps à s'afficher. La bulle est momentanément affichée vide, le scroll est effectué puis le contenu s'affiche et augmente la taille de la bulle. La barre de scroll n'est ainsi plus en bas de la zone. Pour contrer cela, il faut appeler de nouveau cette fonction après le chargement des données.

UPDATESEARCHBAR()

Résumé : Bloque/débloque la barre de recherche au début/à la fin du chargement et lance et supprime l'animation.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : Si la fonction est appelée alors que la barre est débloquée, celle-ci se bloque et l'animation de chargement commence, et vice versa.	

ADDLOGO(TOOL)

Résumé : Pour obtenir une balise HTML contenant le logo d'un des services du CDS	
Entrée : {String} Nom du service (Simbad, VizieR ou Aladin)	**Sortie :** {String} Balise HTML à intégrer au DOM pour afficher à l'écran le logo
Commentaire : RAS	

EVENTS.JS

Contient les fonctions qui s'activent lors d'événements déclenchés par l'utilisateur.

ON READY

Ce fichier js commence par un ensemble de déclarations d'événements à détecter au cours de l'utilisation de l'application. On trouve ici le code qui gère les événements suivants :

- Appui sur la touche « Entrée » : Envoi d'un message au chatbot.
- Déplacement de la souris : Si la souris n'est pas sur une zone proposant la preview, celle-ci n'est pas affichée (Indispensable pour gérer le bug de duplication de la zone de prévisualisation).
- Gestion des raccourcis clavier : F1 → Premier bouton prédiction, F2 → Deuxième bouton prédiction, F3→ Dernier bouton prédiction, F4 → Activation du mode démo, F8 → activation test unitaire
- Appui sur les touches « Haut » et « Bas » : Déplacement dans l'historique ou dans le mode démo.
- Redimensionnement de la fenêtre : Force le redimensionnement des widgets Aladin-Lite.

REFRESHPREVIEW()

Résumé : Définit qu'une balise possédant la classe « dialog » affiche une fenêtre de prévisualisation des données (récupérées depuis Simbad) qui suit la souris tant qu'elle la survole. Définit aussi que le clic de l'utilisateur sur une telle zone lance une recherche complète sur l'objet cliqué.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : Doit être appelée dès que des balises possédant la class « dialog » sont ajoutées, sinon les nouveaux éléments ne seront pas concernés par l'événement.	

REFRESHWEBPAGE(MESSAGE)

Résumé : Rafraichit la page quand l'utilisateur envoie le message « clear » au chatbot.	
Entrée : {String} Message envoyé par l'utilisateur	**Sortie :** Promesse Javascript puis undefined
Commentaire : La promesse Javascript est là uniquement pour étouffer les problèmes dus au côté asynchrone du langage.	

REFRESHACCORDION()

Résumé : Définit que les balises possédant la classe « accordion » sont des listes déroulantes jQuery-UI.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : Doit être appelée dès qu'un menu déroulant est ajouté à la zone de chat.	

HISTORYBEHAVIOR(EVENT)

Résumé : Décrit le comportement à exécuter pour la barre de recherche si l'utilisateur presse les touches « Haut » ou « Bas » et que le mode « Historique » est activé (demoMode à false).	
Entrée : {Event} Variable d'événement Javascript	**Sortie :** Aucune
Commentaire : Appuyer sur la touche « Haut » affiche une précédente requête plus ancienne, sur la touche « Bas » l'inverse.	

DEMOBEHAVIOR(EVENT)

Résumé : Décrit le comportement à exécuter pour la barre de recherche si l'utilisateur presse les touches « Haut » ou « Bas » et que le mode « Demo » est activé (demoMode à true).	
Entrée : {Event} Variable d'événement Javascript	**Sortie :** Aucune
Commentaire : Appuyer sur la touche « Haut » affiche la phrase d'exemple suivante.	

FORMAT.JS

Ce fichier contient les fonctions de formatage des données récupérées de Simbad pour les afficher dans un certain format prédéfini.

DEFAULTFORMAT(SIMANSWER)

Résumé : Affiche toutes les données présentes dans la structure JSON de retour de Simbad dans une liste déroulante. La fonction essaye également au-dessus de cette liste de créer une fiche synthèse à partir des données qu'elle reconnait (celles décrites dans les fonctions suivantes, à part les flux, la distance et la hiérarchie).	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad (voir fonction getInformation(result,meas)).	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Les informations dans la liste sont ordonnées par ordre alphabétique.	

PARALLAXFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives à la parallaxe récupérées de Simbad dans un format proche de celui proposé sur l'interface web du service.	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le format est le suivant → Valeur [Erreur] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

SPECTRALTYPEFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives au type spectral récupérées de Simbad dans un format proche de celui proposé sur l'interface web du service.	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le format est le suivant → Valeur [Erreur] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

COORDINATESFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives aux coordonnées récupérées de Simbad dans un format proche de celui proposé sur l'interface web du service.	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Un champ « Type » peut être manuellement ajouté à la structure de réponse Simbad avant l'appel à cette fonction pour préciser un format de coordonnées entre ICRS, J2000d ou Galactiques. Le format est le suivant → RA Declination [Erreur Erreur Angle] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

PROPERMOTIONFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives au mouvement propre récupérées de Simbad dans un format proche de celui proposé sur l'interface web du service.	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le format est le suivant → RA DEC [Erreur Erreur Angle] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

RADIALVELOCITYFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives à la vitesse radiale récupérées de Simbad dans un format proche de celui proposé sur l'interface web du service.	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le format est le suivant → TypeMesure Valeur [Erreur] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

TEMPFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives à la température récupérées de Simbad dans un format proche de celui proposé sur l'interface web du service	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le format est le suivant → TypeMesure Valeur [Erreur] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

MORPHOFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives à la morphologie récupérées de Simbad dans un format proche de celui proposé sur l'interface web du service.	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le format est le suivant → TypeMesure Valeur [Erreur] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

DIAMETERFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives au diamètre récupérées de Simbad dans un format proche de celui proposé sur l'interface web du service.	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le format est le suivant → TypeMesure Valeur [Erreur] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

ANGULARSIZEFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives à la taille angulaire récupérées de

Simbad dans un format proche de celui proposé sur l'interface web du service.	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le format est le suivant → TypeMesure Valeur [Erreur] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

RADIUSFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives au radius récupérées de Simbad dans un format proche de celui proposé sur l'interface web du service.	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le format est le suivant → TypeMesure Valeur [Erreur] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

PERIODFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives à la période récupérées de Simbad dans un format proche de celui proposé sur l'interface web du service.	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le format est le suivant → TypeMesure Valeur [Erreur] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

FLUXESFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives aux magnitudes récupérées de Simbad dans un format proche de celui proposé sur l'interface web du service.	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad. <Option> {String} Nom du flux à afficher. Par défaut tous.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le format est le suivant → Vignette Valeur [Erreur] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

DISTANCEFORMAT(SIMANSWER)

Résumé : Calcule la distance à partir de la parallaxe récupérée de Simbad et l'intègre dans un format proche de celui proposé sur l'interface web du service.

Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le message renvoie à la fois le résultat mais aussi les étapes de calcul.	

MAIN_IDFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives à l'id principal récupérées de Simbad dans un format proche de celui proposé sur l'interface web du service.	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le format est le suivant → TypeMesure Valeur [Erreur] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

IDSFORMAT(SIMANSWER)

Résumé : Organise les données brutes relatives aux ids récupérées de Simbad dans un format proche de celui proposé sur l'interface web du service.	
Entrée : {Array} Structure de la forme [Description, Valeur, Unité] créée à partir de la structure de retour d'une requête TAP à Simbad.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Le format est le suivant → TypeMesure Valeur [Erreur] Qualité Bibcode. Le bibcode est cliquable et renvoie vers le readme associé sur Simbad.	

HIERARCHYFORMAT(FAMILY, MODE, P, OID, OTYPE)

Résumé : Organise les données brutes récupérées de Simbad autour des liens de parenté (parents, enfants, frères) dans un format affichable sur une fenêtre de chat (certaines requêtes peuvent renvoyer plusieurs milliers de résultats).	
Entrée : {JSON} Structure récupérée à la suite de la fonction getParentsSIMBAD ou getChildrenSIMBAD. {String} « parents », « children » ou « siblings ». {Number} Nombre de fois où la fonction a été appelée récursivement. {String} Nom de l'objet dont on cherche les parentés. {String} Type d'objet utilisé pour filtrer les résultats.	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : Les 3 premiers parentés des résultats sont affichés dans la bulle, les autres sont cachées dans la liste déroulante. Chaque objet peut faire l'objet d'une prévisualisation quand la souris les survole.	

OBJECTLISTFORMAT(LIST)

Résumé : Organise les données brutes récupérées de Simbad depuis Aladin-Lite dans une liste déroulante.	
Entrée : {Array} Liste des objets astronomiques trouvés autour d'un autre objet grâce à Aladin-Lite	**Sortie :** {String} Données correctement formattées et prêtes à l'affichage.
Commentaire : RAS	

CHECKUNDEFINED(MEASURE)

Résumé : Remplace une valeur null, NaN ou undefined par le caractère '¤'.	
Entrée : {String} Valeur de la mesure	**Sortie :** {String} La valeur en entrée si non nulle, sinon '¤'
Commentaire : RAS	

NUMBERNOTATION(NUMBER)

Résumé : Transforme un nombre s'il possède plus de 3 chiffres après la virgule pour ne lui laisser que les 3 premiers.	
Entrée : {Number} Nombre à formatter	**Sortie :** {String} Nombre formatté
Commentaire : Le nombre de chiffres après la virgule peut être modifié facilement en modifiant la valeur de la variable « afterDot » de cette fonction.	

DRAW(COLOR)

Résumé : Dessine et colorie la petite vignette dans le canvas situé à gauche du type spectral comme défini dans « Format.js ».	
Entrée : {String} Couleur de la vignette au format hexadécimal	**Sortie :** Aucune
Commentaire : Ne dessine que les cercles du type spectral.	

PREDICT.JS

Ce fichier contient les fonctions relatives à l'auto complétion et à la prédiction de texte.

CLASS PREDICTIVEDICT

CONSTRUCTOR(CORPUS, MODE)

Résumé : Constructeur de la classe.	
Entrée : {Array} Tableau de phrases pour initialiser la structure. {String} Mode de création de la structure	**Sortie :** Aucune
Commentaire : Si mode = load, la structure est chargée depuis le local storage du navigateur. Sinon elle est créée à partir du corpus.	

BUILDDICT(CORPUS)

Résumé : Fonction qui construit à proprement parler la structure JSON de complétion/prédiction.	
Entrée : {Array} Tableau de phrases pour initialiser la structure.	**Sortie :** Aucune
Commentaire : RAS	

addSentence(sentence)

Résumé : Ajoute un message entré par l'utilisateur dans la structure.	
Entrée : {String} Message de l'utilisateur	**Sortie :** Aucune
Commentaire : Fonction à appeler dès que l'utilisateur envoie un message au chatbot.	

getNexts(word)

Résumé : Cherche les suivants du dernier mot entré par l'utilisateur selon la structure.	
Entrée : {String} Mot entré par l'utilisateur.	**Sortie :** {Array} Tableau contenant les solutions possibles.
Commentaire : Les solutions sont triées par score (plus populaire en premier).	

sortDict()

Résumé : Trie par ordre alphabétique le dictionnaire et ajuste la structure en fonction des modifications.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : La fonction de tri s'arrange pour qu'un mot soit toujours à la même position que son score et que ses suivants.	

autocomplete(word)

Résumé : Cherche dans le dictionnaire les mots pouvant correspondre à ce que tape l'utilisateur.	
Entrée : {String} Mot en cours d'écriture.	**Sortie :** {String} Tableau contenant les solutions possibles.
Commentaire : Les solutions sont triées par score (plus populaire en premier).	

sortByScore(res)

Résumé : Ordonne un tableau de solutions retournées par la structure selon le score de ses entrées.	
Entrée : {Array} Solutions possibles.	**Sortie :** {Array} Solutions possibles triées par score.
Commentaire : RAS	

save()

Résumé : Retourne une copie de la structure sous la forme d'une chaine de caractères.	
Entrée : Aucune	**Sortie :** {String} Copie de la structure
Commentaire : Une fois sous forme de chaîne, on peut enregistrer la structure dans le local storage.	

load(serialized)

Résumé : Construit la structure à partir d'une copie sous forme de chaîne de caractères.	
Entrée : {String} Copie de la structure	**Sortie :** Aucune
Commentaire : RAS	

isOk(struct)

Résumé : Vérifie que le JSON passé en paramètre possède une structure similaire à celle de la structure de base.	
Entrée : {JSON} Structure JSON à comparer	**Sortie :** {Boolean} True si la structure est correcte, false sinon.
Commentaire : RAS	

ON READY

Au chargement de la page, ce script essaye de récupérer une copie de la structure dans le local storage du navigateur. S'il en trouve une, il la charge, sinon il construit une nouvelle structure à partir du corpus de base défini dans la variable corpus. Il gère également deux événements liés à l'auto complétion/prédiction de texte, à savoir l'envoi à chaque caractère entré dans la barre de recherche du dernier mot tapé (ou en cours d'écriture), et le comportement des boutons de prédiction quand l'utilisateur appuie sur l'un d'entre eux.

USERWRITTING(MSG)

Résumé : Demande les suivants si l'utilisateur entre un espace (début d'un nouveau mot) ou une complétion s'il entre un caractère quelconque.	
Entrée : {String} Message en cours d'écriture dans la barre de recherche.	**Sortie :** Aucune
Commentaire : RAS	

DISPLAYPREDICTEDRESULTS(RES)

Résumé : Affiche dans les boutons de prédiction les trois premiers résultats récupérés de la structure.	
Entrée : {Array} Solutions retournées par la structure.	**Sortie :** Aucune
Commentaire : RAS	

BIGWORDS(BUTTON,TEXT)

Résumé : Modifie la taille du texte dans un bouton si celui-ci est trop grand et sort de la zone.	
Entrée : {DOMElement} Bouton où il faut modifier la taille du texte. {String} Texte affiché dans le bouton.	**Sortie :** Aucune
Commentaire : La taille de la police se réduit jusqu'à ce que la largeur du texte soit inférieure à celle du bouton.	

FIRSTWORD(RES)

Résumé : Ajoute une majuscule aux trois premières solutions retournées par la structure.	
Entrée : {Array} Solutions retournées.	**Sortie :** {Array} Le même tableau mais avec une majuscule au début des trois premières solutions.
Commentaire : Fonction utile lorsque le mot complété est le premier de la phrase.	

ADDSENTENCE(SENTENCE)

Résumé : Ajoute une phrase à la structure et lance une sauvegarde de celle-ci dans le local storage.

Entrée : {String} Message entré par l'utilisateur.	**Sortie :** Aucune
Commentaire : RAS	

PREVIEW.JS

Ce fichier contient les fonctions relatives à la prévisualisation proposée lors du survol de certains noms d'objets astronomiques.

HANDLERIN(CTX)

Résumé : Fait apparaître la fenêtre de prévisualisation.	
Entrée : {Objet} Balise survolée.	**Sortie :** {Promesse} Bloque l'exécution du Javascript tant que les données n'ont pas été récupérées de Simbad.
Commentaire : Comportement à effectuer lorsque la souris entre sur une zone offrant la prévisualisation (classe « dialog »).	

HANDLEROUT(CTX)

Résumé : Fait disparaitre la fenêtre de prévisualisation.	
Entrée : {Objet} Balise survolée.	**Sortie :** Aucune
Commentaire : Comportement à effectuer lorsque la souris sort d'une zone offrant la prévisualisation (classe « dialog »).	

UPDATEPOSITION(EVENT)

Résumé : Fait bouger la fenêtre de prévisualisation de sorte qu'elle suive le déplacement de la souris sur une zone offrant la prévisualisation.	
Entrée : {Event} Variable retournée lors du déclenchement de l'évènement.	**Sortie :** Aucune
Commentaire : Comportement à effectuer lorsque la souris se déplace sur une zone offrant la prévisualisation (classe « dialog »).	

CLICKPREVIEW(CTX)

Résumé : Lance une recherche exhaustive des informations récupérables sur Simbad à propos de l'objet cliqué.	
Entrée : {Objet} Balise survolée.	**Sortie :** Aucune
Commentaire : Comportement à effectuer lorsque la souris clique sur une zone offrant la prévisualisation (classe « dialog »).	

BUILDPREVIEW(CTX)

Résumé : Construit la structure de la prévisualisation à partir des résultats récupérés de Simbad.	
Entrée : {Objet} Balise survolée.	**Sortie :** {String} Texte à afficher dans la fenêtre de prévisualisation.
Commentaire : Récupère le nom de l'objet, son type, son identifiant principal, ses coordonnées décimales, son type spectral, sa parallaxe, son redshift et son mouvement propre.	

GETPREVIEW(OID)

Résumé : Va chercher les données attendues sur Simbad.

Entrée : {String} Nom de l'objet étudié.	**Sortie :** {Promesse} Bloque l'exécution du Javascript tant que les données n'ont pas été récupérées de Simbad.
Commentaire : Récupère les données via Simbad TAP et une requête Ajax.	

WELCOME.JS

Ce fichier contient les fonctions liées au message d'accueil affiché lors du chargement de la page.

DISPLAYWELCOMEMESSAGE()

Résumé : Affiche le message d'accueil.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

INFOGETMEASURE()

Résumé : Affiche un message de présentation de l'intention pour récupérer des mesures sur Simbad et VizieR.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

INFOGETHIERARCHY()

Résumé : Affiche un message de présentation de l'intention pour récupérer les enfants, les parents et les frères sur Simbad.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

INFOSHOWIMAGE()

Résumé : Affiche un message de présentation des intentions gravitant autour des images récupérées depuis Aladin-Lite.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

INFOGETCATALOGUES()

Résumé : Affiche un message de présentation de l'intention pour récupérer des catalogues de VizieR.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

INFOFINDOBJECT()

Résumé : Affiche un message de présentation de l'intention pour identifier des objets autour d'un autre.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

INFOLISTOBJECT()

Résumé : Affiche un message de présentation de l'intention pour récupérer une liste

d’objets possédant un même type.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

SEARCHBARTXT(TXT)

Résumé : Remplace le texte écrit dans la barre par celui passé en paramètre.	
Entrée : {String} Message à afficher dans la barre.	**Sortie :** Aucune
Commentaire : RAS	

KNOWLEDGE

Ce dossier contient les scripts utilisés pour récupérer les données sur les différents outils du CDS.

ALADIN.JS

Ce fichier contient les fonctions en rapport avec la gestion des widgets Aladin-Lite.

REQUESTALADIN(OID,WL)

Résumé : Ajoute au chat un widget Aladin-Lite.	
Entrée : {String} Nom de l’objet sur lequel centrer le widget. {String} Longueur d’onde étudiée.	**Sortie :** {Promesse} Permet de ne pas afficher la bulle avant que le widget ne soit chargé.
Commentaire : Avant d’ajouter un nouveau widget, la fonction recentre le dernier affiché sur sa cible de base et change son ID pour éviter les conflits.	

CHANGEFILTER(IDFILTER, OID)

Résumé : Change le filtre dans le widget Aladin	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

SETDEAULTFILTER(WL)

Résumé : Affiche le filtre par défaut de la catégorie demandée	
Entrée : Aucune	**Sortie :** AUcune
Commentaire : RAS	

GETFILTERBYNAME(IDFILTER)

Résumé : Renvoie les paramètres d’un filtre données	
Entrée : Aucune	**Sortie :** {Array} paramètre d’un filtre : nom,hips_services_url ...
Commentaire : RAS	

GOTOALADIN()

Résumé : Recentre le dernier widget Aladin-Lite affiché sur sa cible de base.	
Entrée : Aucune	**Sortie :** {Promesse} Evite que la fonction

	ne s'exécute après que le nouveau widget ne soit ajouté.
Commentaire : RAS	

RESIZEALADIN()

Résumé : Modifie la taille du widget pour l'adapter à la taille de l'écran.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : Fonction appelée lors d'un redimensionnement de la fenêtre.	

REPLACEALADIN()

Résumé : Modifie l'ID du dernier widget Aladin-Lite pour éviter les conflits.	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

CHECKALADIN()

Résumé : Vérifie si un widget Aladin-Lite est affiché dans la zone de chat.	
Entrée : Aucune	**Sortie :** {Boolean} True s'il y en a un, false sinon
Commentaire : RAS	

NEXTTOCHATALADIN(MSG)

Résumé : Ajoute le widget Aladin à coté du chat à la fenêtre rendue visible	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

DELETENEXTTOCHATALADIN()

Résumé : Enlève le widget Aladin à coté du chat et cache la fenêtre	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

DELETEFILTERS()

Résumé : Enlève tous les raccourcis de filtres ajoutés à la fenêtre Aladdin	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

JQUERY ON.CHANGE → CAHNGE LES FILTRES D'ALADIN EN FONCTION DES BOUTONS COCHÉS

ADDBUTTONIMG()

Résumé : Permet d'ajouter et de cocher les boutons de raccourcis pour un filtre choisi dans la liste affiché par le chatbot	
Entrée : Aucune	**Sortie :** Aucune
Commentaire : RAS	

CATALOGUES.JS

Ce fichier contient les fonctions relatives à la récupération de noms de catalogues sur MocServer.

BUILDQUERY(NLURES)

Résumé : Construit la requête MocServer en fonction des paramètres isolés par le traitement du langage naturel.	
Entrée : {JSON} Résultats retournés par Dialogflow	**Sortie :** {Promesse} Bloque le Javascript tant que les résultats récupérés sur d'autres outils comme Sesame ou VizieR ne sont pas arrivés.
Commentaire : RAS	

QUERYVIZIER(QUERY)

Résumé : Envoie la requête construite dans la fonction précédente à MocServer.	
Entrée : {String} URL d'accès à l'outil	**Sortie :** {Promesse} Résultats de la requête quand ils sont arrivés.
Commentaire : Si au bout de 10 secondes les résultats ne sont pas arrivés, la requête échoue.	

JSONLISTTOSTRING(JSONLIST)

Résumé : Isole le nom des catalogues récupérés sur MocServer.	
Entrée : {JSON} Résultats de la requête à MocServer.	**Sortie :** {String} Liste des catalogues, rangés dans une liste déroulante (sauf le premier).
Commentaire : RAS	

JSONLISTTOSTRINGELASTIC(JSONLIST)

Résumé : Isole le nom des catalogues récupérés sur MocServer.	
Entrée : {JSON} Résultats de la requête à MocServer.	**Sortie :** {String} Liste des catalogues, rangés dans une liste déroulante (sauf le premier).
Commentaire : RAS	

CATALOGANSWER(MOCRES)

Résumé : Construit le message en langage naturel pour accompagner les résultats de la requête à MocServer.	
Entrée : {JSON} Résultats de la requête à MocServer.	**Sortie :** {String} Message du chatbot prêt à être affiché.
Commentaire : RAS	

GETPARAMETERSCATALOG(NLURES)

Résumé : Isole les paramètres retournés par le traitement du langage naturel dans des variables.	
Entrée : {JSON} Résultats retournés par Dialogflow.	**Sortie :** Aucune
Commentaire : Les valeurs sont stockées dans les variables globales suivantes.	

ElasticSearch.js

Contient les fonctions nécessaire à l'interrogation du moteur de recherche Elasticsearch (interrogation de Vizier)

ElasticBuildQuery(jsonList)

Résumé : Construit la requête pour Elasticsearch à partir des paramètres fourni par getParametersCatalog	
Entrée : {JSON} Résultats de la requête à MocServer.	**Sortie :** {query} query pour interroger ElasticSearch
Commentaire : RAS	

ElasticSearch(jsonList)

Résumé : Fonction qui gère les autres fonctions de Elastic search	
Entrée : {JSON} Résultats de la requête à MocServer.	**Sortie :** {query} query pour interroger ElasticSearch
Commentaire : RAS	

queryElasticSearch(query)

Résumé : Fonction qui gère l'envoi de la requête à ElasticSearcg	
Entrée : {JSON} Résultats de la requête à MocServer.	**Sortie :** {JSON} resultat de la requête
Commentaire : RAS	

catalogAnswerElastic(résultat)

Résumé : Construit le message en langage naturel pour accompagner les résultats de la requête à MocServer.	
Entrée : {JSON} Résultats de la requête à MocServer.	**Sortie :** {String} Message du chatbot prêt à être affiché.
Commentaire : RAS	

Hierarchy.js

Contient les fonctions de récupération des enfants et des parents sur Simbad.

getParentsSIMBAD(object)

Résumé : Récupère les parents d'un objet sur Simbad.	
Entrée : {String} Nom de l'objet	**Sortie :** {Promesse} Résultats retournés par Simbad dès qu'ils sont arrivés.
Commentaire : Si au bout de 10 secondes les résultats ne sont pas arrivés, la requête échoue.	

getChildrenSIMBAD(object,otype)

Résumé : Récupère les enfants d'un objet sur Simbad.	
Entrée : {String} Nom de l'objet {String} Type d'objet devant être possédé par les enfants (ex : Etoiles)	**Sortie :** {Promesse} Résultats retournés par Simbad dès qu'ils sont arrivés.
Commentaire : Si au bout de 10 secondes les résultats ne sont pas arrivés, la requête	

échoue.

SESAME.JS

Ce fichier contient les fonctions en rapport avec la récupération de données sur Sesame et leur exploitation.

OIDEXISTS(OID)

Résumé : Vérifie via Sesame si le nom d'objet passé en entrée correspond à un objet présent sur Simbad.	
Entrée : {String} Nom de l'objet	**Sortie :** {Promesse} True si l'objet existe, false sinon. La valeur ne peut être renvoyée que quand la fonction a récupéré les résultats de Sesame.
Commentaire : RAS	

REQUESTSESAME(OID)

Résumé : Récupère quelques informations de Simbad via Sesame.	
Entrée : {String} Nom de l'objet	**Sortie :** {Promesse} Résultats retournés par Sesame dès qu'ils sont arrivés.
Commentaire : Sesame renvoie des résultats en XML, il est plus intéressant de les parser en JSON pour continuer avec Javascript.	

SEPARATESYNONYMS(TREE)

Résumé : Isole le champ « alias » dans le JSON de réponse.	
Entrée : {JSON} Version JSON du XML de réponse.	**Sortie :** {Array} Branche du JSON contenant les informations sur les alias
Commentaire : RAS	

GETSYNONYMS(OBJXML)

Résumé : Récupère les différentes façons d'appeler un objet astronomique dans le JSON de réponse.	
Entrée : {JSON} Version JSON du XML de réponse.	**Sortie :** {Array} Liste des synonymes.
Commentaire : Retourne null si l'objet de la recherche n'existe pas.	

GETCOORDINATES(OBJXML,DECIMAL)

Résumé : Isole les coordonnées dans le JSON de réponse.	
Entrée : {JSON} Version JSON du XML de réponse. {Boolean} True si les coordonnées à récupérer sont les décimales, False pour les équatoriales.	**Sortie :** {Array} Tableau de la forme [Right Ascension, Declination]
Commentaire : Retourne undefined si l'objet de la recherche n'existe pas.	

SIMBAD.JS

Contient les fonctions en rapport avec la récupération de données sur Simbad.

Constante measid → correspondance langage naturel – mots clés d'interrogation de Simbad.

REQUESTSIMTAP(OID,MEAS)

Résumé : Récupère les données possibles dans la table associé à la mesure cherchée de Simbad.	
Entrée : {String} Nom de l'objet recherché. {String} Mesure recherchée.	**Sortie :** {Promesse} Résultats retournés par Simbad dès qu'ils sont arrivés.
Commentaire : La requête échoue au bout de 10 secondes si les résultats ne sont toujours pas arrivés.	

INFORMATIONSABOUTMEASURE(RESULT,MEAS)

Résumé : Traite les informations renvoyées par la requête faite à Simbad	
Entrée : {JSO ?} Résultat {String} Mesure recherchée.	**Sortie :** {Array} Résultats
Commentaire : RAS	

GETMEASIDS(MEAS)

Résumé : Gère la correspondance langage naturel – mots clés	
Entrée : {String} Mesure recherchée.	**Sortie :** {string} mot clé associé au mot en langage naturel
Commentaire : La requête échoue au bout de 10 secondes si les résultats ne sont toujours pas arrivés.	

REQUESTSIMTAPOTYPE(OTYPE,MEAS,LIMIT,CONDITION)

Résumé : Récupère une liste d'objets possédant le type précisé.	
Entrée : {String} Type d'objet recherché. {String} Mesure recherchée pour ce type d'objet. {Number} Taille maximale de la liste (-1 = pas de limite autre que celle par défaut sur Simbad, soit 50 000) {String} Condition à ajouter derrière un WHERE SQL	**Sortie :** {Promesse} Résultats retournés par Simbad dès qu'ils sont arrivés.
Commentaire : La requête échoue au bout de 10 secondes si les résultats ne sont toujours pas arrivés.	

GETLONGOTYPE(OID)

Résumé : Récupère le type de l'objet passé en paramètre sous sa forme longue.	
Entrée : {String} Nom de l'objet recherché.	**Sortie :** {Promesse} Résultats retournés par Simbad dès qu'ils sont arrivés.
Commentaire : La requête échoue au bout de 10 secondes si les résultats ne sont toujours pas arrivés.	

GETFLUXES(OID)

Résumé : Récupère les magnitudes associées à un objet astronomique.

Entrée : {String} Nom de l'objet recherché.	**Sortie :** {Promesse} Résultats retournés par Simbad dès qu'ils sont arrivés.
Commentaire : La requête échoue au bout de 10 secondes si les résultats ne sont toujours pas arrivés.	

SIMBADBYCOORDINATES(RA,DEC,RADIUS)

Résumé : Recherche le nom et le type de l'objet le plus proche situé aux coordonnées précisées.	
Entrée : {String ou Number} Right Ascension de l'objet. {String ou Number} Declination de l'objet {String ou Number} Rayon de recherche autour des coordonnées précisées (si possible diamètre de l'objet recherché)	**Sortie :** {Promesse} Résultats retournés par Simbad dès qu'ils sont arrivés.
Commentaire : La requête échoue au bout de 10 secondes si les résultats ne sont toujours pas arrivés.	

GETALL(OID)

Résumé : Simule une requête de type « everything » (récupère toutes les données de Simbad sur un objet).	
Entrée : {String} Nom de l'objet recherché.	**Sortie :** Aucune
Commentaire : La fonction affiche directement les résultats en utilisant le format par défaut.	

SIMANSWERTOSTRING(SIMANSWER)

Résumé : Hub permettant d'attribuer le bon format à la mesure à afficher.	
Entrée : {Array} Résultat obtenu après la fonction getInformation.	**Sortie :** {String} Résultats dans le bon format.
Commentaire : Si on n'associe pas la mesure à son format ici, elle est affichée dans le format par défaut.	

FINDMEASURE(RESULTS)

Résumé : Détermine à partir du résultat renvoyé par la fonction getInformation quelle était la mesure demandée par l'utilisateur.	
Entrée : {Array} Résultats de la fonction.	**Sortie :** {String} Le nom de la mesure.
Commentaire : Pour le cas où toutes les mesures sont demandées, un champ « Everything » est ajouté manuellement au début du résultat de la fonction getInformation et cette fonction renvoie « all ».	

SORTRESULTS(RES)

Résumé : Trie les résultats retournés par la fonction getInformation en fonction de la taille de leur description.	
Entrée : {Array} Résultats de la fonction.	**Sortie :** {Array} Le même tableau mais trié.
Commentaire : Les mesures avec la plus petite description (souvent les moins précises ou les plus générales) sont placées en premier.	

SIMBADCOLUMNNAME(MEAS)

Résumé : Récupère le nom des colonnes Simbad en rapport avec la mesure passée en

paramètre.	
Entrée : {String} Nom de la mesure.	**Sortie :** {Array} Toutes les colonnes correspondantes.
Commentaire : Même fonction que celle utilisée sur cette page du CDS.	

ENCODESIMBAD(STR)

Résumé : Certains noms d'objets contiennent des apostrophes. Pour pouvoir gérer ces objets avec TAP, il faut la doubler.	
Entrée : {String} Nom de l'objet recherché.	**Sortie :** {String} Nom de l'objet au bon format pour TAP.
Commentaire : RAS	

GETFOVFOROBJECT(OBJECTNAME)

Résumé : Détermine un rayon de recherche pas trop mauvais autour d'un objet astronomique.	
Entrée : {String} Nom de l'objet recherché.	**Sortie :** {Number} Valeur du rayon
Commentaire : Fonction utilisée par Aladin-Lite pour trouver un Fov efficace.	

FINDOBJECTEXAMPLE(OTYPE)

Résumé : Trouve sur Simbad un nom d'objet possédant le type passé en entrée.	
Entrée : {String} Type recherché.	**Sortie :** {Promesse} Renvoie le nom de l'objet dès que la recherche sur Simbad a renvoyé ses résultats.
Commentaire : Utilisée dans les fonctions relatives à Aladin-Lite.	

MEASURE2SIMBAD(MEAS)

Résumé : Retourne le nom de la colonne principale associée aux mesures les plus courantes.	
Entrée : {String} Mesure en langage naturel.	**Sortie :** {String} Nom de la colonne.
Commentaire : Les mesures gérées par ces fonctions sont la parallaxe, le redshift, la right ascension et la declination.	

REFSIMBAD(OID)

Résumé : Ajoute à un nom d'objet un hyperlien renvoyant vers sa page Simbad.	
Entrée : {String} Nom de l'objet recherché.	**Sortie :** {String} Même objet avec l'hyperlien.
Commentaire : RAS	

VIZIER.JS

Ce fichier contient les fonctions en rapport avec la récupération de données sur VizieR.

REQUESTVIZTAP(OBJECT,MEAS)

Résumé : Fonction principale de ce script. Contient la stratégie de récupération des données sur VizieR. Récupère les données liées à la mesure précisée pour l'objet donné dans le catalogue le plus récent et dans le plus populaire.	
Entrée : {String} Nom de l'objet	**Sortie :** Aucune.

recherché. {String} Nom de la mesure recherchée.	
Commentaire : L'affichage (l'appel à la fonction displayChatbotMessage) se fait exceptionnellement dans la fonction de récupération des données pour les intentions liées à VizieR pour pouvoir afficher les informations au fur et à mesure qu'elles arrivent plutôt que tout d'un coup.	

GETUCD(MEASURE)

Résumé : Retourne les UCDs liés à la mesure en langage naturel passée en entrée. Ne retourne pas juste l'UCD principal mais tous ceux qui dans leur description mentionnent la mesure.	
Entrée : {String} Nom de la mesure recherchée.	**Sortie :** {Promesse} Les UCDs dès qu'ils sont récupérés.
Commentaire : La fonction échoue si aucune réponse n'est renvoyée 10 secondes après l'envoi de la requête.	

GETCATALOGUES(UCD,COORDINATES)

Résumé : Récupère une liste de catalogues contenant l'UCD passé en entrée et étudiant l'objet astronomique aux coordonnées données.	
Entrée : {String} UCD de la mesure recherchée. {Array ou String} Coordonnées de l'objet au format [RA, DEC] ou type de l'objet.	**Sortie :** {Promesse} Liste des catalogues récupérée sur MocServer avec les paramètres donnés.
Commentaire : Le paramètre « coordonnées » peut aussi contenir un type d'objet recherché (ex : Etoile) plutôt que des coordonnées se référant à une région du ciel ou à un objet précis.	

SORTBYPOPULARITY(JSON)

Résumé : Trie la liste des catalogues retournée par MocServer selon leur paramètre de popularité.	
Entrée : {JSON} Résultat retourné par la fonction précédente.	**Sortie :** {JSON} Même objet mais trié.
Commentaire : ATTENTION ! Cette fonction trie par ordre croissant. Cela signifie que le catalogue le moins populaire sera placé en premier. Pour avoir le plus populaire il faut inverser la liste. Si deux catalogues ont la même popularité, le tri se fait par ordre alphabétique.	

SORTBYDATE(JSON)

Résumé : Trie la liste des catalogues retournée par MocServer selon leur date de publication.	
Entrée : {JSON} Résultat retourné par la fonction getCatalogues.	**Sortie :** {JSON} Même objet mais trié.
Commentaire : ATTENTION ! Cette fonction trie par ordre croissant. Cela signifie que le catalogue le plus ancien sera placé en premier. Pour avoir le plus récent il faut inverser la liste. Si deux catalogues ont la même date de publication, le tri se fait par ordre alphabétique.	

SORTBYHEIGHT(JSON)

Résumé : Trie la liste des catalogues retournée par MocServer selon leur taille (nombre de ligne)	
Entrée : {JSON} Résultat retourné par la	**Sortie :** {JSON} Même objet mais trié.

fonction getCatalogues.	
Commentaire : ATTENTION ! Cette fonction trie par ordre croissant. Cela signifie que le catalogue le plus ancien sera placé en premier. Pour avoir le plus récent il faut inverser la liste. Si deux catalogues ont la même date de publication, le tri se fait par ordre alphabétique.	

TAPREQUESTCOLUMNS(CATALOGUE)

Résumé : Récupère la structure de la table passée en paramètre (le nom et la description de ses colonnes).	
Entrée : {Objet} Catalogue extrait de la liste retournée par la fonction getCatalogues.	**Sortie :** {Promesse} Structure JSON contenant entre autres le nom et la description du catalogue.
Commentaire : La fonction échoue au bout de 10 secondes si aucune réponse n'est retournée.	

GETCOLUMNSNAME(METACATALOG,UCD)

Résumé : Fonction qui exploite le résultat de la fonction précédente pour en tirer les informations utiles pour la recherche de données.	
Entrée : {JSON} Résultat de la précédente fonction. {String} Nom de la mesure recherchée.	**Sortie :** {JSON} Structure JSON de la forme {position : {ra : , dec : }, col : []}
Commentaire : Le champ « position » récupère à part le nom des colonnes relatives aux coordonnées de l'objet, indispensables pour faire une recherche ADQL avec les fonctions POINT et CIRCLE. Le champ « colonne » contient un tableau contenant toutes les colonnes de la table dont la description fait mention de la mesure recherchée. Chaque colonne est résumée dans un tableau de la forme suivante [nom de la colonne dans la table, description, unité].	

FINALREQUESTTAP(SELECT,TABLE,COORD,TYPE)

Résumé : Récupère les données sur VizieR à travers TAP avec toutes les informations récupérées dans les fonctions précédentes.	
Entrée : {Array} Nom des colonnes à étudier. {Objet} Catalogue extrait de la liste retournée par la fonction getCatalogues. {Array} Coordonnées de l'objet recherché dans un tableau de la forme [Right Ascension, Declination]. {String} « Last » ou « Most popular » en fonction du catalogue étudié.	**Sortie :** {Promesse} Une chaîne de caractère contenant les données récupérées et formattées pour l'affichage par la fonction successVizTAP dès qu'elles sont récupérées.
Commentaire : RAS	

SUCCESSVIZTAP(RESULT,TABLE,TYPE)

Résumé : Crée le message en langage naturel pour habiller les résultats récupérés sur VizieR.	
Entrée : {JSON} Résultat de la requête TAP envoyée dans la fonction précédente. {Objet} Catalogue extrait de la liste retournée par la fonction getCatalogues. {String} « Last » ou « Most popular » en fonction du catalogue étudié.	**Sortie :** {String} Message du chatbot à afficher.

Commentaire : RAS

formatTableMCAT(json)

Résumé : Met en forme les réponses sous forme de tableau (5 résultats)	
Entrée : {String} mesure recherchée {JSON} Résultat retourné par la fonction getCatalogues.	**Sortie :** {String} réponse mis en forme
Commentaire : RAS	

formatTableMOBJ(json)

Résumé : Met en forme les réponses sous forme de tableau (5 résultats) pour les objets types	
Entrée : {String} mesure recherchée {JSON} Résultat retourné par la fonction getCatalogues.	**Sortie :** {String} réponse mis en forme
Commentaire : RAS	

getCatalogueType(type)

Résumé : Simple fonction qui renvoie une version rédigée du type de catalogue passé en entrée.	
Entrée : {String} Type du catalogue.	**Sortie :** {String} Version rédigée de ce type.
Commentaire : Cette fonction est relativement inutile. En effet elle se contente de renvoyer « the most popular » quand on lui passe « best » et « the latest » quand on passe « last ».	

splitResults(results,meta)

Résumé : Récupère dans la structure JSON retournée par la fonction finalRequestTAP les résultats liés à une métadonnée précise.	
Entrée : {JSON} Résultat de la requête. {Number} Indice de la métadonnée dans le champ « metadata » du JSON précédent.	**Sortie :** {Array} Tableau contenant les données concernées.
Commentaire : RAS	

splitResultsCAT(results,meta)

Résumé : Récupère dans la structure JSON retournée par la fonction finalRequestTAP les résultats liés à une métadonnée précise.	
Entrée : {JSON} Résultat de la requête. {Number} Indice de la métadonnée dans le champ « metadata » du JSON précédent.	**Sortie :** {Array} Tableau contenant les données concernées.
Commentaire : RAS	

findSpectrum(oid)

Résumé : Indique si des spectres de l'objet passé en paramètre sont présents sur VizieR.

Entrée : {String} Nom de l'objet recherché.	**Sortie :** {Null ou String} null si aucun spectre n'est présent sur VizieR, une phrase à envoyer en tant que message du chatbot indiquant que si dans le cas contraire.
Commentaire : Les spectres ne sont pas affichés à l'écran puisqu'ils prendraient trop de place. Pour fournir cette fonctionnalité sans nuire à l'interface, la fonction donne un lien d'accès à l'interface VizieR de récupération des spectres avec tous les paramètres à entrer pour les trouver.	

GETSPECTRUM(RA,DEC)

Résumé : Regarde sur VizieR si des spectres de la région du ciel aux coordonnées passées en entrée sont disponibles dans un rayon de 0.5 degrés autour de ce point.	
Entrée : {Number} Right Ascension. {Number} Declination.	**Sortie :** {Promesse} La liste des spectres trouvés dès qu'elle est récupérée par ajax.
Commentaire : La requête échoue si aucun résultat n'est retourné dans les 10 secondes suivant l'envoi.	

FINDSPECT(OID)

Résumé : Fonction qui aide à trouver les paramètres pour tracer les données associés sur Aladin	
Entrée : {String} Nom de l'objet recherché.	**Sortie :** {Null ou String} null si aucun spectre n'est présent sur VizieR, une phrase à envoyer en tant que message du chatbot indiquant que si dans le cas contraire.
Commentaire : Les spectres ne sont pas affichés à l'écran puisqu'ils prendraient trop de place. Pour fournir cette fonctionnalité sans nuire à l'interface, la fonction donne un lien d'accès à l'interface VizieR de récupération des spectres avec tous les paramètres à entrer pour les trouver.	

NLU

Ce dossier contient les scripts qui communiquent avec Dialogflow et les fonctions correspondantes aux intentions récupérées de l'outil de NLU.

AROUND.JS

Ce fichier contient la fonction principale et les fonctions de support associées à l'intention « find_object ».

FINDOBJECT(RESULT)

Résumé : Fonction principale associée à l'intention « find_object » détectée par Dialogflow. En fonction des entités isolées après traitement du langage naturel, trouve un objet sur le dernier widget Aladin-Lite ou affiche tous les objets autour d'un autre.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** Aucune
Commentaire : Si aucun nom d'objet a été détecté par Dialogflow dans le message de l'utilisateur et qu'un widget Aladin-Lite est présent dans la zone de chat, la fonction dessine un cercle autour de l'objet pointé par le curseur et indique sur le widget son nom et son type. Si un nom d'objet est précisé, la fonction recherche dans Simbad une	

liste d'objets présents dans un rayon précisé par l'utilisateur ou par défaut 0.5 degrés autour des coordonnées de l'objet précisé.

DRAWSPECTRUM(OID)

Résumé : Affiche les spectres trouvés sur Aladin dans un radius autour de l'objet	
Entrée : {String} oid de l'objet recherché	**Sortie :** Aucune
Commentaire : RAS	

DRAWLIST(LIST)

Résumé : Affiche les objets recherchés autour d'un objet astronomique	
Entrée : {Array} liste d'objet	**Sortie :** Aucune
Commentaire : RAS	

DETERMINERADIUS(RADIUS)

Résumé : Transforme un rayon de recherche en langage naturel (ex : 3 degrés, 7 minutes, ...) en Number en degrés.	
Entrée : {String} Rayon précisé par l'utilisateur dans son message.	**Sortie :** {Number} Le même rayon en number.
Commentaire : RAS	

GETOBJECTLIST(CAT)

Résumé : Fonction appelée dès que le widget Aladin a affiché les objets sur son interface. Récupère ceux-ci et les place dans une liste déroulante.	
Entrée : {JSON} Données récupérées par Aladin-Lite sur Simbad.	**Sortie :** Aucune.
Commentaire : RAS	

DRAWFUNCTION(SOURCE, CANVASCTX, VIEWPARAMS)

Résumé : Fonction récupérée dans la documentation de l'API javascript d'Aladin-Lite. Affiche sur le widget les résultats récupérés de Simbad.	
Entrée : Paramètres renseignés automatiquement lorsque la fonction est appelée en tant que callback après la récupération des données.	**Sortie :** Aucune.
Commentaire : RAS	

KEEPOTYPE(LIST)

Résumé : Filtre la liste des objets établie dans la fonction getObjectList avec le type d'objet précisé par l'utilisateur dans son message.	
Entrée : {Array} Tableau de la forme [Nom de l'objet, Type de l'objet] contenant les objets affichés sur le widget Aladin-Lite.	**Sortie :** {Promesse} La liste filtrée avec juste les objets contenant le bon type.
Commentaire : RAS	

FAMILY.JS

Ce fichier contient les fonctions associées à la récupération de liens hiérarchiques dans Simbad (enfants/parents/siblings), soit les intentions « get_parents », « get_children », « get_siblings ».

GETPARENTCHILD(RESULTS,MODE)

Résumé : Fonction liées aux intentions « get_parents » et « get_children » pour récupérer les enfants et les parents d'un objet astronomique sur Simbad.	
Entrée : {JSON} Résultat retourné par Dialogflow. {String} « parent » ou « child ».	**Sortie :** Aucune.
Commentaire : RAS	

GETSIBLINGS(RESULTS)

Résumé : Fonction liée à l'intention « get_siblings ». Récupère les frères et sœur d'un objet astronomique.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** Aucune.
Commentaire : RAS	

OTYPEFILTER(RESULTS,OTYPE)

Résumé : Filtre la liste des parents/enfants/siblings récupérés dans les fonctions précédentes avec le type d'objet précisé par l'utilisateur dans son message (Fonction à priori non utilisée).	
Entrée : {JSON} Résultat retourné par Simbad.	**Sortie :** Aucune.
Commentaire : RAS	

FINDOTYPE(RESULTS)

Résumé : Identifie l'indice du champ où se trouve le type de l'objet dans le résultat JSON retourné par Simbad.	
Entrée : {JSON} Résultat retourné par Simbad.	**Sortie :** {Number} Indice du champ dans la partie « metadata ».
Commentaire : RAS	

INTENT.JS

Ce fichier contient le reste des fonctions associées aux intentions détectées par Dialogflow.

GETCATALOG(RESULT)

Résumé : Fonction liée à l'intention « get_catalogue » pour récupérer une liste de catalogues VizieR selon certains paramètres.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** Aucune.
Commentaire : RAS	

GETMEASURE(RESULT)

Résumé : Fonction liée à l'intention « get_measure » pour récupérer des données sur Simbad et VizieR.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** Aucune.
Commentaire : RAS	

CHECKOTYPE(OID,MEAS,FIRST)

Résumé : Fonction liée à l'intention « get_measure ». Permet de déterminer si le nom d'objet détecté par Dialogflow n'est pas en fait un type d'objet.	
Entrée : {String} Nom d'objet détecté par Dialogflow. {String} Mesure recherchée sur cet objet. {Number} Nombre de fois où la fonction a été appelée récursivement.	**Sortie :** Aucune.
Commentaire : Si la fonction reconnait un type d'objet, elle appelle la fonction suivante (getMeasureOtype). Sinon le mot est considéré comme non-reconnu par les services du CDS.	

GETMEASUREOTYPE(OID,MEAS)

Résumé : Fonction liée à l'intention « get_measure ». Récupère des données sur Simbad en utilisant non pas un nom d'objet mais un type.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** Aucune.
Commentaire : RAS	

TRYAGAIN(RESULT)

Résumé : Fonction liée à l'intention « try_again». Lance une recherche de données uniquement sur VizieR.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** Aucune.
Commentaire : RAS	

DEFAULTANSWER(RESULT)

Résumé : Envoie le message par défaut à afficher lorsqu'une intention n'est pas reconnue par le chatbot.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** Aucune.
Commentaire : RAS	

SHOWIMAGE(RESULT)

Résumé : Fonction liée à l'intention « show_image » pour afficher un visuel d'un objet astronomique.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** Aucune.
Commentaire : RAS	

LISTOBJECT(RESULT)

Résumé : Fonction liée à l'intention « list_object» pour récupérer une liste d'objets astronomiques répondant à certains critères (type, valeur de redshift/parallaxe/right ascension/declination).	

Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** Aucune.
Commentaire : RAS	

SUGGESTION

Ce fichier contient toutes les fonctions servant aux réponse du chatbot (autre que l'affichage des résultats). Permet de faire des transitions. La suggestion se découpe en plusieurs parties (exemple peut varier selon la question précédente) :

1 Question à propos de l'objet demandé à la question précédente ?

2 Sinon même question à propos d'un autre objet ?

3 Sinon phrase de transition

Variable globales :

suggestmode → savoir si on doit faire une suggestion ou non-reconnu (1ère suggestion)

suggestmode2 → même chose mais 2ème partie (pour la deuxième et 3 suggestion)

befintent → variable qui garde en m »oire l'intention de la question précédente

check → variable qui garde l'itnetnion de la réponse de l'utilisateur à la première suggestion

SUGGESTIONHUB(RESULT)

Résumé : Fonction qui gère l'attribution des fonctions de suggestions en fonction de l'intenttion précédemment détectée	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** Aucune.
Commentaire : RAS	

RANDOMR()

Résumé : Fonction qui revoie une réponse aléatoire(parmi celles écrites) de transition (phrase de transition)	
Entrée :	**Sortie :** {String} phrase de transition aléatoire
Commentaire : RAS	

DONTKNOW()

Résumé : Fonction qui renvoie une réponse aléatoire (parmi celles écrites) si le chatbot n'a pas pas comprus	
Entrée :	**Sortie :** {string} phrase aléatoire
Commentaire : RAS	

TRANSITION_ARBRE(RESULT)

Résumé : Fonction qui gère la fin de la suggestion	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {string} phrase de transition
Commentaire : Le chatbot demande si il peut encore être utile à l'utilisateur	

SUGGEST_MEASURE(RESULT)

Résumé : Fonction qui gère la suggestion de l'intention get_measure	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {string} phrase de suggestion
Commentaire : RAS	

SUGGEST_FAMILY(RESULT)

Résumé : Fonction qui gère la suggestion des intentions get_parent / get_children / get_siblings	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {string} phrase de suggestion
Commentaire : RAS	

SUGGEST_IMAGE(RESULT)

Résumé : Fonction qui gère la suggestion de l'intention show_image	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {string} phrase de suggestion
Commentaire : RAS	

SUGGEST_CATALOGUE(RESULT)

Résumé : Fonction qui gère la suggestion de l'intention get_catalogue	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {string} phrase de suggestion
Commentaire : RAS	

SUGGEST_FINDOBJECT(RESULT)

Résumé : Fonction qui gère la suggestion de l'intention find_object	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {string} phrase de suggestion
Commentaire : RAS	

SUGGEST_LISTOBJECT(RESULT)

Résumé : Fonction qui gère la suggestion de l'intention list_object	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {string} phrase de suggestion
Commentaire : RAS	

DECONSTRUCTRESULTS.JS

Ce fichier contient des fonctions « tampons » entre le code développé pour l'application et Dialogflow. Chaque fonction permet de récupérer un champ important du traitement du langage naturel. Si un jour Dialogflow venait à être changé au profit d'un autre moteur de NLU, il suffirait de changer le contenu de ces fonctions et d'indiquer comment récupérer les champs importants pour le reste de l'application dans les résultats retournés par le nouveau moteur. Toutes les récupérations d'informations venant du

traitement du langage naturel dans le reste du code se font en utilisant les fonctions de ce fichier javascript.

GETINTENT(JSONRES)

Résumé : Récupère dans le JSON de réponse de l'API l'intention de l'utilisateur détectée par Dialogflow.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} L'intention.
Commentaire : RAS	

GETREQUEST(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API le message envoyé par l'utilisateur et traité par Dialogflow.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} La requête de l'utilisateur.
Commentaire : RAS	

GETDEFAULTANSWER(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API le message par défaut associé à l'intention de l'utilisateur comme entré dans Dialogflow.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} Le message par défaut.
Commentaire : RAS	

GETOID(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API le nom de l'objet astronomique parmi les entités détectées par Dialogflow. Entité présente dans la plupart des intentions.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} Un nom d'objet.
Commentaire : RAS	

GETMEAS(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API les mesures parmi les entités détectées par Dialogflow. Entité présente dans toutes les intentions relatives à la récupération de données.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {Array} Les mesures détectées.
Commentaire : Il peut y avoir plusieurs mesures, elles sont donc placées dans un tableau.	

GETCONDITION(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API les conditions parmi les entités détectées par Dialogflow. Entité utilisée dans l'intention « list_object ».	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {Array} Les conditions.
Commentaire : Plusieurs conditions peuvent être précisées par l'utilisateur.	

GETOTYPE(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API le type de l'objet astronomique parmi les entités détectées par Dialogflow. Entité utilisée dans la plupart des intentions.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} Le type.
Commentaire : RAS	

GETWAVELENGTH(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API la longueur d'onde parmi les entités détectées par Dialogflow. Entité utilisée dans les intentions « show_image » et « context.image ».	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} La longueur d'onde.
Commentaire : RAS	

GETCATALOGNAME(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API le nom du catalogue demandé parmi les entités détectées par Dialogflow. Entité utilisée avec l'intention « get_catalogue ».	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} Le nom du catalogue.
Commentaire : RAS	

GETCATALOGYEAR(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API le nom du catalogue demandé parmi les entités détectées par Dialogflow. Entité utilisée avec l'intention « get_catalogue ».	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} Le nom du catalogue.
Commentaire : RAS	

GETCATALOGDESC(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API le nom du catalogue demandé parmi les entités détectées par Dialogflow. Entité utilisée avec l'intention « get_catalogue ».	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} Le nom du catalogue.
Commentaire : RAS	

GETCATALOGWAY(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API le nom du moteur recherche (s'il y a « with Elasticsearch ») parmi les entités détectées par Dialogflow. Entité utilisée avec l'intention « get_catalogue ».	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} Le nom du catalogue.
Commentaire : RAS	

GETAUTHOR(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API le nom de l'auteur d'un catalogue parmi les entités détectées par Dialogflow. Entité liée à l'intention « get_catalogue ».	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} Le nom de l'auteurs.
Commentaire : Cette entité n'est pas gérée dans le code.	

GETCOORDINATESNLU(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API les coordonnées précisées parmi les entités détectées par Dialogflow. Entité liée à l'intention « list_object ».	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} Les coordonnées.
Commentaire : Les coordonnées sont en langage naturel et donc à transformer en langage formel pour être traitées par le code.	

GETRADIUS(RESULT)

Résumé : Récupère dans le JSON de réponse de l'API le rayon précisé parmi les entités détectées par Dialogflow.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} Rayon.
Commentaire : Le rayon est en langage naturel (unité incluse) et doit donc être par la suite transformé en langage formel pour être utilisable.	

PLEASEPRECISE(RESULT)

Résumé : Default intent, Fonction qui informe l'utilisateur que Dialogflow n'a pas trouvé d'intention et que l'on ne puet donc pas donner de réponse à la requête	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String}
Commentaire : RAS	

PLEASEHELP(RESULT)

Résumé : Fonction qui affiche le menu initial	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String} L
Commentaire : Commande help dans le chatbot	

CONTEXTMEASURE(RESULT)

Résumé : Fonction qui demande à l'utilisateur sur quel objet porte sa question, s'il n'y a pas d'informations supplémentaires	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String}
Commentaire : Ne marche plus très bien car rentre en conflit avec les otypes pris en compte par get_measure (et donc plus d'oid obligatoire)	

CONTEXTOBJECT(RESULT)

Résumé : Fonction qui demande à l'utilisateur quel action il doit faire avec l'objet, s'il n'y a pas d'informations complémentaires	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** {String}
Commentaire : RAS	

NLU.JS

Ce fichier contient les requêtes ajax à l'API de Dialogflow et l'aiguillage des différentes intentions.

REQUESTNLU(MSG)

Résumé : Encode le message entré par l'utilisateur puis l'envoie à l'API Dialogflow.	
Entrée : {String} Message de l'utilisateur.	**Sortie :** Aucune
Commentaire : RAS.	

BRAINHUB(RESULT)

Résumé : Associe le nom d'une intention comme défini sur Dialogflow avec la fonction de récupération des données.	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** Aucune.
Commentaire : RAS.	

FAQHUB(RESULT)

Résumé : Associe le nom d'une intention comme défini sur Dialogflow avec la fonction de récupération des données pour le chatbot FAQ	
Entrée : {JSON} Résultat retourné par Dialogflow.	**Sortie :** Aucune.
Commentaire : RAS.	

TOOLS

Ce dossier contient un arsenal d'outils utiles à l'exécution du code.

CHATBOTCOHABITATION

Ce fichier gère la cohabitaion entre les deux chatbots

GETSCROLL()

Résumé : Savoir la position dans la fenêtre	
Entrée :	**Sortie :** {pixel} position
Commentaire : RAS.	

SCROLLCHATBOTDROITE()

Résumé : Fonction qui permet d'aller sur le chatbot de droite	
Entrée :	**Sortie :**
Commentaire : associé à la flèche	

SCROLLCHATBOTGAUCHE()

Résumé : Fonction qui permet d'aller sur le chatbot de gauche (FAQ)

Entrée :	**Sortie :**
Commentaire : associé à la flèche	

Jquery

. Keydown → gère le changement du chatbot grâce au flèches directionnelles du clavier

.resize → gère le resize de la fenêtre window

IsAnObject()

Résumé : Fonction qui permet à l'utilisateur d'indiquer un mot comme un objet astronomique (fonction outrepassant la détection de Dialgoflow)	
Entrée :	**Sortie :**
Commentaire : RAS	

whichChat()

Résumé : Fonction qui permet de savoir dans quel chat on se trouve (au niveau d l'écran)	
voir Entrée :	**Sortie :**
Commentaire : associé à la flèche	

coordinatesConv

Ce fichier contient les fonctions en rapport avec les coordonnées.

wantedType(type)

Résumé : Associe un ensemble de synonymes aux types de coordonnées « galactic », « icrs », « decimal » et « all ».	
Entrée : {String} Coordonnées	**Sortie :** Voir termes ci-dessus.
Commentaire : RAS.	

CooConversion()

Résumé : Offre deux fonctions, CooConversion.GalacticToJ2000 et CooConversion.J2000ToGalactic, qui permettent de passer des coordonnées ICRS aux coordonnées Galactiques.	
Entrée : {String} Coordonnées	**Sortie :** Coordonnées converties.
Commentaire : RAS.	

Deg2Icrs(ra,dec)

Résumé : Convertit des coordonnées décimales en coordonnées équatoriales.	
Entrée : {Number} Right Ascension. {Number} Declination.	**Sortie :** {Array} [Right Ascension ICRS, Declination ICRS).
Commentaire : RAS.	

Deg2DMS(deg)

Résumé : Convertit une valeur en degrés décimales en degrés minutes secondes.	
Entrée : {Number} Nombre en degré.	**Sortie :** {String} Nombre en degrés minutes secondes.

Commentaire : RAS.

ICRS2DEC(ICRS)

Résumé : Convertit des coordonnées au format ICRS en coordonnées au format décimal.	
Entrée : {ICRS} Coordonnées.	**Sortie :** Coordonnées converties.
Commentaire : RAS.	

DEMO.JS

DEMOREMOVE()

Résumé : Analyse la phrase envoyée par l'utilisateur. S'il s'agit d'une phrase présente dans la démo, la fonction la retire.	
Entrée : Aucune.	**Sortie :** Aucune.
Commentaire : RAS.	

RUNDEMO()

Résumé : Fonction qui gère la démo (phrase vitesse ...)	
Entrée : Aucune.	**Sortie :** Aucune.
Commentaire : RAS.	

MYTIMER()

Résumé : Fonction qui gère le timer pour connaître le temps de réponse à chaque requête de la démo	
Entrée : Aucune.	**Sortie :** {float}
Commentaire : RAS.	

RAPPORT()

Résumé : Fonction qui gère la création d'un rapport de la démonstration	
Entrée : Aucune.	**Sortie :** Aucune.
Commentaire : RAS.	

ERROR.JS

UNKNOWNOBJECT(OID)

Résumé : Fait envoyer un message au chatbot pour dire qu'il ne connait pas l'objet en entrée.	
Entrée : {String} Nom de l'objet.	**Sortie :** Aucune.
Commentaire : RAS.	

GENERALERROR(MESSAGE,ERROR)

Résumé : Fait envoyer un message au chatbot pour détailler son erreur et lui dire de regarder la console pour plus d'informations.	
Entrée : {String} Message d'erreur. {Error} Erreur Javascript.	**Sortie :** Aucune.

Commentaire : RAS.

createLog()

Résumé : Crée un script de la conversation dans une chaîne de caractères et l'affiche dans la console, avec un rapport d'erreur en cas de problème.	
Entrée : Aucune.	**Sortie :** Aucune.
Commentaire : Idéalement cette fonction aurait enregistré le log grâce à PHP dans un fichier texte.	

General_tools.js

Ce fichier contient 3 constantes utiles pour les différents calculs mathématiques présents dans les fonctions.

toAlphanumeric(str)

Résumé : Retire les caractères spéciaux de la chaîne.	
Entrée : {String} Chaine à modifier.	**Sortie :** {String} Chaine à modifier.
Commentaire : RAS.	

capitalizeFirstLetter(str)

Résumé : Transforme la première lettre en capitale	
Entrée : {String} Chaine à modifier.	**Sortie :** {String} Chaine modifié
Commentaire : RAS.	

minusFirstLetter(str)

Résumé : Transforme la première lettre en minuscule	
Entrée : {String} Chaine à modifier.	**Sortie :** {String} Chaine modifié
Commentaire : RAS.	

spectralTypeColor(spt)

Résumé : Associe chaque type spectral à une couleur en hexadécimal.	
Entrée : {String} Type spectral.	**Sortie :** {String} Couleur.
Commentaire : RAS.	

mas2rad(mas)

Résumé : Convertit un nombre en mas en un nombre en radians.	
Entrée : {Number} Nombre en mas.	**Sortie :** {Number} Nombre en radians.
Commentaire : RAS.	

km2al(km)

Résumé : Convertit un nombre en kilomètres en un nombre en années-lumière.	
Entrée : {Number} Nombre en km.	**Sortie :** {Number} Nombre en al.
Commentaire : RAS	

al2pc(al)

Résumé : Convertit un nombre en années-lumière en un nombre en parsecs.

Entrée : {Number} Nombre en al.	**Sortie :** {Number} Nombre en pc.
Commentaire : RAS	

CALCULDISTANCE(PARALLAX)

Résumé : Calcule la distance d'un objet à partir de sa parallaxe.	
Entrée : {Number} Parallaxe en mas.	**Sortie :** {Array} [Distance en années-lumière, Distance en parsecs]..
Commentaire : RAS	

ENCODEURL(QUERY)

Résumé : Encode les caractères problématiques avec les URLs.	
Entrée : {String} Chaine à encoder.	**Sortie :** {String} Chaine encodée.
Commentaire : RAS	

GENERATESESSIONID()

Résumé : Génère une clé de session au hasard.	
Entrée : Aucune.	**Sortie :** Aucune.
Commentaire : RAS	

GETRANDOMINTINCLUSIVE(MIN, MAX)

Résumé : Retourne un nombre au hasard entre le min et le max fournis.	
Entrée : {Number} Minimum. {Number} Maximum.	**Sortie :** {Number} Nombre aléatoire.
Commentaire : Bornes comprises dans l'intervalle.	

TOSINGULAR(WORD)

Résumé : Transforme un mot pluriel en un mot singulier.	
Entrée : {String} Pluriel.	**Sortie :** {String} Singulier.
Commentaire : RAS	

REMOVESPACES(STRING)

Résumé : Retire les espaces inutiles à la fin d'un mot.	
Entrée : {String} Mot.	**Sortie :** {String} Mot sans les espaces inutiles.
Commentaire : For a NLU entity	

DELETESPACES(STRING)

Résumé : Retire les espaces inutiles	
Entrée : {String} Mot.	**Sortie :** {String} Mot sans les espaces inutiles.
Commentaire : Pour les conditions	

ISLETTER(LETTER)

Résumé : Indique si le caractère passé en entrée est une lettre ou non.	
Entrée : {String} Caractère.	**Sortie :** {Boolean} True s'il s'agit d'une

	lettre, false sinon.
Commentaire : RAS	

isCoordinates(coord)

Résumé : Indique si la chaîne en entrée pourrait correspondre à un format de coordonnées.	
Entrée : {String} Potentielles coordonnées.	**Sortie :** {Boolean} True s'il pourrait s'agir de coordonnées, false sinon.
Commentaire : RAS	

isComparative(condition)

Résumé : Isole les éléments d'une forme comparative.	
Entrée : {String} Condition détectée par Dialogflow.	**Sortie :** {Array} [Nom de la mesure, comparatif, valeur]
Commentaire : Détecte aussi la comparaison en langage naturel (most, less, -er).	

removeTag(chaîne)

Résumé : Créer un bouton pour pouvoir générer le CSV de la lister	
Entrée : {String} réponse avec la forme html	**Sortie :** {String} la réponse sans les tags
Commentaire : Utiliser pour récupérer les réponses pour les tests unitaires	

buttonCSV(chaîne)

Résumé : enlève les tags html (< > etc...)	
Entrée : {Array} liste d'objet	**Sortie :** Aucun
Commentaire : RAs	

generatorCSV(value)

Résumé : Fonction activée par le bouton, permet de générer un csv) partir de la liste donné	
Entrée : {Array} liste d'objet	**Sortie :** CSV
Commentaire : RAS	

History.js

Au chargement de la page, l'historique est créé vide sauf si un précédent historique est enregistré dans le session storage du navigateur.

History()

Résumé : Ajoute le message dans la barre de recherche à l'historique (ou change sa position si déjà présente).	
Entrée : Aucune.	**Sortie :** Aucune.
Commentaire : RAS	

Levenshtein.js

calculateLevDistace(src,tgt)

Résumé : Calcule la distance de Levenshtein (nombre de caractères à modifier, supprimer ou ajouter pour passer d'une chaine à une autre) entre les deux chaînes passées en entrée.	
Entrée : {String} Première chaine. {String} Deuxième chaine.	**Sortie :** {Number} Distance.
Commentaire : RAS	

isSame(str1,str2)

Résumé : Détermine si deux mots sont identiques selon la distance de Levenshtein.	
Entrée : {String} {String} Les deux chaînes à comparer.	**Sortie :** {Boolean} True si oui, false si non.
Commentaire : Deux mots seront considérés comme identiques si leur distance est inférieure ou égale à 2.	

median.js

median(values)

Résumé : Retourne la valeur médiane d'une liste de valeurs numériques.	
Entrée : {Array} Liste de valeurs.	**Sortie :** {Number} Valeur médiane.
Commentaire : RAS	

parser.js

parserXMLResponse(response)

Résumé : Transforme une structure XML en une structure JSON.	
Entrée : {XML} Structure réponse de Sesame.	**Sortie :** {JSON} Structure transformée.
Commentaire : RAS	

matchTag(tag)

Résumé : Associe un champ de la structure XML de réponse de Sésame avec une version en langage naturel.	
Entrée : {String} Nom du champ	**Sortie :** {String} Version en langage naturel.
Commentaire : Pas utilisée dans le code.	

matchResolver(resolver)

Résumé : Retourne le nom de la base dans laquelle Sesame est allé chercher les données.	
Entrée : {XML} Structure de réponse de Sésame.	**Sortie :** {String} Nom de la base.
Commentaire : Pas utilisée dans le code.	

parseXml(xml)

Résumé : Transforme une structure XML quelconque en une structure JSON	
Entrée : {XML} Structure.	**Sortie :** {JSON}
Commentaire : RAS	

RECORD.JS

Fonction qui gère l'enregistrement pour pouvoir faire la reconnaissance vocale

recordButton et stopButton → bouton qui gère l'activation et le stop de l'enregistrement

HANDLERFUNCTION(STREAM)

Résumé : créer le lecteur vidéo et envoie les données au serveur	
Entrée : {Stream} Donnée d'enregistrement	**Sortie :** Aucun
Commentaire : RAS	

REPTESTU.JS

Contient les réponses du tests unitaires

SENDTOSERVER.JS

Contient toutes les fonctions envoyant des données aux serveurs

SAVETEXTASFILE(TEXT)

Résumé : Enregistre les textes en local	
Entrée : {String} Text	**Sortie :** Aucun
Commentaire : RAS	

SENDLOGSTOSERVER(LOGS)

Résumé : Enregistre les logs sur le serveur	
Entrée : {String} logs	**Sortie :** Aucun
Commentaire : RAS	

SENDCOMMENTSTOSERVER(LOGS, COMMENTS)

Résumé : Enregistre les logs et les commentaires sur le serveur	
Entrée : {String} logs {String} commentaires	**Sortie :** Aucun
Commentaire : Plus utilisé on pasee maintenant par un form (notation.htl) pour récupérer les commentaires des utilisateurs	

DEMOREPORT(REPORT)

Résumé : Enregistre le compte rendu de la démo	
Entrée : {String} logs	**Sortie :** Aucun
Commentaire : RAS	

DOWN(FILE)

Résumé : Envoi l'enregistrement audio du recorder au serveur pour	
Entrée : {Blob) audio file	**Sortie :** Aucun
Commentaire : RAS	

ASKOTYPESIMBAD(VALUE)

Résumé : envoie au serveur un numéro de otype à chercher dans le csv Simbad pour en récupérer l'objet type	
Entrée : {Float} numéro	**Sortie :** {string} otype associé
Commentaire : RAS	

ASKOTYPEVIZIER(VALUE)

Résumé : envoie au serveur un numéro de otype à chercher dans le csv Vizier pour en récupérer l'objet type	
Entrée : {Float} numéro	**Sortie :** {string} otype associé
Commentaire : RAS	

TESTUNIREPORT(REPORT)

Résumé : Enregistre le compte rendu dees tests unitaires	
Entrée : {String} reprot	**Sortie :** Aucun
Commentaire : RAS	

SPEECHRECOGNITION.JS

Contient une autre façon de faire de la reconnaissance vocale avec l'utilisation de Speech Recognition de Chrome (mettre les fonctions startr() et stopr() sur les boutons précédents)

SPEECHSYNTHESIS.JS

Gère la synthèse vocale du programme

TESTUNITAIRE .JS

Contient les fonctions gérant les tests unitaires. Les test unitaires sont faits à partir de trois matrices que l'on croise pour former des requêtes.

Variables globales :

measureTEST, targets → matrices

testMatrix = association possible entre les deux matrices précédentes (1 = oui, 0 = non)

testu, testuDf → savoir si on est en train d'effectuer un test unitaire ou pas.

Errorf → stcoke les questions qui ont produit des erreurs

RUNTESTUNITAIRE()

Résumé : Fonction qui gère le test unitaire	
Entrée : {Stream} Donnée d'enregistrement	**Sortie :** Aucun
Commentaire : RAS	

CHECKTESTU()

Résumé : Fonction qui compare les résultats obtenus par les questions et les réponses références	
Entrée :	**Sortie :** Aucun

Commentaire : RAS

runTestUnitaireDialogflow()

Résumé : Fonction qui gère le test unitaire (juste pour la partie Dialogflow)	
Entrée :	**Sortie :** Aucun
Commentaire : RAS	

checkTestuDF()

Résumé : Fonction qui compare les résultats obtenus par les questions et les réponses références (juste pour Dialogflow)	
Entrée :	**Sortie :** Aucun
Commentaire : RAS	

Wavelength.js

wl2HiPS(str)

Résumé : Associe une longueur d'onde avec un catalogue applicable sur Aladin-Lite.	
Entrée : {String} Longueur d'onde.	**Sortie :** {String} Nom du catalogue.
Commentaire : RAS	

Main.js

Redimensionne la zone de chat pour qu'elle ne prenne que 80% de l'écran.

validate(message)

Résumé : Fonction appelée quand l'utilisateur appuie sur la touche entrée et envoie un message au chatbot.	
Entrée : {String} Message de l'utilisateur.	**Sortie :** Aucune.
Commentaire : RAS	