User interface : Comprends tous les éléments du navigateur en dehors de la fenêtre d’affichage du site web

Browser engine : browser engine est un composant logiciel de base de tous les principaux navigateurs Web. Le travail principal d'un moteur de navigation consiste à transformer des documents HTML et d'autres ressources d'une page Web en une représentation visuelle interactive sur l'appareil d'un utilisateur.

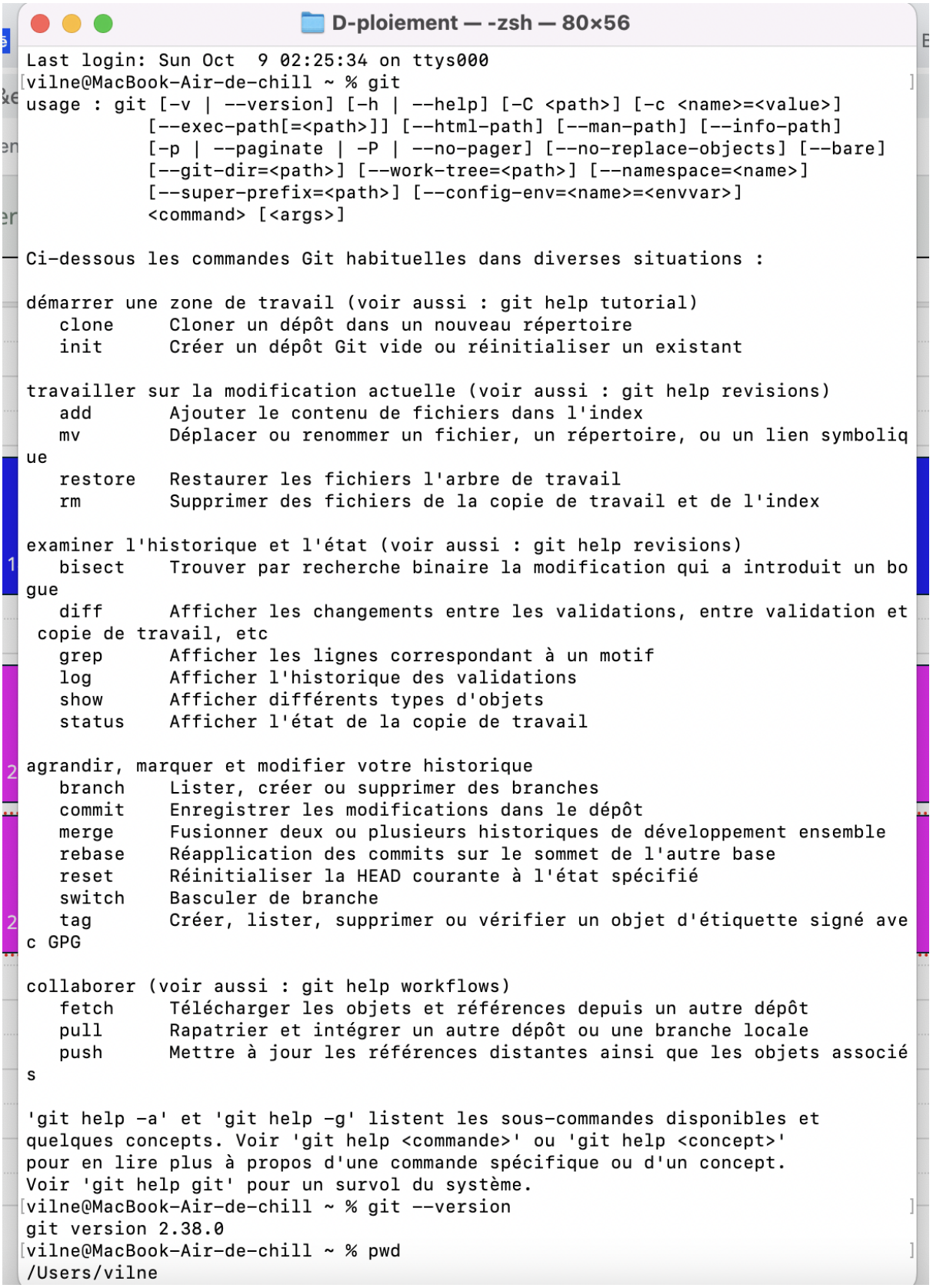
rendering engine : responsable de l'affichage du contenu demandé. Par exemple, si le contenu demandé est HTML, le moteur de rendu analyse HTML et CSS et affiche le contenu analysé à l'écran. Exemple : Blink, Gecko, EdgeHTML, WebKit.

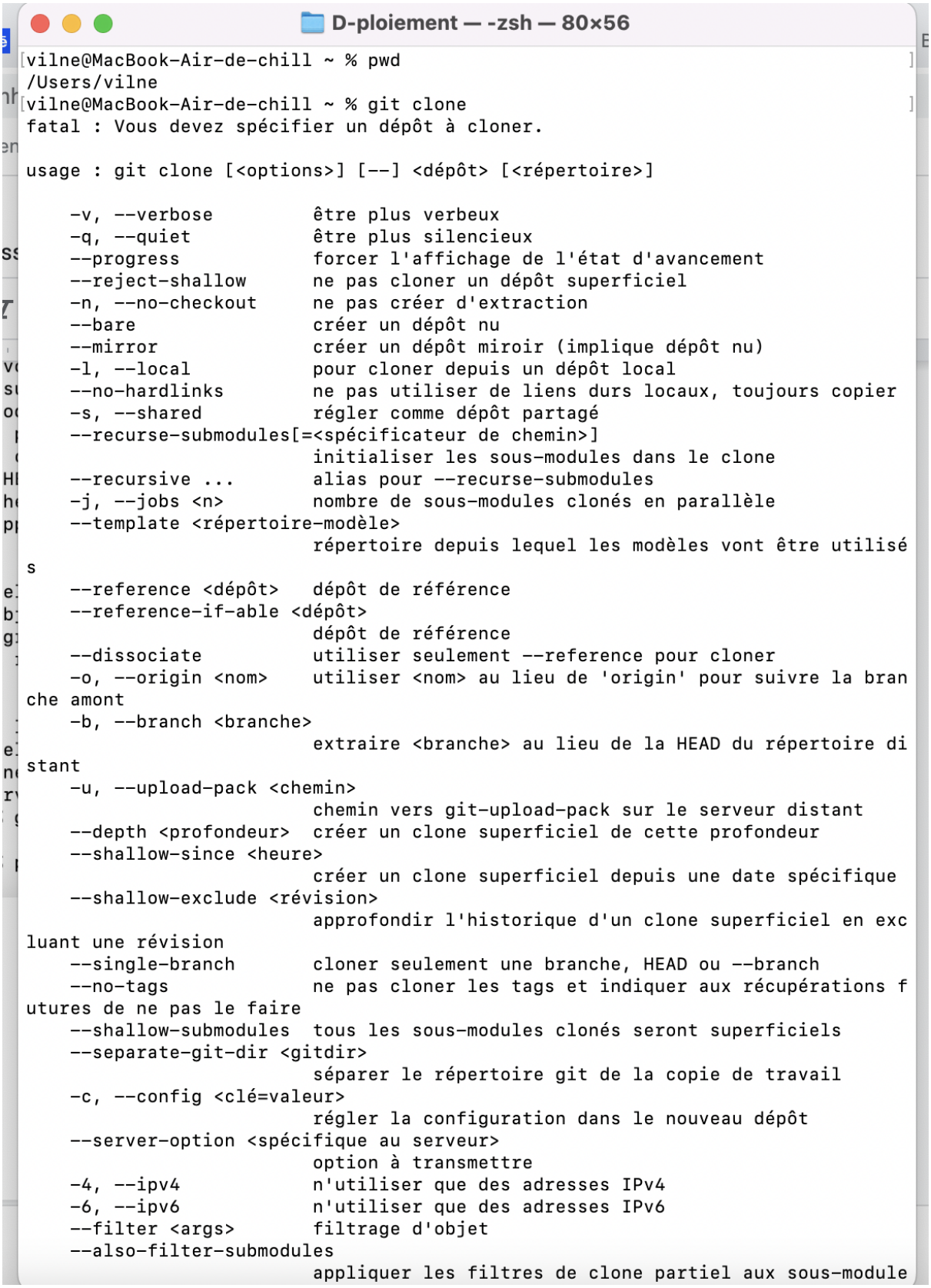
networking : pour les appels réseau tels que les requêtes HTTP, en utilisant différentes implémentations pour différentes plates-formes derrière une interface indépendante de la plate-forme.

javascript interpreter : Utilisé pour analyser et exécuter du code JavaScript.

UI Backend : utilisé pour dessiner des widgets de base comme des listes déroulantes et des fenêtres. Ce backend expose une interface générique qui n'est pas spécifique à la plate-forme. En dessous, il utilise des méthodes d'interface utilisateur du système d'exploitation.

DATA Persistance : Il s'agit d'une couche de persistance. Le navigateur peut avoir besoin d'enregistrer localement toutes sortes de données, telles que des cookies. Les navigateurs prennent également en charge les mécanismes de stockage tels que localStorage, IndexedDB, WebSQL et FileSystem.


D-ploiement — -zsh — 80×56
Last login: Sun Oct 9 02:25:34 on ttys000
vilne@MacBook-Air-de-chill ~ % git
usage : git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
[--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
[-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
[--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
[--super-prefix=<path>] [--config-env=<name>=<envvar>]
<command> [<args>]
Ci-dessous les commandes Git habituelles dans diverses situations :
démarrer une zone de travail (voir aussi : git help tutorial)
clone Cloner un dépôt dans un nouveau répertoire
init Créer un dépôt Git vide ou réinitialiser un existant
travailler sur la modification actuelle (voir aussi : git help revisions)
add Ajouter le contenu de fichiers dans l'index
mv Déplacer ou renommer un fichier, un répertoire, ou un lien symboliq
ue
restore Restaurer les fichiers l'arbre de travail
rm Supprimer des fichiers de la copie de travail et de l'index
examiner l'historique et l'état (voir aussi : git help revisions)
bisect Trouver par recherche binaire la modification qui a introduit un bo
gue
diff Afficher les changements entre les validations, entre validation et
copie de travail, etc
grep Afficher les lignes correspondant à un motif
log Afficher l'historique des validations
show Afficher différents types d'objets
status Afficher l'état de la copie de travail
agrandir, marquer et modifier votre historique
branch Lister, créer ou supprimer des branches
commit Enregistrer les modifications dans le dépôt
merge Fusionner deux ou plusieurs historiques de développement ensemble
rebase Réapplication des commits sur le sommet de l'autre base
reset Réinitialiser la HEAD courante à l'état spécifié
switch Basculer de branche
tag Créer, lister, supprimer ou vérifier un objet d'étiquette signé ave
c GPG
collaborer (voir aussi : git help workflows)
fetch Télécharger les objets et références depuis un autre dépôt
pull Rapatrier et intégrer un autre dépôt ou une branche locale
push Mettre à jour les références distantes ainsi que les objets associé
s
'git help -a' et 'git help -g' listent les sous-commandes disponibles et
quelques concepts. Voir 'git help <commande>' ou 'git help <concept>'
pour en lire plus à propos d'une commande spécifique ou d'un concept.
Voir 'git help git' pour un survol du système.
vilne@MacBook-Air-de-chill ~ % git --version
git version 2.38.0
vilne@MacBook-Air-de-chill ~ % pwd
/Users/vilne


D-ploiement — -zsh — 80×56
vilne@MacBook-Air-de-chill ~ % pwd
/Users/vilne
vilne@MacBook-Air-de-chill ~ % git clone
fatal : Vous devez spécifier un dépôt à cloner.
usage : git clone [<options>] [--] <dépôt> [<répertoire>]
-v, --verbose être plus verbeux
-q, --quiet être plus silencieux
--progress forcer l'affichage de l'état d'avancement
--reject-shallow ne pas cloner un dépôt superficiel
-n, --no-checkout ne pas créer d'extraction
--bare créer un dépôt nu
--mirror créer un dépôt miroir (implique dépôt nu)
-l, --local pour cloner depuis un dépôt local
--no-hardlinks ne pas utiliser de liens durs locaux, toujours copier
-s, --shared régler comme dépôt partagé
--recurse-submodules[=<spécificateur de chemin>]
initialiser les sous-modules dans le clone
--recursive ... alias pour --recurse-submodules
-j, --jobs <n> nombre de sous-modules clonés en parallèle
--template <répertoire-modèle>
répertoire depuis lequel les modèles vont être utilisés
--reference <dépôt> dépôt de référence
--reference-if-able <dépôt>
dépôt de référence
--dissociate utiliser seulement --reference pour cloner
-o, --origin <nom> utiliser <nom> au lieu de 'origin' pour suivre la branche amont
-b, --branch <branche>
extraire <branche> au lieu de la HEAD du répertoire distant
-u, --upload-pack <chemin>
chemin vers git-upload-pack sur le serveur distant
--depth <profondeur> créer un clone superficiel de cette profondeur
--shallow-since <heure>
créer un clone superficiel depuis une date spécifique
--shallow-exclude <révision>
approfondir l'historique d'un clone superficiel en excluant une révision
--single-branch cloner seulement une branche, HEAD ou --branch
--no-tags ne pas cloner les tags et indiquer aux récupérations futures de ne pas le faire
--shallow-submodules tous les sous-modules clonés seront superficiels
--separate-git-dir <gitdir>
séparer le répertoire git de la copie de travail
-c, --config <clé=valeur>
régler la configuration dans le nouveau dépôt
--server-option <spécifique au serveur>
option à transmettre
-4, --ipv4 n'utiliser que des adresses IPv4
-6, --ipv6 n'utiliser que des adresses IPv6
--filter <args> filtrage d'objet
--also-filter-submodules
appliquer les filtres de clone partiel aux sous-module

```
s
    --remote-submodules   tous les sous-modules clonés utiliseront leur branche
de suivi à distance
    --sparse              initialiser le fichier d'extraction clairsemée pour n'
inclure que les fichiers à la racine
    --bundle-uri <uri>    un URI pour télécharger des paquets avant de récupérer
 depuis le distant d'origine

vilne@MacBook-Air-de-chill ~ % git clone https://github.com/vilne92/D-ploiement.
git
Clonage dans 'D-ploiement'...
avertissement : Vous semblez avoir cloné un dépôt vide.
vilne@MacBook-Air-de-chill ~ % ls
Applications    Documents       Movies          Public
D-ploiement     Downloads       Music           Sites
Desktop         Library         Pictures        redshift
vilne@MacBook-Air-de-chill ~ % cd D-ploiement
vilne@MacBook-Air-de-chill D-ploiement % git statusq
git : 'statusq' n'est pas une commande git. Voir 'git --help'.

La commande la plus ressemblante est
        status
vilne@MacBook-Air-de-chill D-ploiement % git status
Sur la branche main

Aucun commit

rien à valider (créez/copiez des fichiers et utilisez "git add" pour les suivre)
vilne@MacBook-Air-de-chill D-ploiement %
```