Web Designer

Constat

- Web peu efficace
  - Performance
  - Esthétiquement limité (complexité) *(de base)*
  - Développement lent et instable

- Nécessité de « simplifier » le web
  - Développement plus simple/plus rapide
  - Ne plus réinventer la roue
  - Créer des outils/environnements tout-en-un

Réponse peu satisfaisante

- Réponse actuelle
  - Développement des Frameworks Front-End (simplifier le Front-End)
  - Popularisation de NodeJS (« unification » par le Front-End des langages)

    ≈ faire disparaitre/cacher le back-end et ses contraintes (performances, testing, contraintes réseaux, etc.)

- Web peu efficace
  - Performance
  - Esthétiquement limité (complexité)
  - Développement lent et **instable**

Résolution **théorique** de seulement **la moitié des problèmes**…

Réponse peu satisfaisante

- **Problème de performance** :

« *Comment en une seule seconde, je peux produire 60 images de 8 millions de pixels, et ne pas réussir à charger une page web de pur texte dans un navigateur ?* »

  - Problématiques réseaux abstraites du développeur
  - Pas d'algorithmique – Pas de théorisation
    - Décorrélation entre la réalité physique de l'ordinateur et le développement du logiciel *(Code pas cache-friendly par exemple)*
    - Peu ou pas d'optimisation *(surutilisation du cache du navigateur, taille mémoire, exécution du code, etc.)*

  - Mentalité Consumériste/Je-m'en-foutiste : « J'ai de la ressource, autant l'utiliser ! »
    - Présuppose que tout le monde possède une machine surpuissante et ne va que sur un seul site
    - Posture sociale et idéologique :

« Aujourd'hui on est passé à autre chose : s'occuper des performances = ne pas s'occuper du design» *(aka ne pas être moderne)*

« Les performances c'est bon pour les devs C des années 1980 qui utilisent des patates comme PC : aujourd'hui, tout le monde a une bonne machine»

Réponse peu satisfaisante

- **Problème d'esthétique**:

  « *Comment faire en sorte que mon site ne ressemble pas à une bouse des années 1990-2000 ?* »

  - C'est un faux problème
    - Ça n'est qu'une question de design, de couleurs, etc.
    - Aujourd'hui il n'y a pas de limitation techniques

  - Réel problème
    - Complexité du CSS (Positionnement, Gestion des styles)
    - Instabilité du CSS (Erreurs silencieuses)
    - Peu d'évolutions marquantes du CSS

    → CSS = un langage de développeurs pas de designer

Réponse peu satisfaisante

- **Problème de vitesse de développement** :

  « *Pourquoi faut-il si longtemps pour développer un site web et si peu pour en avoir l'idée ?* »

  - Redondance forte / Peu d'automatisation
    - Écriture du HTML difficilement automatisable *(faisable mais souvent plus long que de l'écrire)*
    - CSS langage peu formel (encourage la redondance « au cas où »)
    - Il faut coder chaque comportement manuellement

    → C'est comme demander au menuisier d'expliquer à chaque client comment utiliser une porte

  - Mélange des compétences
    - Mix entre compétence techniques (développement, logique) et artistiques (design, mise en forme, UX)
    - Compétences souvent « socialement-incompatibles » (rare sont les développeurs artistes ou les artistes développeurs)
    - Compétences à haut degré de connaissance

Web en général

- **Problème plus global** :

  « *Pourquoi a-t-on arrêté de réfléchir dans le monde du développement web ?* »

  - Décalage fort entre l'informatique à l'université et en « entreprise »

  - Abandon des réflexions
    - Sur le stockage des données (BD obligatoire) => *Quid des systèmes de fichiers ? Quid des données applicatives ?*
    - Sur les performances
    - Sur l'architecture générale *(Client-Serveur toujours pertinent en 2020 ?)*
    - Phénomène de mode qui l'emporte sur le reste

Web en général

- **Enjeux énormes !**

« *Pourquoi gaspiller autant de ressources ? (mémoire, électrique, temps de travail, etc.)* »

« *Pourquoi une technologie aussi basique (mise-en-forme de texte) est-elle devenue si complexe ?* »

- **De vraies questions à se poser :**

« *N'a-t-on pas poussé le web trop loin ?* »

⇒ Trouver une alternative pour faire des applications sans clients à installer et accessibles sur Internet ?

(Est-ce qu'HTML est pertinent aujourd'hui pour des applications web lourde ?)

« *Que peut-on faire pour améliorer les choses ?* »

=> A défaut de changer le monde du web : il faut penser une nouvelle façon de développer en web/développer un outil pour tous

Idée Générale

- Une façon simple et efficace de faire du web (accessible à tous)
- Un outil aussi bien pour les designers que les développeurs !
- Un Concepteur d'IHM/IDE/Debugger moderne (Tout en un)
- Sortir de la logique du Framework web
  - Pour des raisons de performances
  - Plus d'utilité (component intégrés directement)
  - Retour à un HTML-CSS-Javascript

- On cherche à créer un outil pour faire ce que fait un web designer !

=> Le web designer

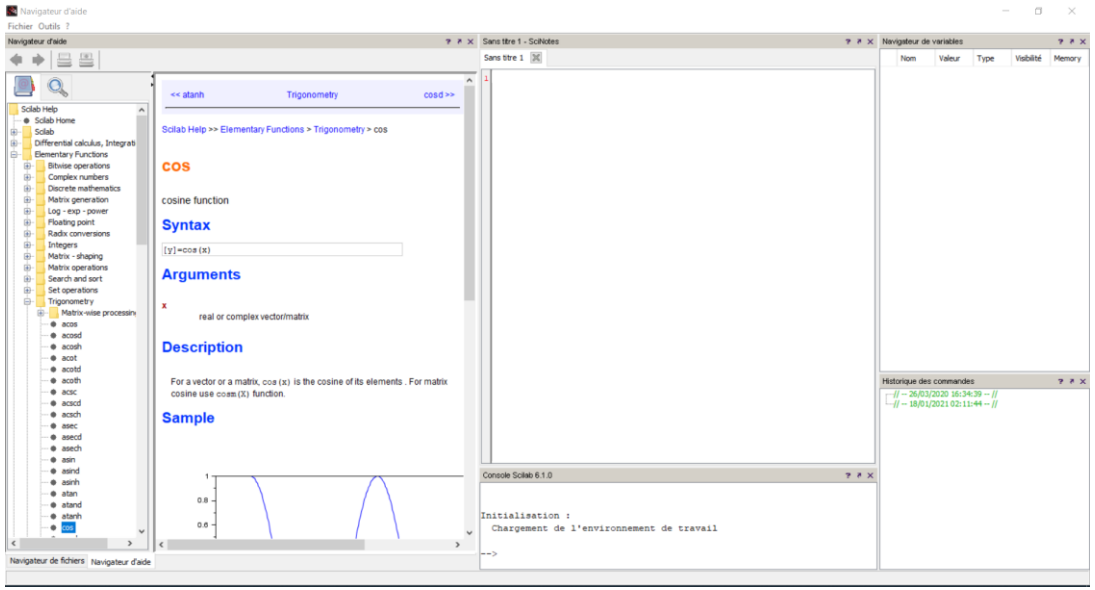
Implications

- Une façon simple et efficace de faire du web (accessible à tous)
  - Être capable **d'abstraire la programmation du web**
  - Être capable de produire/de **générer du « code web » (HTML-CSS)**

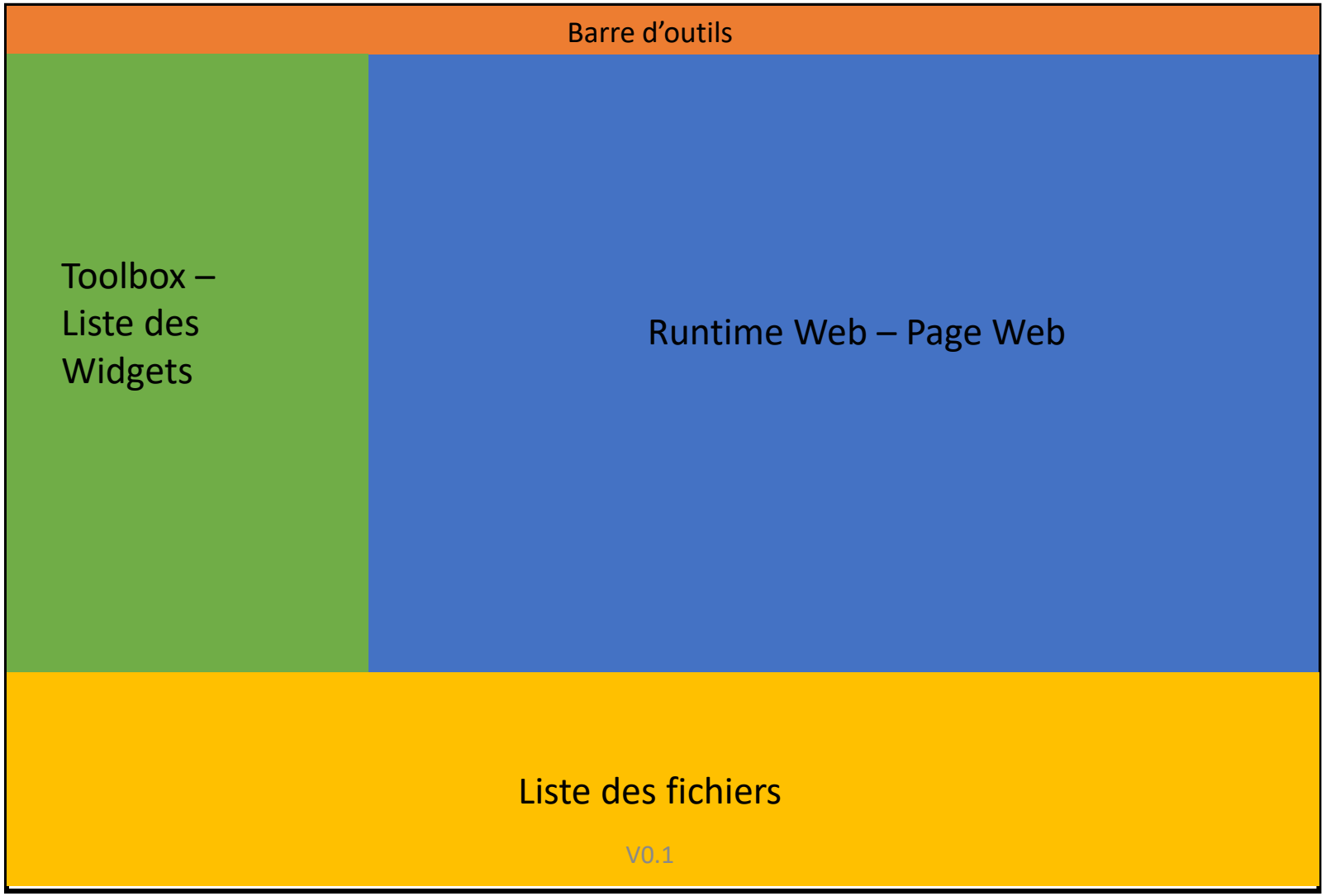
- Un outil aussi bien pour les designers que les développeurs !
  - **Définir les besoins** de chaque groupe
  - Créer un **logiciel** assez **flexible**/customisable sans être limitant

Interface

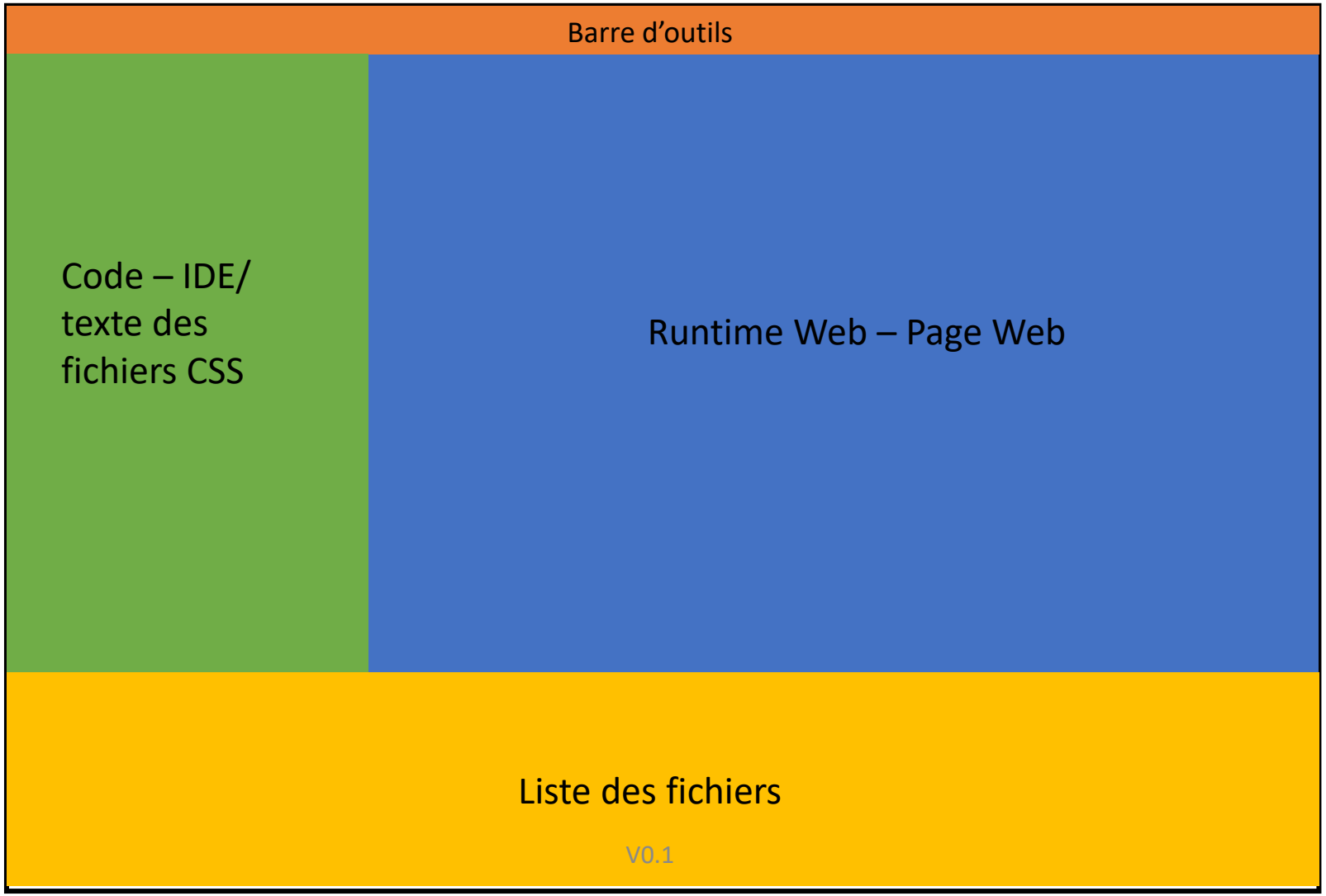
- Créer une interface riche et flexible
- Inspirée de l'interface Scilab
  - Gérer un **ensemble de panneaux** qui remplissent des fonctions précises
- Enregistrement de preset/configurations ?
- Barre d'outil qui s'adapte selon la fenêtre active ?



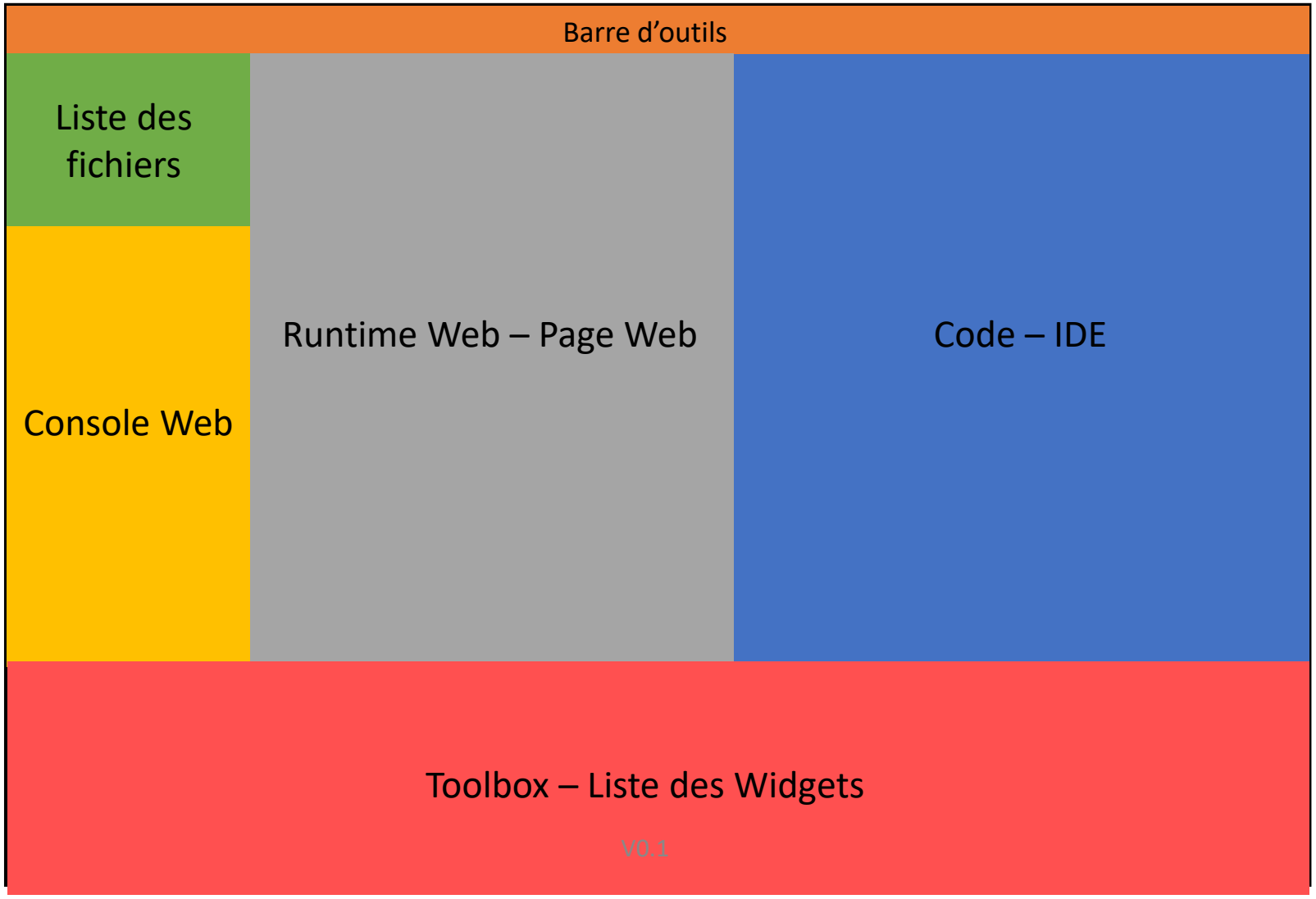
Interface - Designer



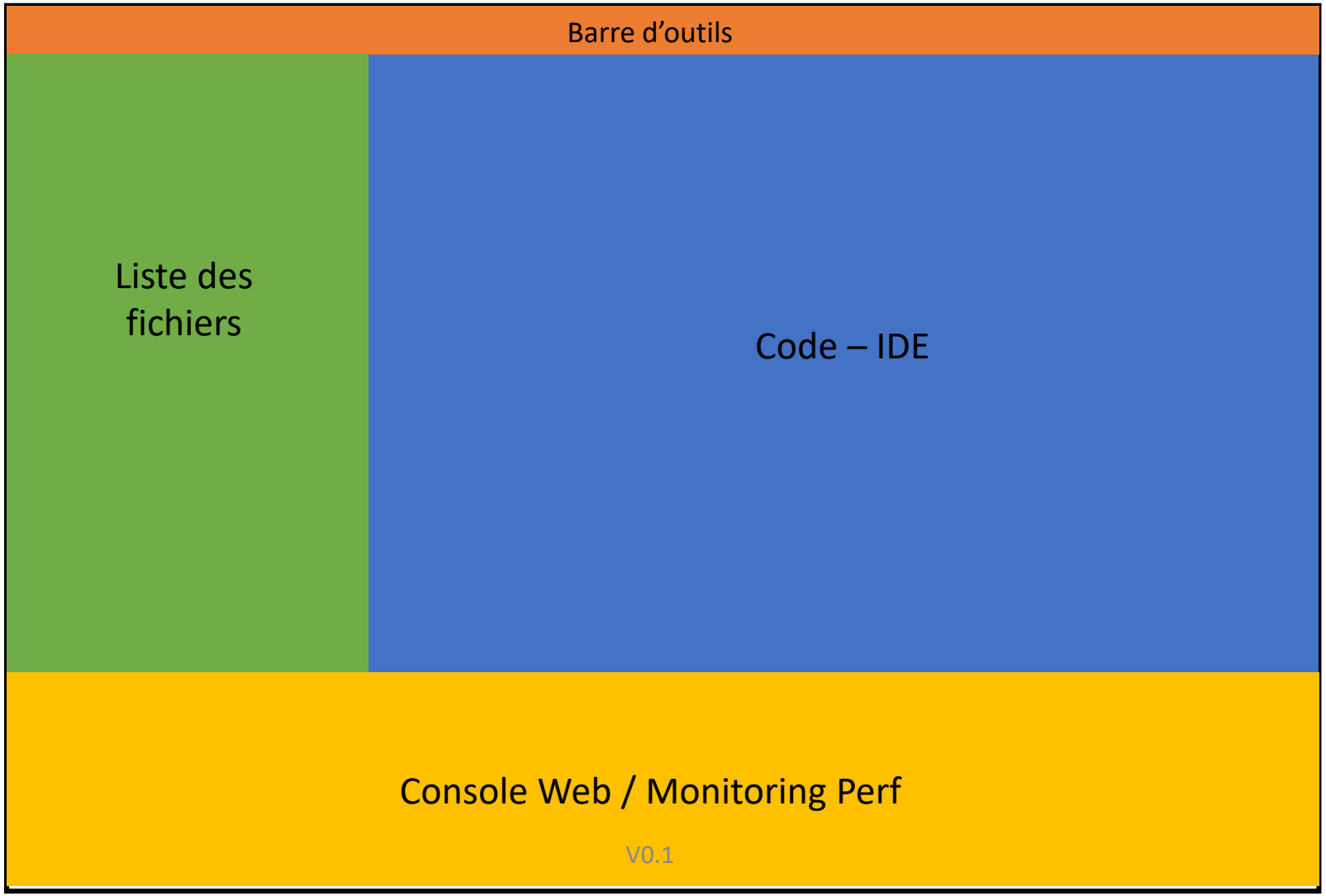
Interface – « Technical Artist »



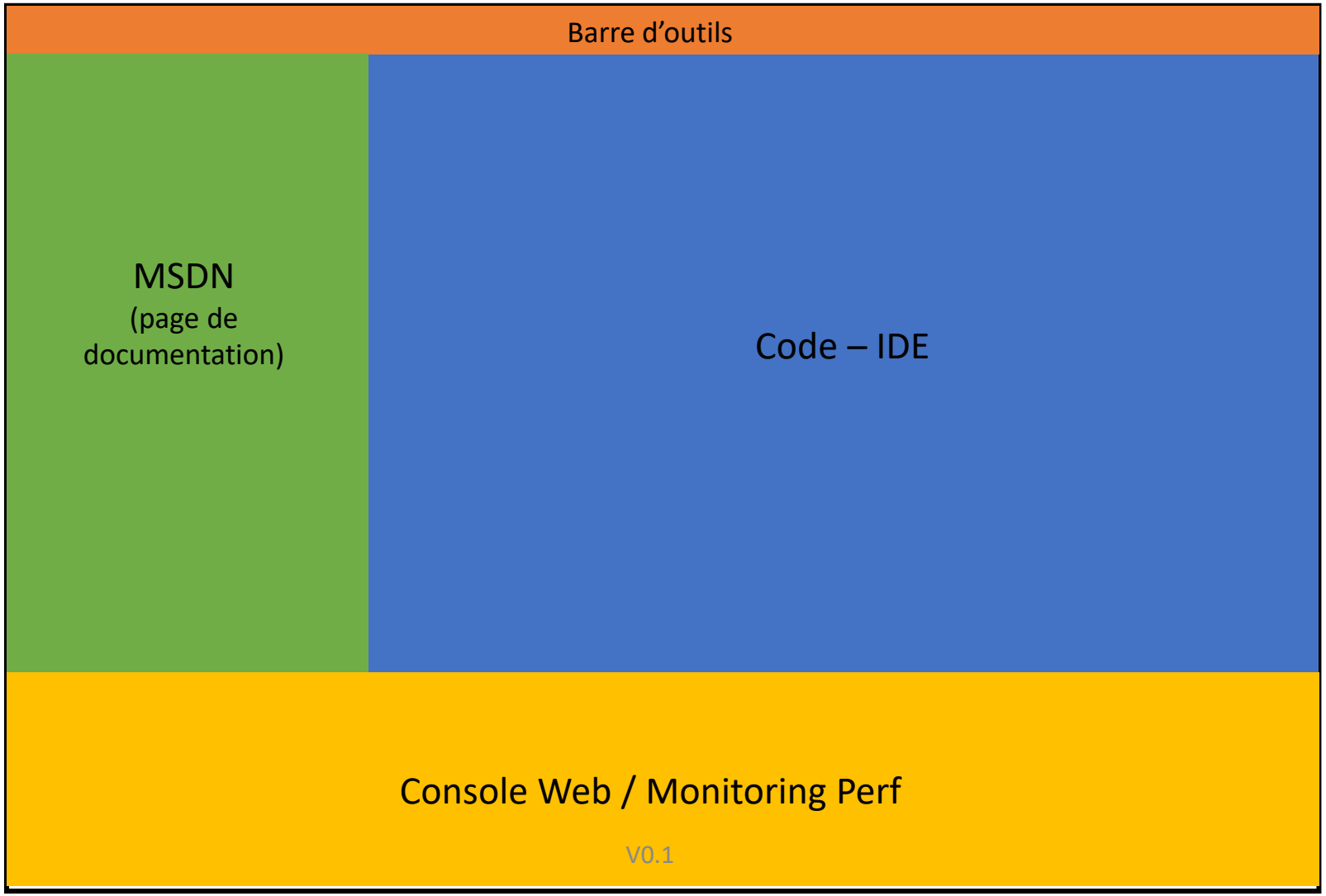
Interface – « Technical Artist 2 »



Interface – Développeur



Interface – Développeur Bis



Styles

- Revenir à l'essence d'un style :

  Regrouper un ensemble cohérent sous une même direction artistique

  ≠ mettre un style différent pour chaque élément !

- Revenir à la définition artistique ?

> **II** (1699). **Dans les arts de l'espace et du temps.** Manière particulière (personnelle ou collective) de traiter la matière et les formes en vue de la réalisation d'une œuvre d'art; ensemble des caractères d'une œuvre qui permettent de la classer avec d'autres dans un ensemble constituant un type esthétique. | *Style pastiché* (cit. 2), *imité par un faussaire* (cit. 5). | *Le style d'un peintre, d'une école.* → **Facture, faire** (cit. 227.1 et *supra*), **genre, goût, manière, touche** (→ Conjonction, cit. 3; écriture, cit. 13; imitateur, cit. 5). — *Le style d'un dessin, d'un tableau* (→ Dénoter, cit. 3; naïf, cit. 5). | *Style d'esquisse* (cit. 2). | *Style d'une statue* (→ Fouille, cit. 1;

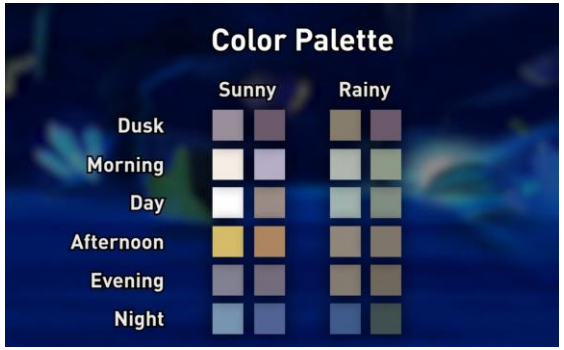
- Pistes :
  - Mix entre CSS dans le HTML (personnalisé/styleless) et Style
  - Créer des superstyles (des ensembles qui regroupent plusieurs styles CSS)
    - Pour faire des palettes
    - Pour faire des organisations riches (positionnements)
    - Pour donner plus de sémantique/plus de modularité au projet

Superstyles

- Souvent dans le design les styles ne sont pas isolés mais ont un lien fort entre eux :

  Un « Superstyle » est un ensemble/une composante cohérent(e) d'un design

  - Palette de couleurs pour un designer



  - Organisation générale des informations
    - Avoir un style épuré
    - Avoir un style informatif
    - Etc.

Enregistrement d'un « Design »

- Un « Design » = ensemble des composantes artistiques du site
  - Couleur, placement, etc.
  - Travail indépendant du contenu

- Faire des preset de superstyles ?
  - Combiner l'ensemble des composantes stylistiques sous un seul preset (enregistrable)
  - Rendre réutilisable le travail de l'artiste/UI/UX designer

    (On aimerait dans l'idéal pouvoir passer sans trop d'effort d'un design à un autre/Appliquer un même design à plusieurs sites)

Système de layer

- Faciliter la superposition d’éléments

Liste des panneaux possibles

- IDE – Éditeur de Texte/code
- Runtime Web – Page Web
- Console Web – Monitoring Perf
- Explorateur de fichiers – Liste de fichiers
- Toolbox – Liste des widgets
- Panneau de config d’un widget sélectionné
- Documentation ? (Onglet Web ou intégré ?)
- Panneaux gestion des versions
- Etc.

IDE – Éditeur de Texte/code

- Contient:
  - Coloration syntaxique
  - Système de vérification de balisage (cf. fermeture d'accolades)
  - Overview fichier
  - Raccourcis basiques
  - Analyse « sémantique » basique (cf. Intellisense sur VS)

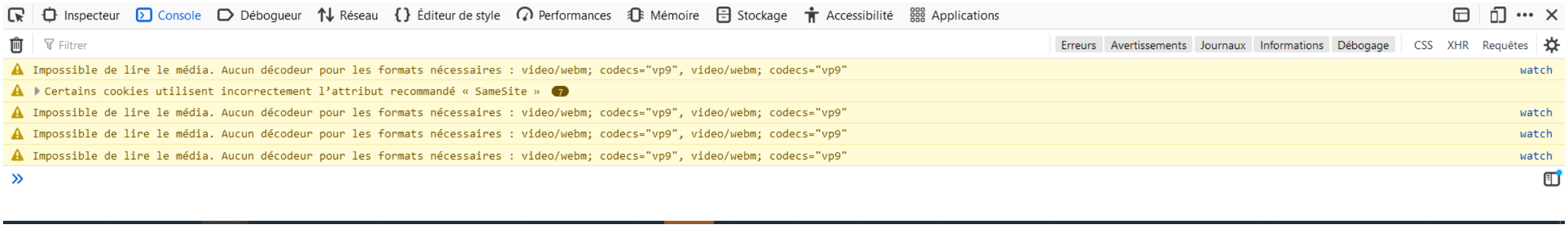
```
73          return (raytracingType)r;
```

Runtime Web – Page Web

- Lance un serveur localhost ?
- Juste rendu d’une page web ?

Console Web – Monitoring Perf

- La console de base de n'importe quel navigateur web
- Inclure la partie performance de la console ? (les mettre ensemble ou les séparer)
- Inclure la partie inspecteur de la console ? (les mettre ensemble ou les séparer)
- Principal outil de monitoring/débug ?



Explorateur de fichiers – Liste de fichiers

- Alexis

Toolbox – Liste des widgets

Config des widgets

Documentation

CI/CD dans le web designer

- Intégrer un outil pour gérer les versions depuis le web designer
  - Basique : Panneau Git
  - Outil plus poussé ?

- Automatiser le CI/CD ?
  - = Automatiser l'automatisation
  - Complètement nouveau aujourd'hui (pas de concurrence)
  - Porte ouverte vers gestion lien back-end (rendre le logiciel plus complet)
  - Beaucoup de pistes possibles

- Quelques idées….

Piste 1 : générateur de pipeline de déploiement

- Générer - selon des critères précis - un système complet de déploiement adapté
  - On peut penser des « kits/preset » de déploiement (pour simplifier)
  - S'adapter selon les besoins
  - Enfin faire du propre dans la gestion du déploiement

- Concrètement :
  - Génération des scripts de build (à un niveau basique) => (pom.xml, etc.)
  - Automatisation de l'allocation des ressources adaptées (dockers, VM, etc.)
  - Gestion clean des credentials et différents comptes
  - Génération des scripts de déploiement (et leurs fichiers de config)

Piste 2 : générateur de serveur web/API REST

- Générer – en fonction des ressources – un serveur REST basique
  - Complexité logique derrière le REST limitée => Simple à automatiser + intérêt à l'automatiser grand
  - Générer selon des critères (langage de prog, techno, performances, etc.) le serveur adapté.

Piste 3 : Panneau gestion des versions