Web Developer
Scripts Clients

01 – Introduction

A. Martel 2025-2026

Généralités – Applications

- Pages Web
- Scripts principalement côté client (objet de ce cours) :
  - Modifier l'apparence de la page après chargement
  - Lancer des requêtes sur événement (via AJAX)
  - Interagir avec l'utilisateur
  - Développer des applications complètes : React, etc.
- Mais également côté serveur (via Node.js)

Généralités – Historique

- Créé en 1995 par Brendan Eich (Netscape)
- Langage de programmation de scripts orienté objet à prototype
- Standardisé par l'ECMA en 1997 – dernière version : ECMAScript 2024

Généralités – Interpréteurs

- Langage interprété
- Interpréteurs embarqués dans les navigateurs :
  - JavaScriptCore (Safari)
  - V8 (Chrome, new Edge, Opera)
  - SpiderMonkey (Firefox)
- Node.js utilise aussi V8 (hors navigateur)

Généralités – EDI

- EDI utilisé : Webstorm ou VS Code
- Tests dans Chrome ou Firefox
- Référence JavaScript via Mozilla
- Tests JS seul via JSFiddle

Généralités – Premier projet

- Ouvrons notre éditeur
- Créons un premier projet
- Ajoutons un dossier js
- Créons le fichier js/main.js dont le contenu sera :

```
alert('Hello World !');
```

Généralités – Premier projet

Dans notre page html, nous devons charger le fichier js que nous venons de créer :

```
    …
    <script src="js/main.js"></script>
</body>
```

Généralités – Premier projet

Il est également possible de placer du code JS directement dans la page html :

```
    …
    <script>
        alert('Hello World !');
    </script>
</body>
```

Pour des raisons de gestion du code (réutilisation) et du cache, il est préférable d'utiliser la méthode via un fichier JS.

Généralités – Premier projet

Il nous reste à tester notre premier script en ouvrant la page html dans un navigateur.

Généralités – Premier projet

Nous venons d'afficher une boîte de dialogue à l'aide de la fonction alert.

Il existe d'autres boîtes de dialogue que nous verrons très bientôt.

Si nous désirons afficher un message sur plusieurs lignes, nous pouvons utiliser le caractère \n.

```
alert('Hello\nWorld !');
```

Généralités – Instructions

En JavaScript, une instruction est toujours terminée par un point-virgule, sauf s'il s'agit d'un bloc d'instructions :

```
instruction1;
instruction2;
instruction3;
bloc1 {
    …
}
```

Le point-virgule n'est pas obligatoire dès lors que les instructions sont placées sur des lignes différentes.

Le respect de cette contrainte constitue une règle de bonne pratique.

Généralités – Indentation

Afin de permettre une lecture plus aisée, il est nécessaire d'indenter le code.

L'indentation correspond au « recul » des instructions par rapport à la marge gauche :

```
bloc1 {
   instruction1;
   bloc2 {
      instruction1;
   }
}
```

Généralités – Indentation

Idéalement, une indentation correspond à 4 espaces.

Surtout ne pas utiliser des tabulations.

La plupart des éditeurs modernes remplacent les tabulations par des espaces.

Ici, les espaces sont matérialisées par des astérisques :

```
bloc1 {
****instruction1;
****bloc2 {
********instruction1;
****}
}
```

Généralités – Indentation

Après chaque début de bloc, on ajoute une indentation.

Après chaque fin de bloc, on retire une indentation.

Généralités – Commentaires

Pour documenter notre code, il est possible d'y ajouter des commentaires :

```
/* Commentaire
   Sur plusieurs
   Lignes */

// Commentaire sur une seule ligne
```

Généralités – Console

Pour afficher les résultats de l'exécution de notre code, nous pouvons également utiliser la console JS via Chrome.

Il s'agit d'un outil intégré aux différents navigateurs

Très utile pour déboguer et afficher des informations.

La fonction à utiliser pour y afficher du texte est :

```
console.log('Hello World dans la console');
```

Bonnes pratiques

- Séparer le code JS du HTML : utiliser un fichier `.js` externe plutôt que <script> intégré
- Charger le script correctement : en bas du <body>
- Déboguer avec console.log() : éviter alert(), trop intrusif
- Écrire un code lisible : indentation (4 espaces), point-virgule en fin de ligne
- Tester régulièrement et lire la console : détecter tôt les erreurs
- Consulter la documentation officielle : developer.mozilla.org