LES 22 ÉTAPES ESSENTIELLES POUR CODER UNE SANDBOX EN JAVASCRIPT

la 21ème va vous surprendre !

Bonjour !

- Gildas Lormeau
- Développeur JS depuis 10+ ans
- Auteur d'extensions pour Chrome : JSONView, SingleFile
- Auteur de la librairie zip.js
- Co-fondateur de seo4ajax.com

Besoins

- Permettre l'interprétation dans un navigateur de code fourni par l'utilisateur
- La sandbox doit :
  - Fonctionner dans Chrome et les autres navigateurs modernes
  - Être accessible via un appel de fonction similairement à `eval`
  - N'accepter en paramètre que des expressions
  - Empêcher l'accès aux APIs autres que celles disponibles dans ES5 ou ES6 (e.g. APIs DOM & BOM)
  - Empêcher l'accès aux variables globales, et à `this`, `self`, `eval`, `setTimeout`, `setInterval`
  - Exécuter le code de manière synchrone
  - Être sécure !

Exemples d'appels

- `safeEval("42") // 42`
- `safeEval("window") // undefined`
- `safeEval("this") // undefined`
- `safeEval("document") // undefined`
- `safeEval("{ let x = 42 }") // Uncaught SyntaxError: Unexpected identifier`
- `safeEval("eval('42')") //  Uncaught ReferenceError: eval is not defined`


CODONS !

Extension « Yet Another Blocker »

- Extension de type AdBlock/Ghostery etc.
- Permettre le bloquage et/ou la transformation de requêtes ou réponses HTTP
- Règles définies sous forme de fonctions JavaScript
- Objectifs :
  - Rendre accessible ~~à l'utilisateur~~ au développeur toute la puissance offerte par les APIs de Chrome
  - Permettre l'écriture de règles de bloquage/filtrage plus avancées que les bloqueurs existants

Exemples de configuration (1/3)

```
{
    defaultRules: [
        "default"
    ],
    rules: {
        default: {
            block: {
                request: [
                    details => details.method != "GET"
                ]
            }
        }
    }
}
```

Exemples de configuration (2/3)

```
{
    defaultRules: [
        "default"
    ],
    rules: {
        default: {
            block: {
                request: [
                    details => details.method != "GET"
                ],
                response: [
                    details => details.url.includes("analytics")
                ]
            }
        }
    }
}
```

Exemples de configuration (3/3)

```
{
    rules: {
        default: {
            block: {
                request: [ details => details.method != "GET" ]
            }
        }
    },
    hosts: {
        "twitter.com": {
            rules: [ "default" ],
            block: {
                request: [ details => details.url.includes("analytics") ]
            }
        }
    }
}
```

Avancement

- Fait :
  - Bloquage/Autorisation des requêtes/réponses
  - Interface utilisateur basique
- RAF :
  - Transformation des requêtes/réponses
  - Enrichissement de l'interface utilisateur
  - Validation de la configuration (via TypeScript ?)
  - Intégration d'un mode debug
  - Mode d'emploi
  - ...