DOM : La Window (fenêtre du navigateur)

En DOM, il éxiste un objet encore au dessus du document c'est **la window**. Cet objet contient pas mal d'information et fonction nous permettant d'interargir avec la fenêtre du navigateur (redimenssionner, refermer, scroller etc ...)

Vous retrouverez la documentation officiel de la window juste ici :

Documentation officiel

Obtenir la taille de la fenêtre

Il est possible d'utiliser la window pour récupérer sa largeur (width) et sa haute (height) :

```
// Obtenir la largeur en pixel de la fenêtre :
window.screenX // number
// Obtenir la hauteur de la fenêtre en pixel :
window.screenY // number
```

Obtenir la position du « scroll »

Il est possible d'utiliser la window pour obtenir la position vertical et horizontal du « scroll » (la barre de défilement) :

```
// Obtenur la position du scroll vertical
window.scrollY
// Obtenir la position du scroll horizontal
window.scrollX
```

Personnaliser le titre de l'onglet

Il est possible de récupérer ou changer le titre de votre onglet (titre de la fenêtre) :

```
// Obtenir le titre de la fenêtre :
window.title
// Changer le titre de la fenêtre :
window.title = 'Nouveau titre !'
```

Fermer la fenêtre

```
// Ferme la fenêtre
window.close()
```

Ouvrir un nouvelle onglet

```
// Ouvre un nouvel onglet
window.open('https://monsite.com')
```

Utiliser le « prompt » de la fenêtre

```
// Ouvre une modal contenant la question et un champ à remplir
// une fois le champs remplie et valider on obtient sa valeur
// dans la constante
const resultat = window.prompt('quelle age avez-vous ?')
```

Redimensionner la fenêtre

```
// On redimensionne la fenêtre en spécifiant la largeur en pixel
// et la hauteur en pixel
window.resizeTo(200, 500)
```

Faire défiler la barre de défillement (scroll)

```
// Bouge la barre de défilement horizontal et vertical à l'emplacement
// donnée en pixel
window.scrollTo(0, 0)
```

Utiliser les timers, interval et boucle de jeux

`setTimeout`

Il est possible graçe à la window d'attendre un peu de temps avant de lancer des instructions. C'est que l'on appel un `timer` :

```
// setTimeout permet d'attendre un nombre de milliseconds donné
// avant de lancer une fonction
window.setTimeout(() => {
  console.log("Je m'affiche après 1 seconde")
}, 1000)
```

`setInterval` et `clearInterval`

Tout comme les `timers` il possible de lancer une fonction toutes les XX millisecond :

```
// Je créé un "interval", une fonction qui se lance
// toutes les secondes
const interval = window.setInterval(() => {
  // Ici tic tac s'affiche toutes les secondes et ça
  // jusqu'a l'infinie !
  console.log('Tic Tac !')
}, 1000)

// Je créer une fonction qui attend 1h
window.setTimeout(() => {
  // J'arréte l'interval précédent
  window.clearInterval(interval)
}, 60 * 60 * 1000)
```

Les intervalles utilisent les resources de votre **processeur** (CPU). Ici, on demande on processeur un calcul récurant. Le **processeur** n'est pas conçu pour réaliser ces opérations. Il ne peut pas échoué une demande, on dit que le processeur est **error less**.

Il est donc dangereux de surcharger le processeur !

Il éxiste un composant, **la carte graphique** qui posséde lui aussi processeur c'est le **GPU**. Ces processeur sont souvent bien plus puissant ! Cependant, il est beaucoup moins précis et permet totalement les erreurs on dit qu'il est **error prone**.

`requestAnimationFrame`

C'est une fonction qui utilise le **GPU**, spécialement conçu pour les jeux ou toutes applications avec beaucoup d'naimation à la seconde. Génralement toutes applications avec du traitement graphique vont utiliser ce `requestAnimationFrame`

```
function gameLoop() {
  console.log('Tic Tac')

  // Je demande à mon GPU de relancer
  // ma fonction
  window.requestAnimationFrame(gameLoop)
}

// Je demande à mon gpu de lancer
// ma fonction
window.requestAnimationFrame(gameLoop)
```

Les événements

Il est possible d'ajouter des event sur la window, ses événements ne sont pas les même que le document ou bien l'élément.

`load`

Évenement déclancher lorsque la page (le document) à finie de charger (afficher le html, le css, inclue tout les scripts etc ...)

```
window.addEventListener('load', () => {
  console.log('Ma page est prête !')
})
```

`resize`

Évenement déclencher lorsque l'utilisateur redimensionne la fenêtre

```
window.addEventListener('resize', () => {
  console.log('Il y a redimensione de la fenêtre !!! ')
})
```

`scroll`, `copy`, `cut`, `paste` etc ...

Il éxiste de nombreux événement comme :

- Le `scroll` : Déclenché lorsque l'utilisateur actionne la barre de défilement
- le `copy` : Déclenché lorsque l'utilisateur fait un « copier » (CTRL-C)
- le `cut` : Déclenché lorsque l'utilisateur fait un « couper » (CTRL-X)
- le `past` : Déclenché lorsque l'utilisateur fait un « coller » (CTRL-V)

etc ...

Pour aller plus loin

La window possède aussi de nombreuse *technologie* (librairie) présent dans votre navigateur :

- L'historique : `window.history`
- La location : `window.location`
- Local Storage : `window.localStorage`

et bien plus ! ...