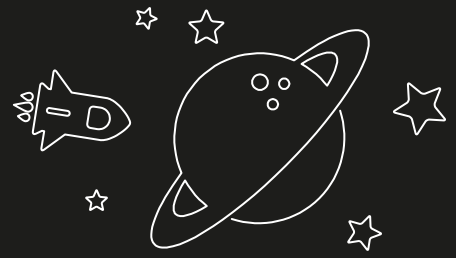

Aller plus loin
JS



LE PROTOTYPAGE

2

- Tout type est “wrappé” par un objet
- Tout objet contient un propriété ***prototype***
- ***prototype*** contient l’ensemble des méthodes et propriétés accessibles de l’objet hérité
- Héritage

var Object2 = Object.create(Object1);

Déclaration d'un objet

- Accolade (A)

```
var myObject = {
    p1: “foo”
};
```

- Constructeur (C)

```
function MyObject() {
    this.p1 = “foo”
}
var myObject = new MyObject()
```

Publique

Propriétés

- **(A)**

```
{
  p1: “foo”
};
```

- **(C)**

```
function MyObject() {
  this.p1 = “foo”;
}
```

Méthodes

- **(A)**

```
{
  m1: function() {}
};
```

- **(C)**

```
function MyObject() {
  this.m1 =function() {};
}
```

Prototype

```
myObject.prototype.p1 = “foo”;
myObject.prototype.m1 = function() {};
```

Privé

Propriétés

- **(A)**

Impossible

- **(C)**

```
function MyObject() {
  var pr1 = “foo”;
}
```

Méthodes

- **(A)**

Impossible

- **(C)**

```
function MyObject() {
  var mr1 =function() {};
}
```

Privilégié

Propriétés

- **(A)**

Impossible

- **(C)**

Impossible

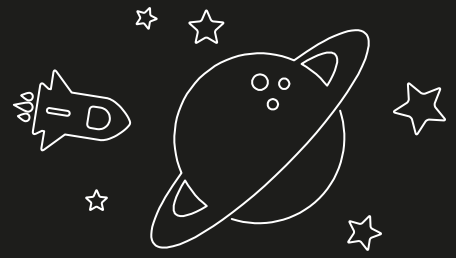
Méthodes

- **(A)**

Impossible

- **(C)**

```
function MyObject() {
  var pr1 = “foo”;
  this.ml1 =function() {
    console.log(pr1);
  };
}
```



Exercice 3 - Prototypons

8

./exercice-3/prototype.js

Reprendre l'exercice 1 - Manipulation des chaînes

- Rendre toutes les fonctions accessibles pour chaque String, excepté prop_access

  Exemples:

  ucfirst("ma chaine") => "ma chaine".ucfirst()

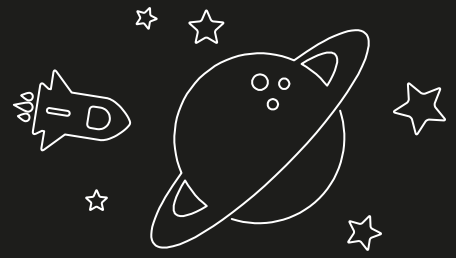
  vig("ma chaine", "ma clé") => "ma chaine".vig("ma clé")

- Rendre prop_access accessible pour chaque Object

  Exemple:

  prop_access(object, "animal.type.name") => object.prop_access("animal.type.name")

Commit : [DONE] exercice 3



Exercice 4
Here we go!

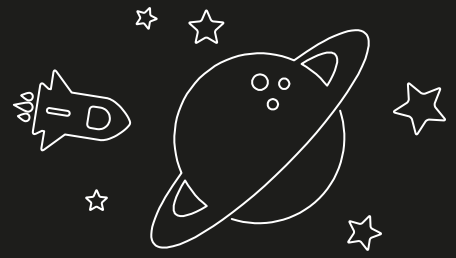
10

“

https://github.com/kmarques/esgi-cours/javascript/exercice-4.md

11

Commit : [DONE] exercice 4



Les Exceptions

12

- 6 types d'exceptions

EvalError : erreur dans un *eval()*

RangeError : utilisation d'un nombre en dehors des valeurs possibles

ReferenceError : utilisation d'une variable non déclarée ou hors scope

SyntaxError : erreur de syntaxe dans le code soumis à un *eval()*

TypeError : utilisation d'une fonction n'appartenant pas au type

URIError : utilisation de caractères illégaux dans une fonction URI

- Possibilité de créer ses propres exceptions

```
function MyError(param1, param2, ...) {
   var instance = new Error("custom message");
   Object.setPrototypeOf(
          instance, Object.getPrototypeOf(this)
   );
   If (Error.captureStackTrace) {
     Error.captureStackTrace(instance, MyError);
   }
   return instance;
}
```

Gestion des exceptions

try {

Permet d'exécuter des instructions à risque

throw *"exc" | new Error("exc");*

} catch (error) {

Catch les exceptions levées

If (error instanceOf MyError)

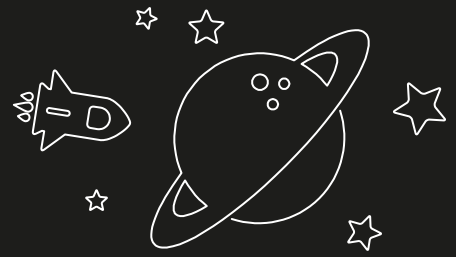
ou

If (error.name === "MyError")

} finally {

Permet d'exécuter des instructions même si des exceptions sont levées

}



Exercise 5
THE Exception

16

./exercice-5/exception.js

- Reprendre l'exercice 3, uniquement la méthode prop_access

- Créer une exception UndefinedPropertyError(path, property, object)

  Message: "Property '{property}' not exist for path '{path}', expected one of : [available object properties]"

  EX: UndefinedPropertyError('animal.gender', 'gender', {animal: {type: "dog", name: "spoky"}})

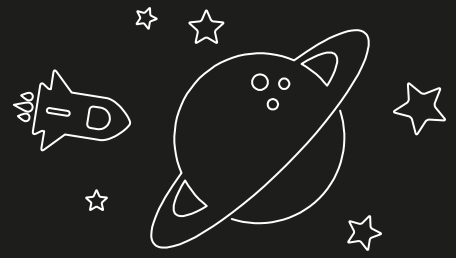
  => "Property 'gender' not exist for path 'animal.gender', expected one of : [type, name]"

- Créer une fonction **test** qui catch l'exception et affiche

  Si exception, **"Exception caught"**

  Sinon, **JSON.stringify(valeur retourné)**

Commit : [DONE] exercice 5



Les Promises

18

- Une *Promise* est un objet
- Permet d'exécuter des opérations asynchrones
- Complétion/Rejet sont toujours exécutés en fin de boucle événementielle
- Permet de rendre le code plus lisible et plus réactif


```
var promise = new Promise(
        function(resolve, reject) {
                // Do Something
                If (cond) {
                        resolve(result)
                } else {
                        reject(result)
                }
        }
);
```

resolve => représente la bonne complétion d'une promise

reject => représente le rejet d'une promise

Les notions de complétion et rejet sont définies par l'utilisateur

Le chaînage et la composition

Chaînage

.then(resolve, reject)

.catch(reject) <=> then(null, reject)

Composition

.all([prom1, prom2, prom3])

Attend la fin de toutes les complétions

S'arrête au premier rejet

.race([prom1, prom2, prom3])

S'arrête à la fin de la première complétion

S'arrête au premier rejet

Nouvelle écriture

async

Permet d'exécuter du code séquentielle avec des Promises tout en restant dans un contexte de parallélisation

Valeur de retour => resolve

Exception => reject

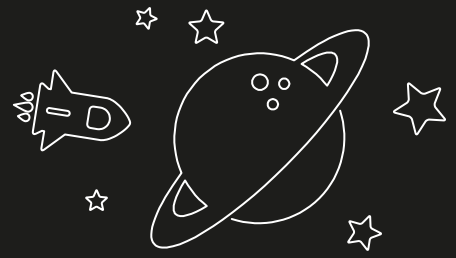
await

Attend la fin d'exécution d'une Promise

resolve => Valeur de retour

reject => Lève une exception

Attention!! *await* n'est disponible que dans une fonction *async*



Exercise 6
Je promets

./exercice-6/promise.js

Simulation d'appels serveurs

- Créer une Promise getStudents qui récupère une liste d'étudiants entre 1 et 2 secondes

  EX: [{ name: "Dupont", cours: [1, 3, 5] }, { name: "Lea", cours: [2, 4] }, { name: "Charles", cours: [1] }]

- Créer une Promise getCourses qui récupère une liste de cours entre 2 et 4 secondes

  EX: [{ id: 1, name: "JS" }, { id: 2, name: "PHP" }, { id: 3, name: "C#" }, { id: 4, name: "F#" }, { id: 5, name: "CSS" }]

- Créer une Promise qui mappe à l'ensemble des étudiants les cours associés entre 1 et 4 secondes

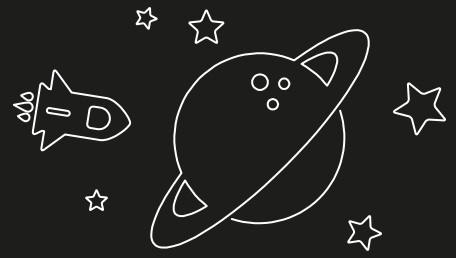
  EX: [{ name: "Lea", cours: [{ id: 2, name: "PHP" }, { id: 4, name: "F#" }] }, …]

- Créer une Promise qui contrôle le temps d'accès global
  - Celle-ci doit rejeter si le temps max dépasse 7 secondes
- Afficher la fonction et le temps estimé pour chaque Promise

  EX: "getStudents:2"

- **Afficher "Merge OK" si tout s'est bien passé sinon "Timout"**

Commit : [DONE] exercice 6



WEB APIS

25

DOM API

- Représentation sous forme d'AST d'un document HTML ou XML

- Deux types de noeuds
  - Element : représente un tag dans le code source
  - Attribute : représente un attribut d'un tag

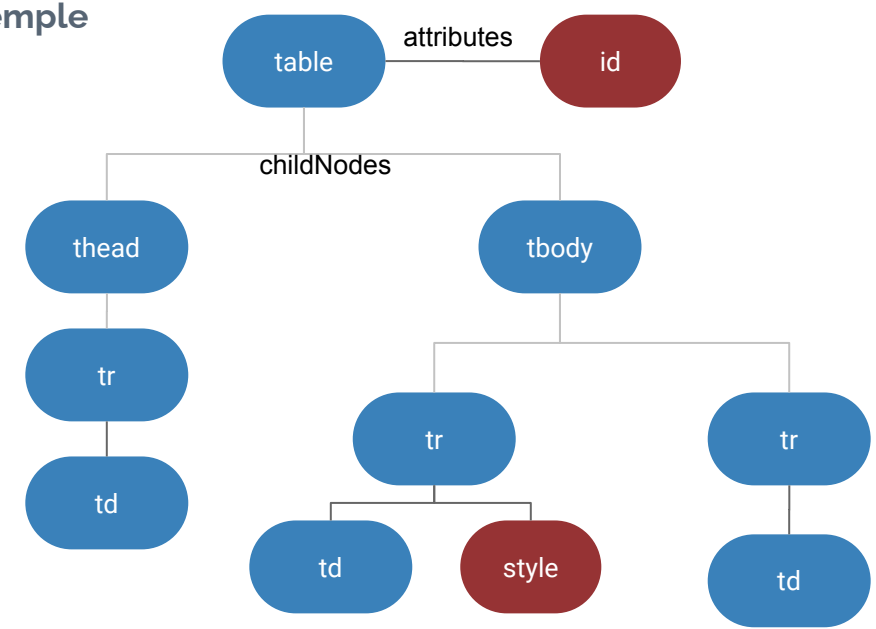
- Chaque noeud du DOM est un Object contenant ses propres méthodes/propriétés selon son type

- L'API DOM est standardisée par la W3C

  => garanti une même base fonctionnelle quelque soit le langage

Exemple

```
<table id="table1">
        <thead>
                <tr>
                        <td></td>
                </tr>
        </thead>
        <tbody>
                <tr style="border: 1px solid black">
                        <td></td>
                </tr>
                <tr>
                        <td></td>
                </tr>
        </tbody>
</table>
```



Manipulation du DOM

Les accesseurs

Eléments globaux

window : fenêtre/onglet du DOM

document : ensemble du DOM

Sélecteur DOM

document.getElementById(monId)

=> élément avec ID monId

{element}.getElementsByClassName(maClass)

=> sous-éléments avec Class maClasse

{element}.getElementsByTagName(monTag)

=> sous-éléments avec Tag monTag

Sélecteur CSS

{element}.querySelector()

=> 1er sous-élément correspondant

{element}.querySelectorAll()

=> tous les sous-éléments correspondants

Les modifieurs

Création

document.createElement(“tag name”)

=> crée un noeud de type “tag name”

document.createTextNode(“mon texte”)

=> crée un noeud de type texte ayant pour valeur “mon texte”

Insertion

{element}.appendChild(newElement)

=> ajoute un sous-élément à la fin

{element}.insertBefore(newElem, refElem)

=> ajoute un sous-élément avant un sous-élément existant

Suppression

{element}.removeChild(ElemToDelete)

=> supprime un sous-élement

Manipulation d'un DOMElement

{element}.parentNode
=> accède à l'élément parent

{element}.childNodes
=> accède aux sous-éléments

{element}.attributes
=> accède à tous les attributs

{element}.getAttribute("monAttr")
=> accède à l'attribut "monAttr"

{element}.className
=> accède à toutes les classes sous forme de String

{element}.classList
=> accède à toutes les classes sous forme de List

{element}.style
=> accède à l'object style de l'élément
=> facilite sa manipulation

Gestion des événements

{element}.onClick = func
=> exécute une fonction au click

** **
=> exécute une fonction au hover

{elTarget}.addEventListener("monEvent", func)
=> exécute func lors de l'événement "monEvent"

{elTarget}.removeEventListener("monEvent", func)
=> supprimer func lors de l'événement "monEvent"

{elTarget}.dispatchEvent(event)
=> diffuse l'événement event à l'élément

event.preventDefault()
=> annule l'effet d'un event
=> ex: annuler un click, une saisie clavier

event.stopPropagation()
=> annule la propagation d'un event vers l'élément target

Création

Var event = new Event("customName");
=> crée un event de type customName

Var event = new CustomEvent("customName", {detail: customData});
=> crée un event de type customName avec des données complémentaires

HISTORY API

- Permet la gestion de la navigation dans le navigateur

- 2 possibilités:
  - window.location
  - window.history

window.location

Les modificateurs

var urlBrowser = location.href
=> accède à l'url courante

location.assign("http://exemple.com")
location.href = "http://exemple.com"
=> charge la nouvelle page

location.reload()
=> recharge la page courante

location.hash = "42"
=> modifie l'ancre de l'URL
=> "http://exemple.com#42"

location.replace("http://exemple.com")
=> Remplace par la nouvelle page sans historique

Les accesseurs

location.pathname
=> accède au chemin de l'URL
=> "/mon-cours/javascript"

location.search
=> accède aux query params de l'URL (String)
=> "?page=1&admin=true"

...

window.history

Les modificateurs

history.back()
=> affiche la page précédente

history.forward()
=> affiche la page suivante

history.go(x)
=> retourne ou avance de x pages

history.pushState(stateObject, title, path)
=> modifie l'URL en cours
=> ajoute une nouvelle entrée à l'historique
=> définit un objet associé à l'entrée de l'historique

Les accesseurs

history.state
=> retourne l'objet associé à la page en cours

STORAGE API

- LocalStorage
  - Persistence: infini
  - Scope: hostname/protocole
  - Type: String
- SessionStorage
  - Persistence: session
  - Scope: hostname/protocole
  - Type: String
- Cookie
  - Persistence: date d'expiration
  - Scope: document
  - Type: String

Local/SessionStorage

storage.getItem(key)
=> retourne la valeur

storage.removeItem(key)
=> supprime la valeur associé à la clé

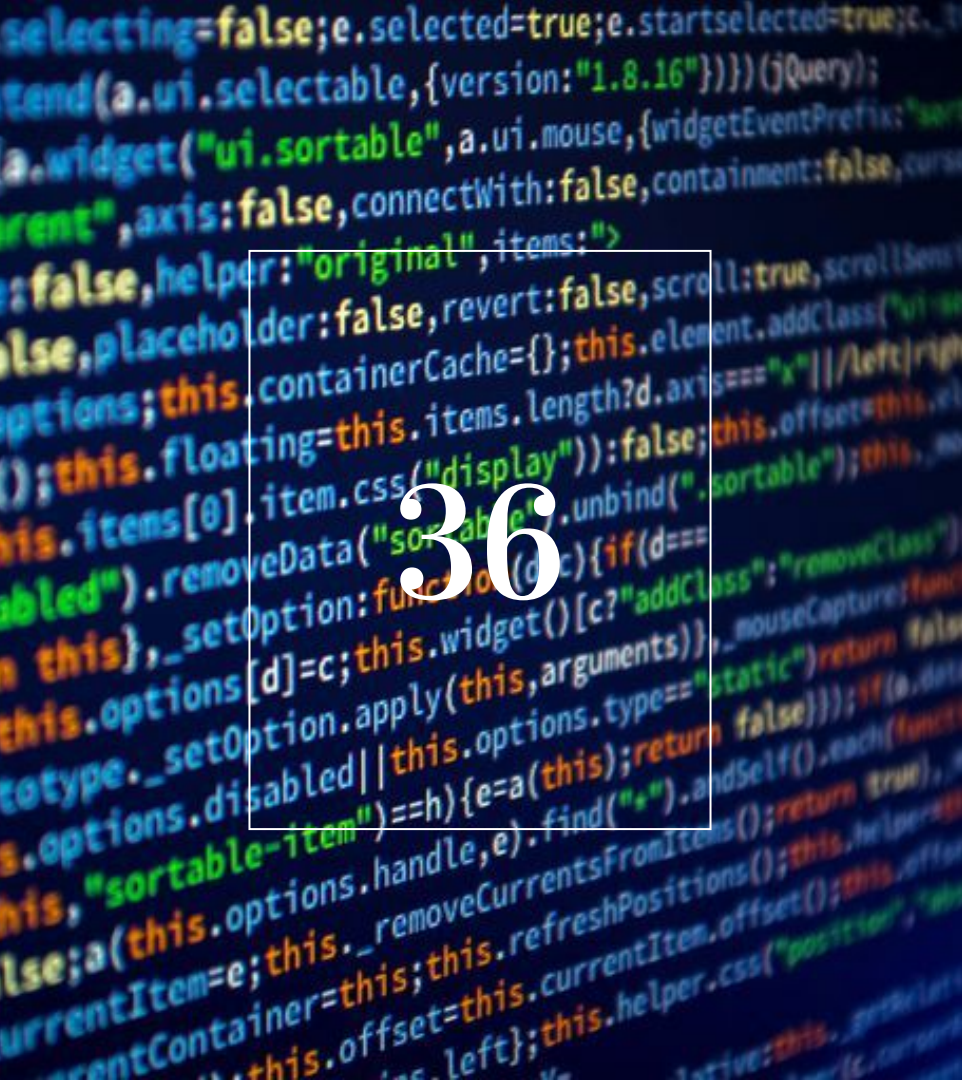
storage.setItem(key, value)
=> ajoute la value à la key

storage.clear()
=> vide toute la base

Cookies

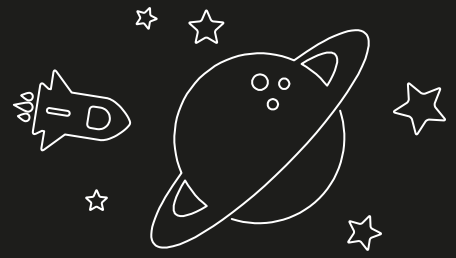
var myCookies = document.cookie
=> retourne tous les cookies de la page (String)

document.cookie="key=value"
=> ajoute un cookie value associé à la clé key



AUTRES API

- Navigator
  - Dispose d'API natives
    - navigator.geolocation
    - navigator.permissions
    - navigator.battery
    - navigator.bluetooth
    - ...
- Sensors API
  - Accelerometer, Orientation, Proximity, ...
- ServiceWorker
  - Exécute des tâches JS en fond
- FileReader
  - Permet de lire des fichiers locaux
- ...



LES MODULES

37

Les modules

- Permet de générer des libs JS
- Les variables sont scopées aux modules, hors variables globales
- Système d'import/export

Utilisation Web

- <script type="module" src="./main.js"></script>
- Utiliser un serveur Web

Utilisation Node

- Nommer les fichiers en .mjs

Exemple

Library.js

Méthode multi-export (ES6)

```
export const myVar = 10;
export function myFunc() {};
export default function myDefault() {};
```

Méthode single-export (CommonJS)

```
const myVar = 10;
function myFunc() {};

module.exports = {
        myVar: myVar,
        func1: myFunc
};
```

Main.js

Méthode multi-export (ES6)

```
import {myVar as myVar2, myFunc, default as func2} from
"./library.js";
 => importe toutes les fonctions nommées de la lib

import func2 from "./library.js";
 => importe la fonction par défaut de la lib

Import func2, {myVar} from "./library.js"

import * as lib from "./library.js";
 => importe tous les exports dans un objet lib
```

Méthode single-export (CommonJS)

```
const func1 = require('./library.js').func1;
const var1 = require('./library.js').myVar;
```