Quelques éléments de JavaScript

IUT Orléans – Département Info.

Licence Pro Web et Mobile

2022-2023

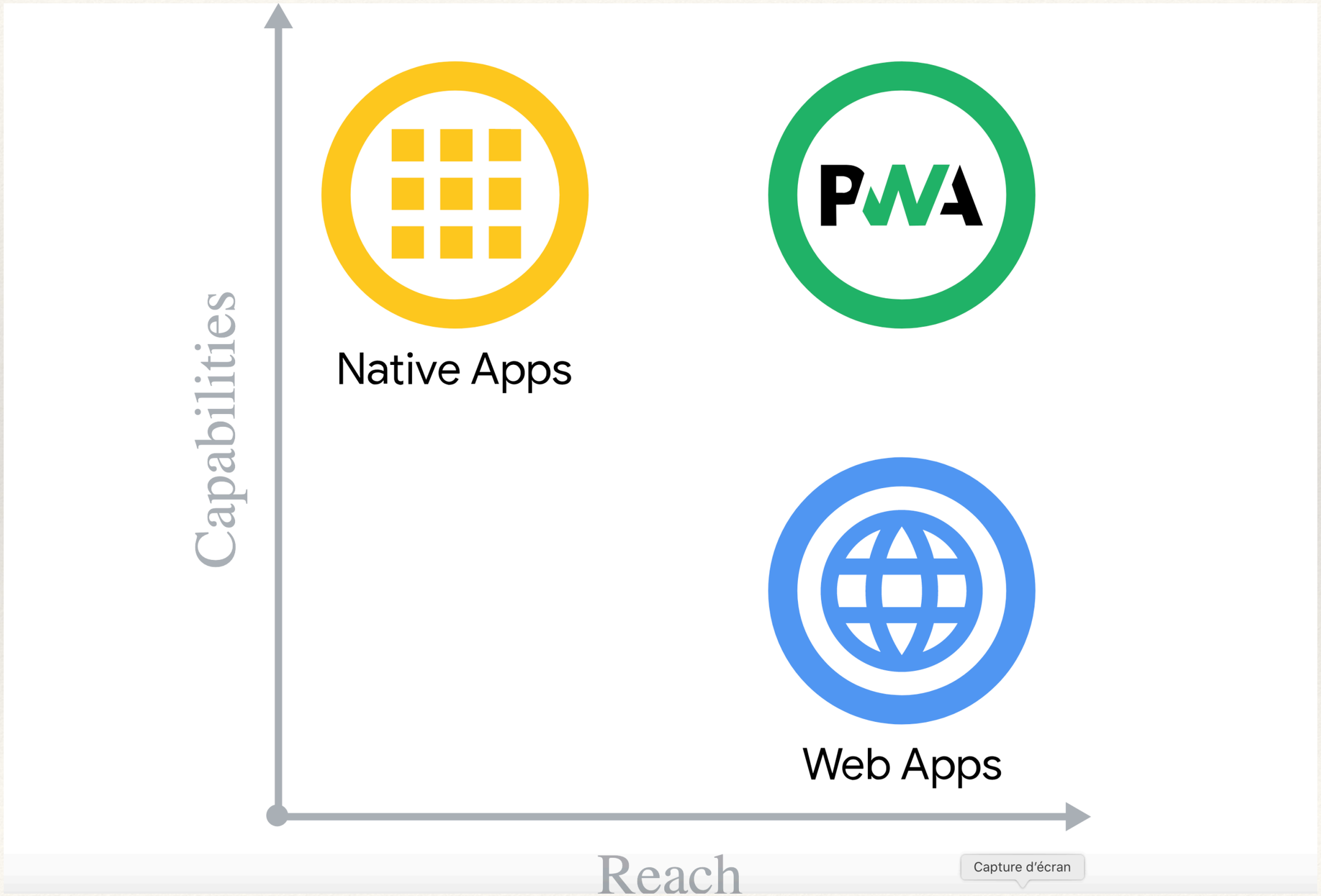
Web

Gérard Rozsavolgyi

Quelques Utilisations fréquentes de Javascript

- Calculs
- Aide ergonomique
- Animations
- Menus
- Saisie et vérification de données
- Galleries photo
- Connexions aux APIs
- SPA = Single Page Applications
- PWA = Progressive Web Apps

Progressive Web Apps



Introduction

- Extension du code HTML5 (cf:'onMouseOver()' ...)
- Aspects dynamiques ajoutés à HTML5 :
    - Vérifications
    - Menus
    - Animations
    - Ajax : Meilleure interactivité et fluidité
    - Dessins sur Canvas
- Langage Interprété par le navigateur
- Solution relativement universelle (à peu près tous les navigateurs)

Origines

- LiveScript développé par Netscape
- Repris par Sun fin 1995 sous le nom de JavaScript
- Normalisation européenne sous le nom de ECMA Script

Attention

- JavaScript
- ECMAScript
- → En gros, le même langage
- → **Ce n’est pas du Java**
- TypeScript => JavaScript, plus structuré ! (Angular)

Versions

- JavaScript 1.0 Netscape 2.0, IE 3.0
- JavaScript 1.1 Netscape 3.0, IE 4.0
- JavaScript 1.8 dans Firefox > 3.0
- JavaScript 2.0 spécifié depuis 2005, pas implémenté => ECMAScript 6 ou ES2015 ou ES7/ES8 … aujourd'hui
- Transpileurs: Traceur ou Babel

HTML5 et JavaScript

- Dessins sur Canvas
- GeoLocalisation
- Recherche d'éléments :

  document.getElementsById(« xx »)
- Sauvegarde hors-ligne
- Voir par exemple http://slides.html5rocks.com/ JavaScript vs Flash

JavaScript et Java

- Syntaxe style Java,C++ mais nombreuses différences
- Javascript accède aux objets du navigateur mais pas les applets Java
- Code interprété, intégré à la page HTML ou placé dans un fichier externe .js
- Dans une applet, code extérieur à la page.
- On peut faire appel à du code javascript à partir d'une applet java
- JavaScript manipule essentiellement des objets prédéfinis (possible d'en créer de nouveaux) mais avec une syntaxe spéciale

Fonctions en JavaScript

Function

Paramètres non typés

return non obligatoire

Objets en old JS

```
// Déclaration
function MaClass(Arg1, Arg2, ...) {
    this.Attribut1 = UneExpression;
    this.Attribut2 = Arg1;
    this.Attribut3 = Arg2;
    this.Methode1 = UneFonction;
    return UneAutreExpression;
}
```

// Utilisation

- MonInstance = **new** MaClass(Expression1, Expression2, ...)

- Accès aux attributs : MonInstance.AttributX
- Accès aux méthodes : MonInstance.Methode1();

- *UneFonction* associée à la méthode *Methode1* a accès aux propriétés de l'objet par le symbole *this.* :
function UneFunction () {**return (this.**Attribut1 + "-" + **this.**Attribut2); }

Objets en ES6

```
class Personne{
    constructor(id, prenom, nom){
        this._id = id;
        this._nom = nom;
        this._prenom = prenom;
    }
    toString(){
        return `${this._nom} ${this._prenom}`;
    }
    get nom() {
        return this._nom.toLowerCase();
    }

    set nom(newNom){
        if(newNom){
            this._nom = newNom;
        }
    }
};
let p = new Personne(3,'John', 'Doe');
console.log(p);
p.nom='Azerty';
console.log(p.nom);
```

HTML et JavaScript (I)

- <script>
  contenu du script...
  </script>
- ou
- <script src="myscript.js"></script>

HTML et JavaScript (II)

- <script SRC="http://monsite.com/mon.js"> </script>
- ou
- ```
  <script>
    /** Commentaires : // ou /* et */ */
     // Variables :
     let texte = "Mon chiffre préféré est"
     const variable = 7
     alert(texte + variable);
  </script>
  → Variables locales et globales
  ```
  ```

Exemple (Date). (I)

```
<script>

    function afficheDate(){
        let dt=new Date();
        let cal="" + dt.getDate() + "/" +
            (dt.getMonth()+1)+"/"+dt.getYear();
        let hrs = dt.getHours();
        let min = dt.getMinutes();
        let tm = ""+hrs+":"+min;
        return("nous sommes le "+cal+"a"+tm);
    }

</script>
```

Exemple Date (II)

On appelle la fonction dans la page HTML :

```
<body>
   <script>
        console.log(afficheDate());
   </script>
    ../..
```

Attention: Majuscules/Minuscules !

Exemple : Dernière modif. d'un document

```
<html>
    <head>
        <title>Page d'accueil</title>
    </head>
    <body>
        <h1>Le site de l'Association Mondiale JS </h1>
        <p> dernière modification de cette page :
        <script>
            lastmod = document.lastModified;
            if (lastmod.length == 0)
                console.log(« Inconnue »);
            else console.log(lastmod);
            </script>
        </p>
    </body>
</html>
```

(D'après la date de dernière modif connue du système)

Les Objets prédéfinis de JavaScript

- Objets usuels :
  → ***String*, *Number*, *Date*, *Math*, *Array***
- Objets liés à la fenêtre :
  → ***Window*, *document*, *parent*, *top frames[ ]*, *screen*…**
- Objets liés au navigateur :
  → ***Navigator*, *location*, *history***
- Objets liés au document HTML :
  → ***HTMLElement*** : ***Form*, *Checkbox*, *input*, *select*, *submit***, etc.

Les types en JS

```
s = new String("Coucou")
typeof(s)
"object"
i=10
typeof(i)
"number"
```

Etrangetés

```
> null == 0
< false
> null > 0
< false
> null >= 0
< true
```

```
> "\t" == 0
< true
> " " == 0
< true
> "          " == 0
< true
> "  \t   \n          " == 0
< true
>
```

null, undefined, etc.

Non-zero value

null

0

undefined

Propriétés système

- navigator.appCodeName
- navigator.appName
- navigator.appVersion
- navigator.language
- navigator.platform
- navigator.plugins[]
- navigator.userAgent
- screen.availHeight
- screen.availWidth
- Ex: proprerties.html

window

- Méthodes : open, prompt, alert, confirm, etc.
- Fenêtre qui en ouvre une autre:
- `let win;`

```
let strWindowFeatures =
"menubar=yes,location=yes,resizable=yes,scrollbars=yes,status=yes";
function openRequestedPopup() {
    win = window.open("http://
www.developper.mozilla.org/", "Mozilla MDN",
strWindowFeatures);
}
```

Debogage

- Sous Firefox ou Chrome : Ouvrez les Outils de développement Web.(Debugger JS)
- Sous IE : Outils, Options Internet, Avancé / Afficher une notification à chaque erreur de script…
- Utiliser alert(« x= »+x) ou console.log(« x= »+x)

ECMAScript et DOM

- **ECMAScript** = Spécifications communes de la syntaxe, grammaire, types de bases (String, Object, Number, etc.) des langages de script (JavaScript et JScript)
- Mais ECMAScript ne parle pas des facilités d'interface utilisateur (menus, scrollbars, évènements, ...) ou **de la manière de manipuler le contenu et la structure des documents** (texte, structure logique, feuilles de styles, ...).
  → **DOM** (Document Object Model)

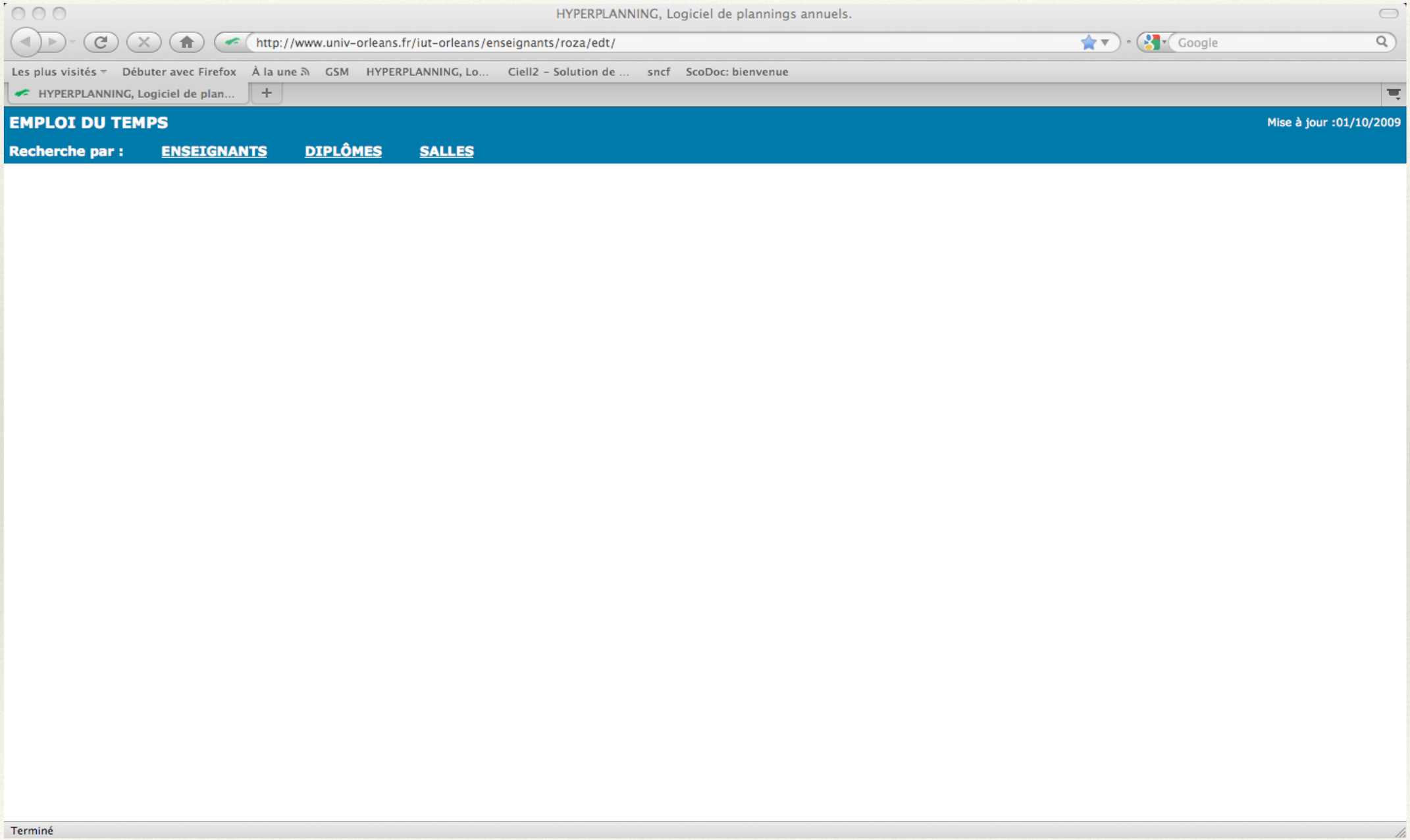
DOM

- DOM niveau 1 : Support XML 1.0 et HTML 4.0
- DOM niveau 2 : Namespaces, CSS, manipulations d’arbres par l’utilisateur
- DOM niveau 3: Compléments XML, XPath, etc.

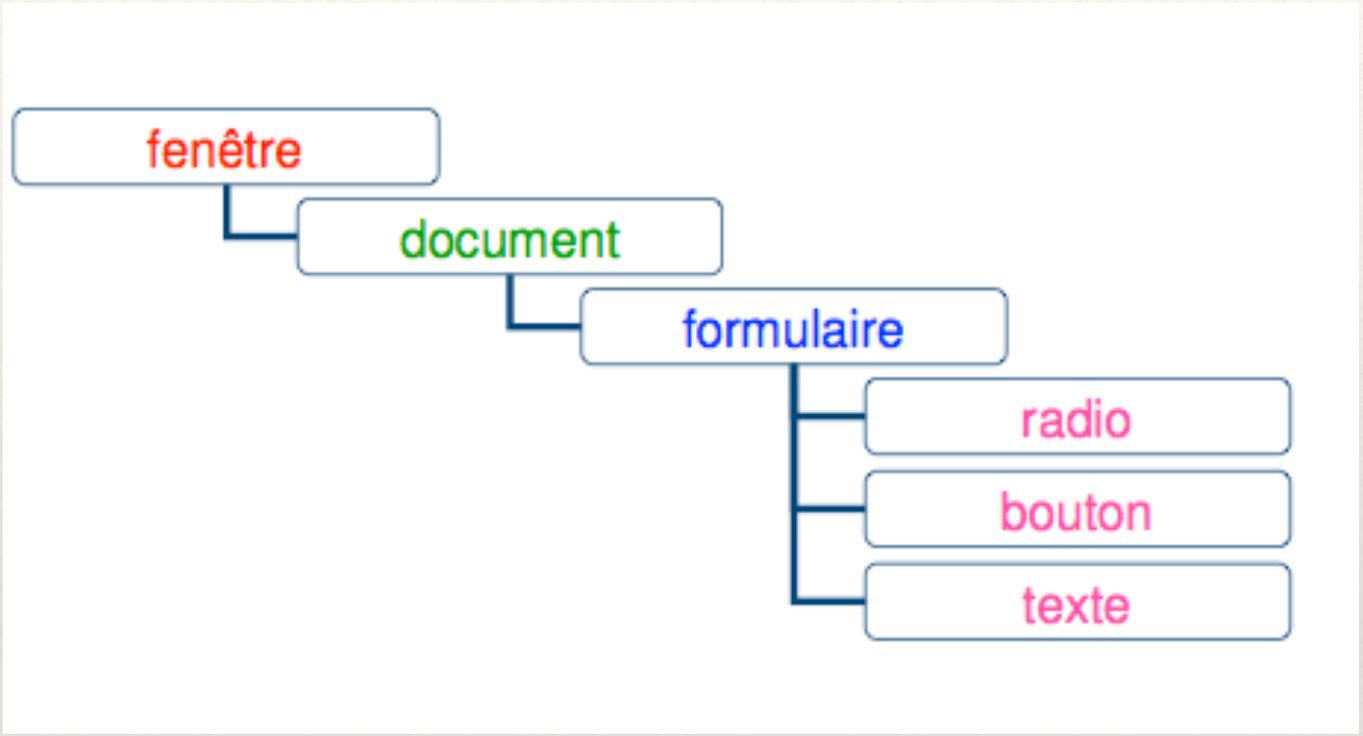
Principe de DOM: Document Object Model

- Divise une page HTML en objets
- Permet d'accéder à ces Objets, de les interroger, de les manipuler
- Ces objets se décrivent à travers une ***hiérarchie***

Objet de base : Fenetre



Document et autres entités HTML



Technologies complémentaires (XML)

- DOM for MathML 2.0: generic API générique pour l'accès aux documents MathML 2.0 (pour les matheux)
- DOM for SMIL Animation: API générique pour SMIL 1.0 (Synchronized Multimedia Integration Language) pour les animations
- DOM for SVG 1.0: API générique pour les documents SVG 1.0 (Scalable Vector Graphics) pour les graphiques 2D

Récupération d'Attributs


```
let image = document.getElementById(« monimage »)
let imgurl = image.src;
let image = document.images[0];
let width = parseInt(image.getAttribute(« WIDTH »));
image.setAttribute(« class », « mini »);
```

Création et insertion d'un Noeud DOM

```
<p id="lastmod"></p>
<script>
let lmod =
document.createTextNode(document.lastModified);
let dernmaj = document.createTextNode("Date de dernière
mise à jour: ");
document.getElementById("lastmod").appendChild(dernmaj);
</script>
```

Insertion d'un Noeud DOM

```
function insertAt(parent,child,n){
        if (n<0 || n > parent.childNodes.length)
            throw new Error(« index invalide »);
            else
                if (n==parent.childNodes.length)
                    parent.appendChild(child);
                else parent.insertBefore(child, parent.childNodes[n]);
}
```

Tableaux

```
const premiers = [2,3,5,7];
const n = premiers.length;
const jours = [« lundi », « mardi », « mercredi »];
```

```
function equalsTabs(a,b){
  if (a.length != b.length) return false;
  for (let i=0;i<a.length;i++)
      if (a[i] != b[i]) return false;
  return true;
}
```

Objets

```
const vide = {}
let point = {x:0, y:0};
let etudiant ={
        'nom': « Jean-Claude »,
        « numero ss »: « 14508294567 »,
        'naissance': « 27/09/1999 »,
        formation: {
                intitule: « Licence Professionnelle »,
                annee: « 2019-2020 »,
                groupe: « LP3A »
        }
}
  let secu = etudiant[« numero ss »]
```

Héritage

```
let obj = {}; / / obj récupère les méthodes de Object.prototype
obj.x = 1;  / / et est doté de son champ propre x
let p = inherit(obj);
p.y = 2;
let q = inherit(p);
q.z = 2 ;
let s = q.toString();
q.x + q.y  / / hérités
```

Utilisation d'expressions régulières

```
function validerForm() {
    const name=document.form1.nom.value;
    let reg = new RegExp("^[A-Za-z]{2,}$");
    if (!reg.test(name)) {
        alert("Veuillez entrer un nom avec des lettres !");
        return false;
    }
    const email = document.form1.email.value;
    reg = new RegExp("^[0-9a-z-_+.]+@[0-9a-z-_+]+[.][a-z]{2,4}$");
    if (!reg.test(email)) {
        alert("Veuillez entrer un email correct.");
        return false;
    }
    return true;
}
```

Vérifier une date avec des expressions régulières

```
function verifDate(input) {
  const isoDate = new RegExp("^([0-9]{2})/([0-9]{2})/([0-9]{4})$");
  let matches = isoDate.exec(input.value);
  if (!matches) {
    input.setCustomValidity(input.value + " pas date valide (jj/mm/aaaa) !");
    return;
  }
  const dateRefabriquee = new Date(matches[3], (matches[2] - 1), matches[1]);
  if ((dateRefabriquee.getMonth() == (matches[2] - 1)) &&
      (dateRefabriquee.getDate() == matches[1]) &&
      (dateRefabriquee.getFullYear() == matches[3])){
  / / la date saisie est correcte -- pas de message à afficher
      input.setCustomValidity('');
   }
  else {
    input.setCustomValidity(input.value + " date inexistante !");
  }
}
```

ECMAScript >=6

- let pour la portée des variables
- boucles for of
- Mot clef class
- Générateurs (yield)
- Paramètres variables dans les appels de fonction
- Templates String
- Objets littéraux
- Arrow function ("lambda")
- modules
- Promises

Traceur

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8" />
        <title>ECMAScript 6 </title>
    </head>
    <body>
        <script
    src="https://google.github.io/traceur-compiler/bin/traceur.js">
        </script>
        <script src="https://google.github.io/traceur-compiler/src/
bootstrap.js">
        </script>
        <script type="module">
            // Notre code ES6 ici
        </script>
    </body>
</html>
```

let

```
function swap(x, y) {
    if (x  !=  y) {
        var old = x;
        let tmp = x;
        x = y;
        y = tmp;
    }

    console.log(typeof(old));    / /  number
    console.log(typeof(tmp));    / /  undefined
}
```