Technologie Web

Contents

code cours : prototype.

Chapitre 0 : Introduction

Dans le cadre d'une communication entre un navigateur et un server, il est important de distinguer le permier du second. Pour communiquer ils utilisent le protocol http (*hypertext transfer*) et est organisé par le W3C (*World Wide Web Consortium*)

- **Navigateur** : Utilisé par l'internaute qui envoie des demandes, réceptionne des réponses et gère l'affichage.
- **Server** : Machine phyqique et logiciel capable de répondre aux demandes de fichiers

Html et DHTML

La différence entre elles deux est que le DHTML ajout au css et au html du javascript qui modifira le html et le rendra dynamique. d'où le D de DHTML

Le Javascript

Le javascript est un langage créé pour écrire des scripts (language interprété). Il est exécuté du côté client.

> tip : CTRL+SHIFT+J lance la console de débug dans firefox tip : CTRL+SHIFT+K lance la console de développement dans firefox

Le code Javascript s'exécute de façon synchrone. Le code est exécuté de façon asynchrone dans le cas d'un appel à une fonction suite à un événement.

Les commentaires

```
/*commentaire*/
//commentaire
```

Javascript désactivé On peut indiquer à l'utilisateur que javascript est désactivé par l'intermédiaire d'un message dans les balises **noscript**

console d'erreurs avec le raccourci Ctrl+Shift+J

console de développement avec le raccourci Ctrl+Shift+K

Insertion du code On définit les fonctions dans la head puis on les exécute dans le body

On peut demander une exécution suite à une manipulation de l'utilisateur.

via la propriété onclick="CODE JS" d'un bouton
ou via href="javascript:CODEJS" d'un lien "a"

on peut aussi avoir un script externe

```
<script src="nomFichier.js"></script>
```

Créer une table en html

```
<table>
    <tr>
        <th>titre coll 1</th>
        <th>titre coll 2</th>
        <th>titre coll 3</th>
    </tr>
    <tr>
        <td>1</td>
        <td>2</td>
        <td>3</td>
    </tr>
    <tr>
        <td>4</td>
        <td>5</td>
        <td>6</td>
    </tr>
    <tr>
        <td>7</td>
        <td>8</td>
        <td>9</td>
    </tr>
</table>
```

css pour changer le tableau

```
table:tr:nth-child(odd) // ligne impaires du tableau
table:tr:nth-child(3) // 3e ligne du tableau
```

chapitre 1 : Les variables

le nom d'une variable est sensible à la case.

typeof(var)

typeof(v) retourne le type de la variable placé en paramètre

- number
- undefined
- function
- string
- object

Types primitifs

Il existe 4 types primitifs en Javascript : *number*, *String*, *boolean*, *undefined*. Ces 4 types fonctionnent par référence

Let

```
let NOM_VARIABLE = 1234;
let maFonction = function(x,y){ return x+y };
let monNom = "Antoine";
```

Var

Les variables déclarées avec var sont valables dans le scope de la fonctoin et sont déclarées (mais pas initialisées) au début de la fonction.

En Pratique, on va éviter de les utiliser car ses particularités

const

constante locale au scope du bloc.

Strings

Un string peut s'écrire entouré de "" ou de ''.

Apostrophe inverse Quand on veut : un texte sur plusieurs lignes ou contenant des expressions à évaluer

```
let string = `je suis ${nom} et j'ai ${age}`
```

Opérations sur les Strings

- **Concaténation**: str + str
- **Longueur**: str.length
- **Extraction**: str[numLettre] ou str.charAt(numLettre)
- **Test**: str.startWith(str)
- **Test**: str.endsWith(str)
- **Test**: str.includes(str)
- **Extraction**: str.substring(début,longueur)
- **Extraction**: str.substring(debut,fin)
- **Recherche**: str.indexOf(str)
- **Recherche**: str.lastIndexOf(str)
- **Décomposition**: str.split(séparateur)

Date

```
let maintenant = new Date();
let minutes = maintenant.getMinutes();
let heures = maintenant.getHours();
```

Le hissage (= hoisting)

En javascript, les définitions des fonctions et des variables déclarées avec var sont hissées au début. Attention, car seul la déclaration est hissée et non l'initialisation

Nombres

- o.... : octal
- ox... : hexa
- 0o... : octal [ES6]
- 0b... : binaire

conversion

```
Number(x)
String(x)
Boolean(x)
parseFloat(x)
```

En fonction de x, il se pourrait que la conversion soit impossible.

- NaN : si ce n'est pas un nombre
- Infinity: 5/0
- -Infinity: -5/0

Chaptre 2 : Fonctions utiles

Affichage dans la console

```
console.log(message);
```

Pop-up

Message

```
alert(message);
```

Confirmation Retourne un booleen (true ou false)

```
let entreeUtilisateur = confirm(message);
```

Entrée de l'utilisateur (Attention) Retourne un string qu'il faut caster.

```
let entreeUtilisateur = prompt("message",VALEUR_DéFAUT);
```

DOM

On peut écrire directement du code HTML sous la forme d'un string que l'on ajoutera à la page. Cependant il n'est pas très efficient et ne sera pas utilisé dans la suite du cours

```
document.write ("code HTML");
```

Chapitre 3 : les Boucles

for / while / if /switch

?:

if

```
if(COND){}
else if{}
else{}
```

> Clean Code: Si l'on veut tester que la variable test est vraie on ne test pas (variable == true) mais (variable) Dans le cas de la négation, on ne test pas (cond != true) mais (!cond)

Boucle for-in and for-of

```
for(index in tab){...}// skip les indexes vides
for(element of tab){...}//objets
```

for of: si il y a un trou -> undifined

chapitre 4 : Les fonctions

En javascript les fonctions peuvent être placées en arguments d'autres fonctions. Si on redéfinit 2 fois la même fonction, seul la 2e sera conservée.

> attention aussi à l'hoistage et à l'utilisation de var

Fonction de base

```
function myFunction(var1,var2){
    return var1 + var2;
}
```

un code décrit sous la forme d'un objet fonction doit être évalué à chaque appel et est donc peu efficace (Clean Code : a éviter)

```
const maFonction = new function("var1","var2","return var1+var2;);
```

fonctions

```
let affiche = function osefNom (var,var){return var +var;}
```

Attention aux déclarations hoistées et donc non initialisées

Arguments

Surcharge et trop peu de paramètres Si une fonction est appelée avec plus d'arguments que nécéssaire, ils seront ignorés (surcharge)

à l'inverse, si il y en a trop peu, on les remplace par undefined.

Passage des arguments Les types primitifs effectuent un passage par valeur. càd une copie de la valeur de la variable. tandis que les objets sont passées par référence et donc les originaux sont envoyés en paramètre.

Return avec multiples arguments

```
sommeProduit(x,y){
    let somme = x+y;
    let produit = x*y;
    return {somme,produit}
};//retourne un objet {somme:VAL,produit:VAL}
```

Default argument

```
function maFonction(nom="antoine"){...}
```
```## chapitre 5: Les Tableaux

Les tableaux en JS sont Hétérogènes (différents types dans un même tableau) et dynamiques (t

La lecture hors borne d'un tableau donne undefined,

### Construction

#### Littéral
```` javascript
let tab = [1,2,3];
```

Constructeur

```
let tab = new Array(1,2,3);
```

Tableau vide

```
let tab = [];
let tab = new Array();
```
```

Tableau avec une taille x

```
let tab = new Array[x];
let tab = [,,,,,,];
```

Accéder à un élément

```
tab[indice]
```

taille d'un tableau

```
tab.length
tab["length"]
tab.length = 23 // change la taille du tableau
```

diminuer la taille du tableau engendre la perte des valeurs stockées

Tableau associatifs

```
tab["clé"] = valeur
```

Chapitre 6: Objets

En Javascript, les objets sont des tableaux associatifs (voir chapitre 5).

L'orienté objet en javascript ne fonctionne pas sous la forme de classe/instance mais sur un modèle basé sur des prototypes. On peut ajouter, supprimer, modifier les attributs et méthodes à la volée.

```
let personne = {};
personne["attr"] = 12 // equivalent a personne.attr = 12
```

On peut utiliser comme attribut n'importe quel mot (même les mots réservés), même plusueurs mots ou encore des nombres. attention cependant à ne pas mettre de point.

Pour vérifier si un objet possède une propriété on peut utiliser:

```
"age" in personne
```

Si une propriété n'existe pas, la lecture de cette propriété retournera undefined

Les sucres syntaxiques

Un sucre syntaxique est une simplification du code pour la rendre plus facile à écrire et à comprendre par un Humain

code Original :

```
let h = {
    "nom" : "bob",
    "age" : 42
```

```
    "toString" : function (){...}
};
```

peut s'écrire sous la forme :

```
let h = {
    nom:"bob",
    age:42,
    toString(){...}
};
```

Parcourir les propriétés d'un objet

```
for(let prop in h) {h[prop]}
```

Types de création d'objets

1. Création d'objets (à la volée)

```
function name(var){
    let h = x;
    let q = y;
    return {h,q};
}
```

2. Créer des objets hérités à la volée (prototypes gérés manuellement) (bad)

```
let proto = {}
proto.maFonc= function(){}// création de la fonction à ajouter
```

3. Créer fonction constructrice (prototypes automatiques) (mieux)

```
function Constructeur(v){
    this.h = h;
    this.n = n;
    };
    Constructeur.prototype.newFunction = function(){ ... };
    let obj = new Constructeur(...);
```

Chapitre 7 : Prototype

Le but d'un prototype est d'éviter de devoir intégrer une méthode à touts les objets. Le prototype fonctionne car il existe un lien entre un objet et son prototype et lorsque l'on ne trouves une propriété dans l'objet, on va chercher dans son prototype.

Créer

```
let monProto  {...}; // littéral
```

get et set

```
Object.getPrototypeOf(obj,Proto);
Object.setPrototypeOf(obj,Proto);
Object.is
assign(objParent,objChild);
```

Modifier

```
monProto.function= function(){...};
```

Heritage

```
let monObjet = objet.create(monProto);
```

Assigner des propriétés

```
object.assign (P3,{attr:val,attr:val});
```

Ecrase les propriétés existante en cas de conflit

Comment sont gérés les prototypes en interne

Javascript utilise une propriété : [[proto]] ou **proto** (appelé "dunder-Proto"). Qui est une référence vers un prototype. Si on utilise une propriété non existante de l'objet, JavaScript va chercher dans son prototype si elle n'existe pas. Donc, dans le cas d'une modification dans le prototype, tous les objets qui utilisent ce prototype sont impactés.

Constructeur

Un constructeur est une fonction qui possède le même nom que l'objet à créer avec une majuscule au début (ex: var Chien = new Chien("tom","wouf wouf");

Avantages L'avantage des constructeurs est l'attribution automatique du prototype aux objets crées. On ne doit plus créer le prototype à la main.

étapes lors d'un appel à new

1. Javascript crée un objet vide
2. On crée son prototype
3. On crée un constructeur
4. on exécute un this = nouvel objet.

Instance of

```
Tom instance of Personne // expression booléenne
```

Vrai si:

- l'objet à été crée par le constructeur Cons
- cons est le proto de tom
- si cons est le proto du proto de tom

Chapitre 8: Le DOM

DOM : "Document Object Model" : Protocole utilisé pour permettre à JS de communiquer avec le navigateur Les Objets d'une page sont ordonnés sous la forme d'un arbre.

L'objet Document

L'objet document est l'objet principal qui désigne la page.

- head: head of the html file
- body: contenu de la page
- images: tableau reprenant toutes les photos de la page
- links: tableau reprenant toutes les liens "a"
- title: titre de la page
- forms: tableau des forms de la page

innerHTML et textContent

- élément.innerHTML //retourne le code HTML sous la forme d'un String.
- élément.textConent // le texte dans la balise

  textContent en écriture va réécrire toute la balise

Arbre

- élément.children : liste des enfants d'un élément
- élément.childNodes : cite les balises enfant et les bouts de texte entre elles
- élément.parentNode : pointe ves le noeud parent
- élément.lastElemntChild :
- élément.firstElementChild :
- élément.firstChild :
- élément.lastChild :
- élément.parentNode :
- élément.previousSibling : //y compris les balises avec du texte
- élément.nextSibling ://y compris les balises avec du texte
- élément.previousElementSibling // sans le texte
- élément.nextSiElementSibling // sans le texte

cibler des éléments

- document.getElementById("identificateur"); // il ne peut y avoir qu'un seul élément avec un même ID
- document.getElementsByTagName("h1");

- document.getElementByClassName(“important”);
- document.querrySelector(“#menu”); // sélécteur css
- document.querrySelectorAll(“a.extérieur”);

  document.getElementById ne fonctionne que sur document

Changer le style d’un élément

- élément.style.color = “blue”;