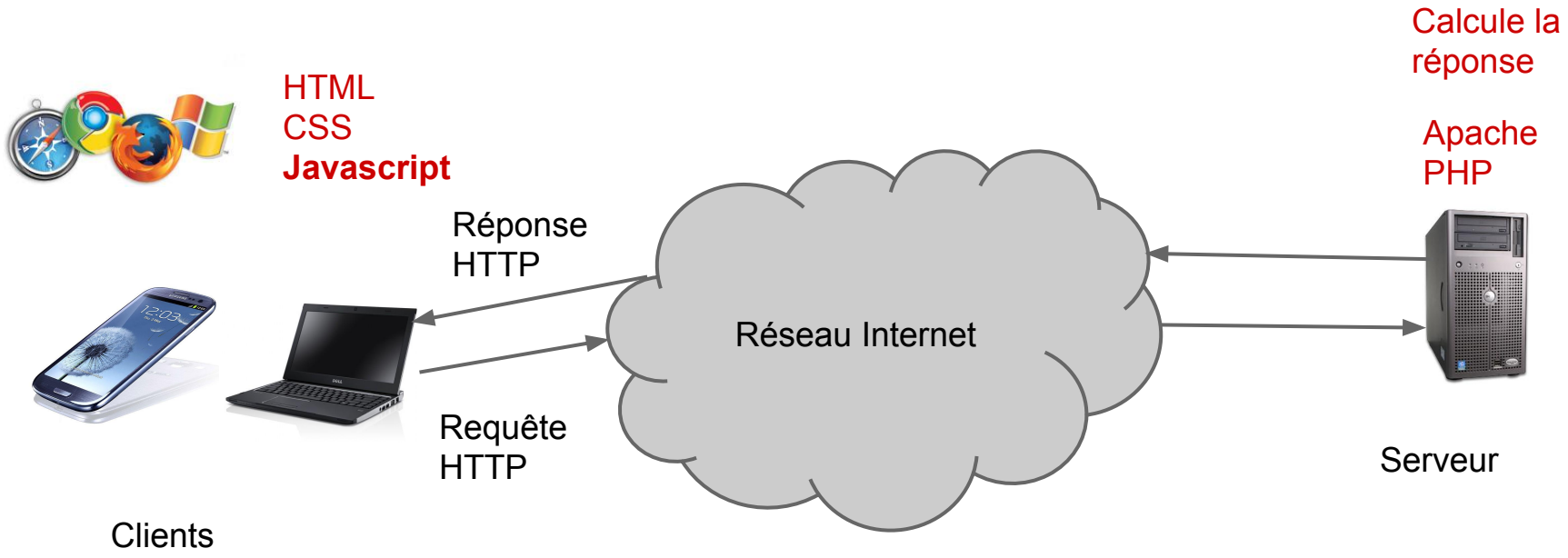
Technologies du web

6 - Javascript

Architecture web



Introduction

- HTML = structuration de la page web
- CSS = présentation de la page web
- Javascript (JS) = **interaction** dans la page web coté client (navigateur)

Client léger / client lourd

Léger : dans un navigateur, pas d'installation. Les traitements sont sur un serveur central.

Lourd : installé sur l'ordinateur généralement, application autonome, les traitements sont sur l'ordinateur.

Applications web

Les applications web permettent aujourd'hui des interactions utilisateur "riches" (réactivité, ergonomie) proche d'un client lourd et ceci grâce à **Javascript**

Une partie des traitements sont fait dans le navigateur, allégeant ainsi fortement la sollicitation des serveurs

Généralités

- Développé par Netscape en 1995 (livescript)
- Standardisé sous le nom de ECMAScript par le W3C
- **Rien à voir avec le langage JAVA**.

Versions

ECMA Script 5 : 2011

ES 6 / Ecma Script 2015 en cours d’adoption

Nombreux ajouts syntaxiques, meilleure prise en charge des classes et des objets, nouveaux types etc...

ES7 / Ecma Script 2016

Operateur **

Array.prototype.include

Généralités

Langage basé objet (et non orienté objet)

Syntaxe proche du C/Java/PHP

Notion de fonction anonyme et de ‘fermeture’

Pour quoi faire...

- Charger ou modifier une page **partiellement**
  - chargement au défilement (ex tumblr...)
  - autocompletion et recherche immédiate (ex google…)
  - Affichage d'un flux d'information (ex twitter)
  - ...
- Réagir à une action de l'utilisateur (bouton, lien, formulaire..)
  - Vérifier / valider les données avant envoi
  - Modifier la page en fonction de la saisie et/ou d'information recu
  - ...
- Faire des animations, jeux
  - défilement d'images,
  - doodles

Un langage de script

C’est un langage de programmation associé aux pages HTML et **exécuté par le navigateur**.

Le serveur envoie le code HTML, le code CSS **ET** le code Javascript **sous forme de texte**.

Le code est analysé et **exécuté ensuite par le navigateur**

Inclure le code dans le HTML

index.html

```
<html>
    <head>
        <script type="text/javascript" src="monfichier.js">
        </script>
    </head>
</html>
```

monfichier.js

```
console.log('un super message');
```

La console Javascript

chrome / chromium

Tools -> Javascript console

firefox

Tools -> Web developer -> Web console

Afficher un message

```
// affiche dans la console
console.log('mon message');

/* affiche une boîte de dialogue */
alert('un autre message');
```

Commentaires

```
// ceci est un commentaire sur une ligne
/* ceci est un commentaire
sur plusieurs ligne */
```

Suite d’instructions

```
// une instruction par ligne
// chaque instruction se finit par un ;

alert('bonjour');     // execute une fonction
var x;                // une variable sans valeur
var y = 2;            // une variable initialisée
x = y + 3;            // operation mathématique
console.log(x);       // execute une fonction
```

Variables

```
// ES5 : a une portée dans la fonction courante
var x = 1, y;

// ES6 : a une portée dans le {} courant
let y = 3;

// ES6 : variable non reassignable
const pi = 3.14;
```

Utilisation de la console

```
console.log(x);
console.log(x, y);
console.log(x + 1, y + 1);
```

Les types de données

- Les Nombres
- Les Chaînes de caractères
- Les Fonctions
- Les Objets
- Les Booléens

Les nombres

- Entier ou Flottant
  - var x = 2;
  - var pi = 3.1415;

- `typeof(x);` // renvoie ‘number’

Mathématiques usuelles

Nombre entier : `x = 2;`

Nombre flottant : `z = 3.1415;`

Opérations + - * / % ** : `u = z / x;`

Priorité : k = (x + z) / 3;

```
g = Math.sin(x);
a = Math.round(z);
  // voir http://www.w3schools.com/js/js_math.asp
```

Chaîne de caractères ou string

Un morceau de texte littéral

Entre simples 'quotes' (double quote toléré)

Utiliser + pour la concaténation

`typeof(x)` renvoie ‘string’

```
alert('bonjour');
alert('');                          // vide
alert('salut' + ' ca va ?'); // concatenation
```

Opérations sur les chaînes

```
var un_texte = 'jean';
var un_autre_texte = 'bonjour ' + un_texte; //concatenation
Var un_autre_texte_2 = `salut ${}`// templating ES6

un_texte = un_autre_texte.toUpperCase();
un_texte = un_autre_texte.toLowerCase();
un_texte = un_autre_texte.slice(2); // suppr 2 premiers char
console.log(un_text.length);
```

http://www.w3schools.com/jsref/jsref_obj_string.asp

Créer des fonctions

```
function mon_operation(a, b) {
   var c = (a + b) / 2;
   return c; // valeur de retour !!
};

var res = mon_operation(3, 7);
console.log(res);     // ??
```

Fonction et variable

Attention, une fonction peut être affectée à une variable **(différent de l'appel de fonction):**

```
var resultat = mon_operation(1, 2); // appel de la fonction

var ma_func = mon_operation; // !! référence à la fonction
ma_func('toto'); // appel de mon_operation
typeof(ma_func); // renvoie 'function'
```

Portée des variables

Une variable déclarée (avec **var**) **dans** une fonction n'est utilisable que dans celle-ci. Elle est dite **locale**.

Une variable déclarée **avant une fonction** est accessible et modifable depuis celle-ci

Une variable est dite **globale** lorsque est définie au début du programme. Elle est utilisable dans toutes les sous fonctions.

Portée des variables exemple

```
var a = 2 ; // variable globale
function f(x) {
    var b = 3;    // variable locale à la fonction
    a = x + b;   // a n'est pas définit dans la fonction
}
console.log(a, b);   // a = 2 et b est undefined !!
f(2);
console.log(a);  // a = 5
```

Fonction dans des fonction

Par défaut la portés des variables est des fonctions est limité à la fonction parente.

```
function f() {
     // corps de la fonction f
     function g() {
          //corps de la fonction g()
     }
     g(); // appel de g -> ok
}
f(); // ok
g(); // erreur g n’est pas accessible
```

Boucle

```
for (var i = 0; i < 5; i++) {
   console.log(i);
}
```

Condition

```
function abs(x) {
    if (x < 0) {
        x = -x;
        return x;
    } else {
        return x;
    }
}

console.log(abs(2), abs(-3));
```

Les valeurs particulières

- **undefined** : quand une variable n’est pas initialisé
- **null** : initialisé mais **sans valeur**
- **NaN** : Not a number

Les booléens

- true or false
- Toute variable dont la valeur **est différente** de 0, null, undefined, '' (chaîne vide) est assimilé à vrai.
- Operateur **!!** -> conversion vers un booleen

```
var x = 0, y = 'toto', f = false, v = true;
console.log(!!x, !!y);
typeof(f); //renvoie 'boolean'
```

Logique booléene

```
a == b      // egalité de valeur 2 variables
a === b     // egalité de valeur et de type
!= et !==    // différence avec/sans transtypage
< et >       // comparaison
<= et >=
!            // Operator Non : test si la valeur est equivalente
             // à false (ie 0, chaine vide, undefined, null)
!!           // Operateur NonNon : test si la valeur est true
a && b      // ET LOGIQUE : Vrai si a et b sont vrais
a || b      // OU LOGIQUE : Vrai si a ou b est vrai
```

Structure de controle

- Execution conditionnelle : If
- Execution en boucle : for

Structure de données

- Objets particuliers (!= PHP)

Liste [] : ensemble ordonné d'élément (de tout type)

Table associative {} : ensemble clé / valeur

Les clés sont des strings

Les valeurs sont de tout type

Listes

```
var ma_table = [4, 6, 9,'toto'];

console.log( ma_table );
console.log( ma_table.length ); // taille
console.log( ma_table[1] ); // 2nd element !!!
```

Listes

```
var table2 = [4,6,9,11];

console.log(table2);
for (var i = 0; i < table2.length; i++) {
   var val = table2[i];
   table2[i] = val + 1;
}
console.log(table2);
```

Opérations sur les listes et chaînes

```
var t = 'un super message';
var sp = t.split(' ');  // séparation selon un caractère

sp.push('et'); // ajout d’un element à la fin
sp = sp.concat(['un', 'autre', '!']); // concatenation
sp = sp.slice(1); // supprime le premier element

var j = sp.join('_');
console.log(j);
```

Fonction en paramètre

Il est possible de passer une fonction en paramètre d'une autre :

```
function my_function(l, f) {
    for (var i = 0; i < l.length; i++) {
        f(l[i]);  // execute f avec un élément en paramètre
    }
}
my_function([1,2,3], function (p) { console.log(p); });
```

Operation sur les listes

```
var ma_liste = [1,2,4,5];

// multiplie chaque element par 2
ma_liste2 = ma_liste.map(function (e) { return e * 2; }

// affiche les elements de la liste
ma_liste.forEach(function (e) { console.log(e); }

// les elements superieures à 2
ma_liste3 = ma_liste.filter(function (e) { return e > 2; }

// ES6
for (let i of [1,3,5]) { console.log(i); }
```

Tableau associatif / Objet Javascript

```
var ages = { jean: 24,
             julien: 32 };

console.log( ages.jean );
console.log( ages['jean'] );
x = 'jean';
console.log(ages[x]);   // ??
```

Parcours de tableau associatif

```
var ages = { jean: 24,
             julien: 32 };

for (var prenom in ages) {
   console.log(prenom, ages[prenom]);
}
```

Tableau associatif / Objet Javascript

Peut contenir des **sous objets** (liste ou objet)

```
var person1 = {
   nom: 'john',
   age: 12,
   telephones: ['0101010101', '0606060606'],
   adresse: { rue: 'place de la mairie',
              ville: 'grenoble'
            }
};
console.log(person1.adresse.rue)
person1.adresse.telephones.push('0909060606');
```

Tableau associatif / Objet Javascript

peut aussi contenir des **fonctions**

```
var tempManager = {
   temperatures: [13, 14, 12],
   mean: function () {
     var sum = this.notes.reduce(function (a, b) { return a + b; });
     return sum / this.note.length; // !! zero divide possible
   }
};
tempManager.temperatures.push(8); // ajoute une nouvelle note
console.log(tempManager.mean()); // affiche la moyenne
```

JSON Javascript object Notation

```
{ "nom": "bob",
  "age": 12,
  "tels": ["0601010101", "04010101010"],
  "adresse": { "CP" : 34200,
               "ville": "Sete"}
}
```

```
JSON.stringify(obj);  // conversion objet -> string
JSON.parse(a_json_string); // conversion string -> objet
```

Conversion de type

String -> Number

```
var i = parseInt('12');
var f = parseFloat('13.14');
var n = parseInt('toto'); // renvoie NaN
```

Object -> String

```
var d = new Date('12/12/2012');
d.toString();
```

Utilisation d’objets

- Un objet est une **structure de données avec des fonctions**
- Construction d’un nouvel objet avec l’operateur *new*
- Utilisation de l’**operateur point** pour appeler les fonctions

```
var d = new Date('12/11/2014');  // création d’un objet Date
var m = d.getMonth();    // appel d’une fonction sur d
typeof(d); // renvoie ‘object’

var alist = new Array(); // equivalent à var alist= [];
var anobj = new Object(); // equivalent à var anobj= {};
```

Oui mais sinon….

- On a vu les rudiments du langage.
- À quoi ca sert ???

- À manipuler la page HTML / CSS en fonction du contexte
  - Analyser et réagir à un formulaire
  - Faire des animations
  - ...

Variable globale de la page

- window : représente la fenêtre

```
console.log(window.location);
```

- document : représente le DOM

DOM Document object model

- Les balises HTML
- sous forme hierarchique
- sélectionnable par un sélecteur CSS

Accessible en javascript

Exécuter du code après chargement

Le code Javascript pour manipuler le DOM doit être exécuté seulement quand celui est chargé

```
document.addEventListener('DOMContentLoaded', function () {

    console.log('Aloha');

});
```

Exemple

```
<html>
  <head>
          <script type="text/javascript" src="monfichier.js"> </script>
  </head>
  <body>
      <div id="menu">
           <a href="#" id="menu1">Un lien</a> | <a href="#" id="menu2">Un autre lien</a>
      </div>
      <h1>Un titrel</h1>
      <p> Un texte de paragraphe <b>avec du gras</b></p>
      <ul id="laliste">
        <li class="ma-classe">elem 1</li>
        <li class="ma-classe">elem 2</li>
      </ul>
  </boby>
</html>
```

Selectionner un/des elements

```
// selectionne par id
var menu = document.getElementById('menu');

// selectionne le premier element du selecteur
var elem = document.querySelector(".ma-classe");

// selectionne tous les éléments du selecteur
var elems = document.querySelectorAll(".ma-classe");

for (var i = 0; i < elems.length; i++) {
    console.log(elems);

}
```

Evenements Javascript

Il est possible d'exécuter une fonction sur un événement survenant sur un élément du DOM.

- click
- hover or mouseover
- focus
- change
- submit
- ...

Evenements

```
var btn = document.querySelectorAll("a");

btn[0].addEventListener("click", function () {
      // do something here
      alert("menu 1 has been clicked");
});
```

Manipulation des classes et attributs

```
var div = document.querySelector("#menu");

// access aux classes
console.log(div.id);
var classes = div.classList;
console.log(classes);
classes.add("red");
classes.remove("blue");
classes.toggle("hidden");



// acces aux attributs
div.setAttribute("title", "main menu");
console.log(div.getAttribute("title"));
```

Contenu des balises

```
var myText = document.querySelector("p");

// Get HTML
var myHtml = myText.innerHTML;
console.log("innerHTML:  " + myHtml);
// set HTML
myText.innerHTML = "nouveau texte en <i>italique</i>";

// get text content (without tags)
var myContent = myText.textContent;

console.log("textContent:  " + myContent);
```

Naviguer dans le DOM

```
var l = document.querySelector('ul#laliste'),

var parent = l.parentNode;
var children = l.children;
```

Ajout/Suppression d’élement

```
// create node
var newLink = document.createElement('a');

// set attributes
newLink.id    = 'menu3';
newLink.href  = 'http://www.google.com';
newLink.title = 'recherche sur google';
newLink.innerHTML = 'google'; // link text

// insert it in the dom
document.getElementById('menu').appendChild(newLink);


// suppression
var link = document.querySelector('a#menu1');
link.parentNode.removeChild(link);
```

Animation

- Definition d’animation en CSS
  - .animation { text-decoration : blink; }
- Déclenchement par ajout de classe

```
document.getElementById('menu').classList.add(animation);
setTimeout(function () {
        document.getElementById('menu').classList.remove(animation);
 }, 1000);
```

Exécuter du code

En réponse à un évènement

- clique sur un lien - **click**
- clique sur un bouton - **click**
- envoie de formulaire - **submit**
- modification d’un champ input - **change**
- mouvement de souris - **mousemove**
- action clavier - **keypress**

Exemple de formulaire

```
<html>
     <head>
          <script type="text/javascript" src="monfichier.js"> </script>
      </head>
     <body>
       <form>
           <label for="nom">Nom</label><input id="nom" name="nom" type="text"/><br/>
           <input type="button" id="reset" value="reset"/>
           <input type="button" id="check" value="check"/>
       </form>
       <div id="info"></div>
   </body>
</html>
```

Exemple de formulaire

```
document.addEventListener('DOMContentLoaded', function () {

    var reset = document.getElementById('reset');
    reset.addEventListener("click", function () {
        document.getElementById('nom').value = '';
        document.getElementById('info').innerHTML = '<i>formulaire effacé</i>';
    });

    var check = document.getElementById('check');
    check.addEventListener("click", function () {
        var name = document.getElementById('nom').value;
        document.getElementById('info').innerHTML = '<b>Nom vide</b>';
    });
});
```

En conclusion

- Javascript est un langage exécuté dans le navigateur
- Il permet de modifier le code HTML dynamiquement en réponse à des événements
- http://www.w3schools.com/js/default.asp