
JS

AGENDA

1. Introduction
2. Javascript: Le coeur du langage.
3. Les boucles et les conditions.
4. Les fonctions en Javascript.
5. Les structures de données: Les objets et les tableaux.
6. Le DOM (Document Object Model).
7. La gestion d'événements.
8. Les formulaires.
9. Les cas pratiques.

1 Introduction

4

Introduction

- Un langage de programmation est un langage formel utilisé lors de la conception, ou l'exploitation d'un SI.
- Un langage est composé d'une syntaxe et répond à des besoins spécifiques:
  - Java: Utilisé pour créer des applications web ou Desktop.
  - R: Utilisé pour faire des statistiques
  - HTML: Utilisé pour créer des pages Web.
  - Latex: Utilisé pour créer des documents PDF.

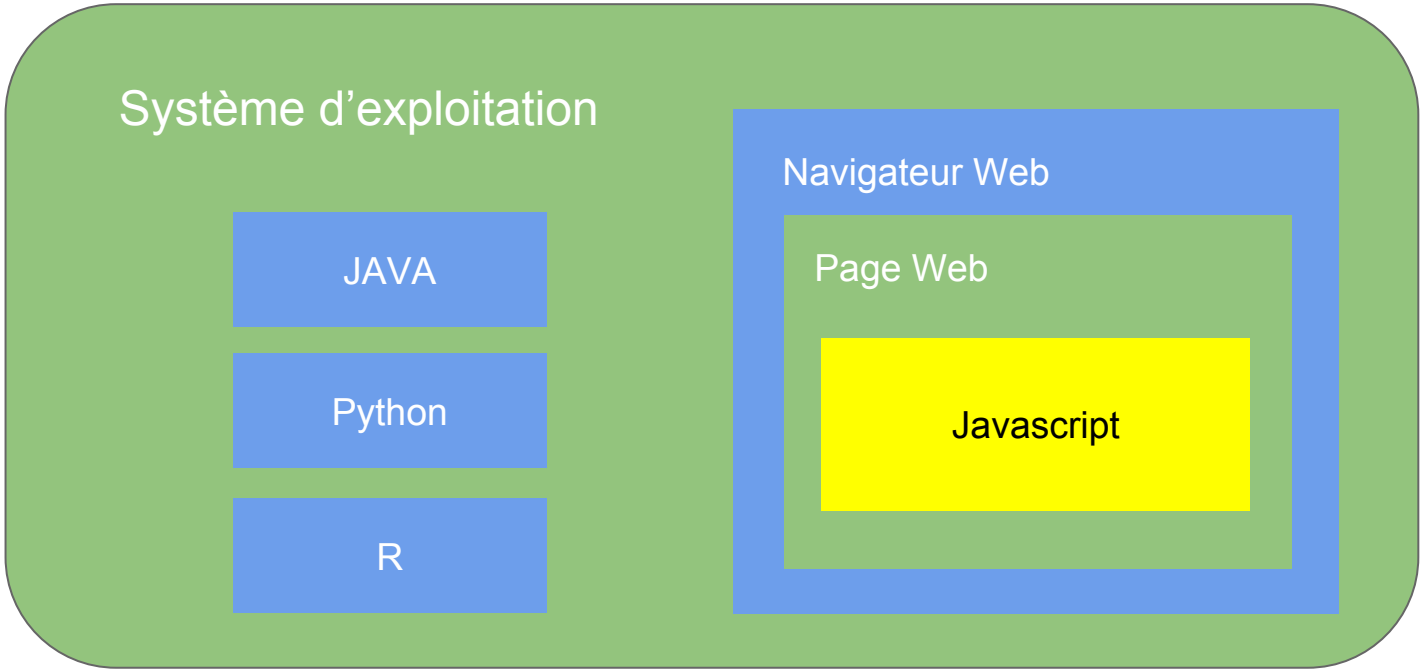
5

Introduction

- Javascript est un langage de programmation comme C++, Java, ou .net, etc...
- Javascript présente par rapport au autres langages de programmation des caractéristiques différentes:
  - Javascript est un langage qui s'exécute dans un navigateur Web au sein d'une page Web.
  - On a un accès limité au système d'exploitation, et au hardware (USB, etc...)

6

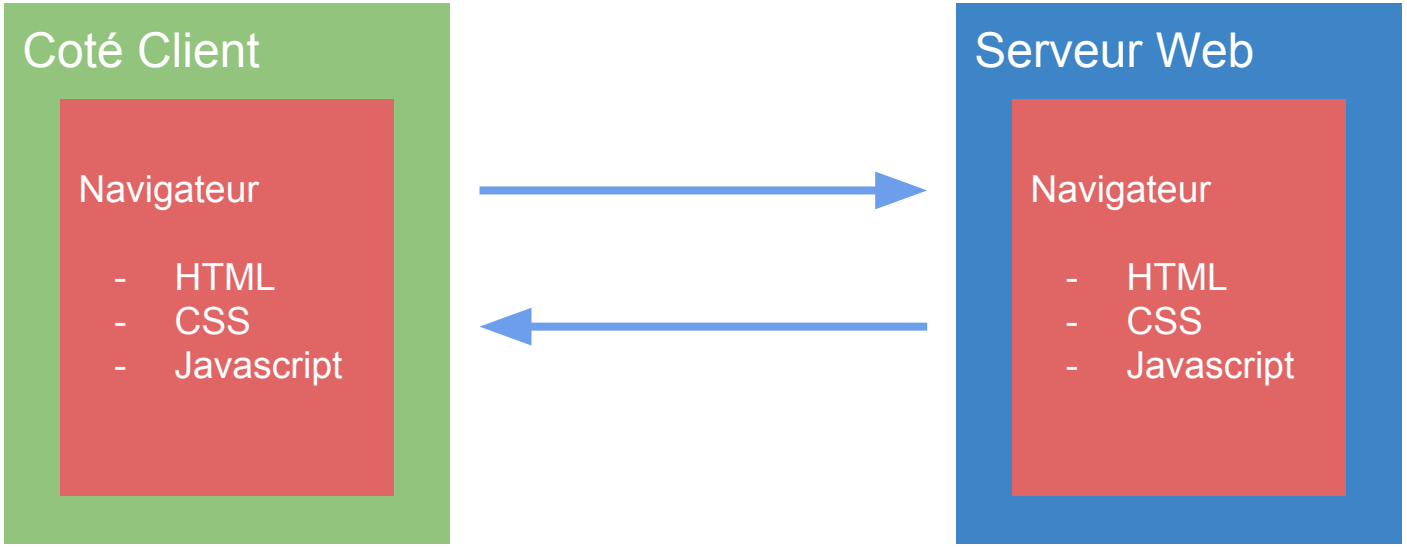
Introduction



7

Javascript

- Javascript est un langage exécuté côté client.

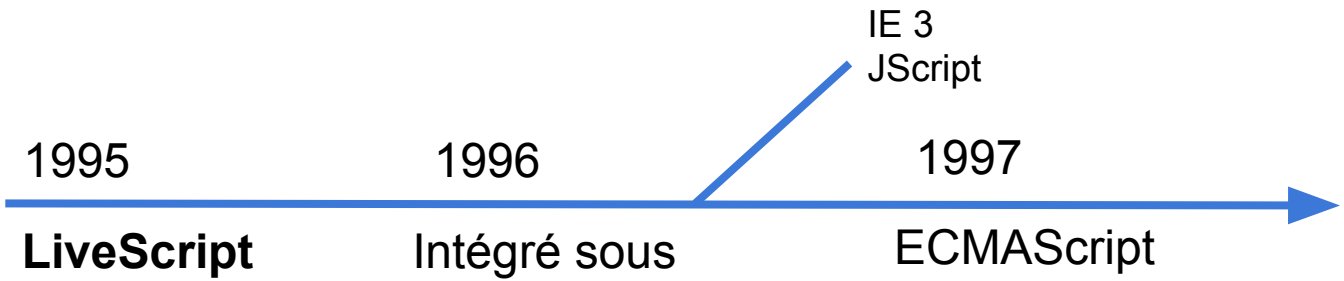


- Javascript Désactivé.
- Différentes versions de navigateurs.

8

Historique

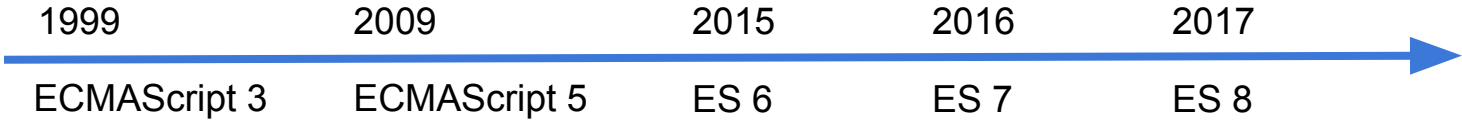
Bref historique de Javascript:



JavaScript ≠ Java

9

Historique



- IE
- FireFox
- Chrome
- Safari
- Opera

- **ECMAScript** est un standard, et c'est aussi un langage de programmation, il définit des règles, détails, et guidelines.
- **Javascript** est une implémentation du standard ECMA (ECMA-262)!

10

Compatibilité

- **Javascript** est compatible avec tous les systèmes d'exploitation MAC, PC, Linux, Unix…
- **Javascript** est compatible avec tous les langages de programmation ASP.Net, PHP, HTML, JSP…
- **Javascript** est compatible avec tous les éditeurs VisualStudio, NetBeans, Eclipse…

11

Prérequis

HTML

CSS

12

Outils

- Aptana
- SublimeText
- Notepad++
- Brackets

13

Où placer le JS ?

- Le code Javascript peut se placer soit:
  - Directement dans les balises HTML, à l'aide d'un gestionnaire d'événements.
  - Entre des balises propres au JS.
  - Dans un fichier externe, portant l'extension .js, qui sera inclus dans la page web.

14

Où placer le JS ?

Balise HTML:

```
<a href="#" onclick="alert('Salut')">Cliquez</a>
```

- Un gestionnaire d'événements est un attribut qui se place à l'intérieur d'une balise HTML, il commence par **on** suivi de l'événement en question (**click**, **dblclick**, **mouseover**, **mouseout,** etc...).
- L'action est lancé dès que l'événement spécifié se produit.

15

Où placer le JS ?

Balise propre au JS:

```
<script type="text/javascript">
<!--

//-->
</script>
```

- Ces balises peuvent se placer
  - Soit dans l'entête (**head**) de la page HTML.
  - Soit dans le corps (**<body> ... </body>**) de la page.

16

Où placer le JS ?

Fichier externe .js

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

- Cette ligne de code peut se placer:
  - Soit dans l'entête (**head**) de la page HTML.
  - Soit dans le corps (**<body> ... </body>**) de la page.

17

2

JavaScript: Le Coeur du Langage

18

La structure JS

- Javascript est un langage interprété et non compilé.
- Javascript est sensible à la casse, contrairement à d'autres langages comme le HTML.
- Javascript permet d'écrire plusieurs instructions sur une même ligne.
- Le ";" permet de marquer la fin d'une ligne, mais il n'est pas obligatoire.

19

La structure JS

- Les espaces n’ont pas d’effet sur l’exécution du code Javascript.
- Une chaîne de caractères (**string**) peut être entourée de “ ” ou de ‘ ‘ en Javascript.

20

Les commentaires

- JavaScript permet comme tout autre langage de rajouter des commentaires qui ne seront pas interprétés lors de l'exécution du code.
- // permet de rajouter une ligne de commentaire.
- /* commentaire */ permet d'ajouter un commentaire en plusieurs lignes.

21

Où placer son code ?

- Il est préférable de mettre le code à la fin de la page pour s’assurer de la lecture du code HTML.
- Placer le code à la fin de la page permet d’améliorer la performance de la page.

22

Utiliser la console

- La commande console.log() permet d'ajouter un message dans la console.
- console est une fonctionnalité qui n'est pas supportée par tous les navigateurs.
- On peut aussi utiliser:
  - console.debug
  - console.info
  - console.warn
  - console.error

23

La syntaxe JS

- En Javascript, la syntaxe définit l'ensemble de règles à suivre afin de:
  - Créer des variables.
  - Assigner des valeurs aux variables.
  - Faire un traitement sur des variables (conditions, boucles, opérateurs, etc...).

24

Les variables

- Une variable est utilisée afin de stocker une valeur donnée (string, number, boolean, array, object,...).
- Javascript utilise le mot clé **var** afin de déclarer une variable.
- Javascript est indulgent si on utilise pas le mot clé var.

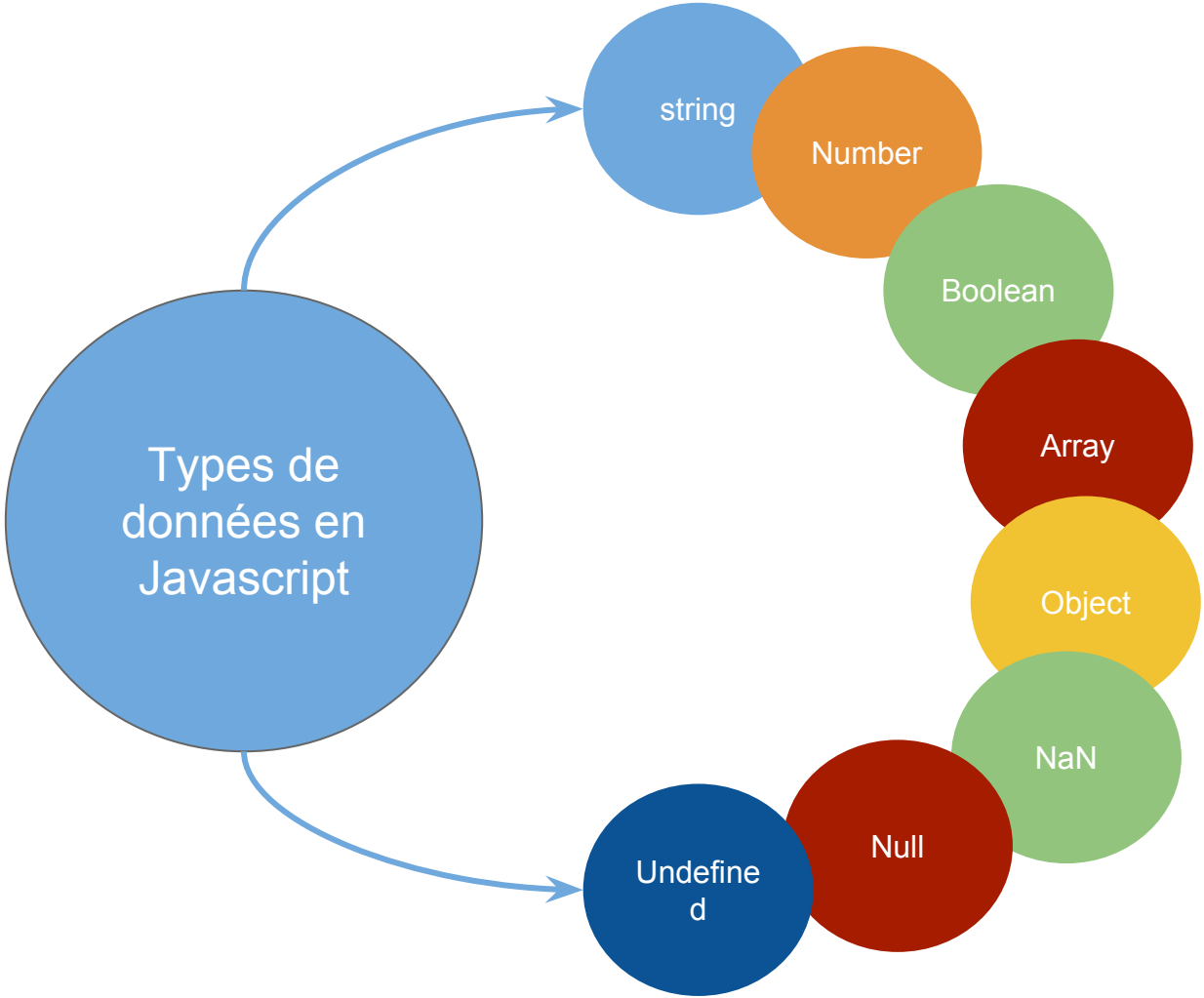
25

Les variables

```
var pi = 3.14;
var animal = "dog", language = "javascript";
var x = 5;
var test = true;
var lastName = "Alain", firstName = "Kockx", job =
"manager";
var destination;
var students = ["Alain", "Claude", "Bart"];
var family = {pere:"Claude", mere:"Brigitte",
enfant:"Emilie"};
```

26

Les variables



- Les noms de variables doivent respecter les règles suivantes:
  - Commencer par un caractère alphabétique.
  - Peut contenir des numériques.
  - Le seul caractère spécial autorisé est le “_” underscore.
  - Les noms réservés ne peuvent pas être utilisés tels que: void, true, false, else, etc...

Liste des mots clés en Javascript

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

- Portée des variables:
  - **Local:** Une variable locale est définie à l'intérieur d'une fonction.
  - **Global:** Une variable globale est visible dans tout le code Javascript.
- Initialisation des variables:
  - En JavaScript, il n'est pas nécessaire de déclarer les types de variables.
  - var status = "open";

    status = 0;

    status = true;

30

Les opérateurs

- Les opérateurs en Javascript permettent de réaliser des opérations entre variables, on peut avoir:
  - Des opérateurs d’affectation, ou concaténation.
  - Des opérateurs arithmétiques.
  - Des opérateurs logiques.
  - Des opérateurs de comparaison.
  - Des opérateurs d’incrémentation et de décrémentation.
  - Des opérateurs ternaires.

31

Les opérateurs

Les opérateurs les plus utilisés

Opérateur	Description
=	Assignation
+	Addition
-	Soustration
*	Multiplication
/	Division
++	Incrémenter
--	Décrémenter
&& / ||	Et Logique / Ou logique
!	Non logique

32

Les opérateurs

Les opérateurs arithmétiques:

Opérateur	Description	Exemple (X = 10)	Résultat
+	Addition	X+5	15
-	Soustration	X-2	8
*	Multiplication	X*2	20
/	Division	X/3	3
%	Modulo	X%3	1
++	Incrémenter	X++	11
--	Décrémenter	X--	9

33

Les opérateurs

Les opérateurs logiques:

Opérateur	Description	Exemple (x=1, y=2)	Résultat
&&	Et logique	x < 2 && y >1	true
||	Ou logique	x < 2 || y >1	true
!	Non logique	! (x > y)	true

34

Les opérateurs

Les opérateurs de comparaison:

Opérateur	Description	Exemple
==	Egale	
!=	Différent	
>	Supérieur	
<	Inférieur	
>=	Supérieur ou égale	
<=	Inférieur ou égale	

Les opérateurs d’assignation:

Opérateur	Usage	Equivalent
+=	x += y	x = x+y
-=	x -= y	x = x-y
*=	x *= y	x = x*y
/=	x /= y	x = x/y
%=	x %= y	x = x%y

- Les opérateurs ternaires:
  - condition ? expr1 : expr2;

3

Les boucles et les conditions

38

Les conditions

- Les structures conditionnelles en Javascript se font avec les mots clés **if** / **else if** / **else.**
- Nous avons plusieurs configurations possibles pour les conditions.
  - if (cond) { traitement }
  - if (cond) { traitement 1 } else { traitement 2 }
  - if (cond1) { traitement 1 } else if (cond2){ traitement 2 } else { traitement 3 }

39

Les boucles

- Une boucle en Javascript permet d’utiliser un code de façon répétée jusqu’à ce que la condition de la boucle arrive.
- Nous avons plusieurs configurations possibles pour les boucles:
  - for
  - for in
  - while
  - do while

40

Les boucles

```
for (var i=0; i<10; i++){
    console.log("i = "+i);
}
```

41

Les boucles

Parcourir un objet

```
for (var key in myObject){
    console.log("Key: "+key + " / value :"+myObject[key]);
}
```

Parcourir une liste

```
for (var index in myList){
    console.log("element :"+myList[index]);
}
```

42

Les boucles

- Le mot clé **break** permet de sortir d'une boucle.
- Le mot clé **continue** permet de passer à l'itération suivante sans finir le traitement en cours.

43

4

Les fonctions en Javascript

44

Les fonctions

- Une fonction est un bloc de code auquel on donne un nom et que l'on peut appeler plusieurs fois.
- Le nom de la fonction est une variable et il faut observer les mêmes critères de choix de noms.

45

Les fonctions

```
var a = 5, b=10;

function multNumbers(){
	var product  = a * b;
	console.log("Produit: "+product);
}
multNumbers();
multNumbers();
```

```
var a = 5, b=10;

function multNumbers(num1, num2){
	var product  = num1 * num2;
	console.log("Produit: "+product);
}
multNumbers(12,3);
multNumbers(40,5);
```

46

Les fonctions

```
function multNumbers(num1, num2){
        console.log(num1);
        console.log(num2);
        var product  = num1 * num2;
        return product;
}
var product_1 = multNumbers(12,3);
var product_2 = multNumbers(40,5);
console.log(product_1);
console.log(product_2);

// Cas particuliers...
var product_3 = multNumbers(1,2,5,7);
var product_4 = multNumbers(1);
console.log(product); // Il faut la définir comme une variable
globale
```

47

5

Les Structures de données: Les Objets et les tableaux

48

Les objets

- Un objet en javascript est une séquence de propriétés et de valeurs écrits avec une syntaxe **clé / valeur**.
- On peut initialiser un objet de 2 manières différentes.
- On peut assigner une fonction à un objet comme étant un attribut.

49

Les objets

```
var person1 = new Object();

person1.name = “Fred”;
person1.age = 40;
person1.sexe = ‘M’;

console.log(person1);
```

```
var person2 = {name:“Fred”, age:40, sexe: ‘M’};

console.log(person2);
console.log(person2.name);
console.log(person2.age);
console.log(person2.sexe);
```

50

Les objets

```
var person1 = {name:“Fred”, age:40, sexe: ‘M’};
var person2 = {name:“Tim”, age:32, sexe: ‘M’};

function personInfos(){
      console.log(this.name + “ est agé de “+this.age)
}
person1.logDetails = personInfos;
person2.logDetails = personInfos;

person1.logDetails();
```

51

L’objet Math

- L’objet **Math** contient des fonctions très utiles pour faire des opérations mathématiques.
- L’objet **Math** contient aussi des propriétés tels que PI.
- **Math** est présent dans javascript de façon native sans importer une librairie spécifique.

52

L’objet Math

```
var a =10, b = 15, c =2.1;

var minimum = Math.min(a,b,c);
var arrondi = Math.round(c); // arrondi au plus proche
var arrondi = Math.floor(c); // arrondi à l’inférieur
var arrondi = Math.ceil(c); // arrondi au supérieur

var rand = Math.random(); // random entre 0 et 1

var pi = Math.PI; // Le nombre PI
```

53

Les Dates

- L’objet **Date** contient des fonctions très utiles pour faire des opérations sur des dates.

54

Les Dates

```
var today = new Date();
var maDate = new Date(2015,05,06,14,20,26);

console.log(maDate.getDate());
console.log(maDate.getDay());
console.log(maDate.getMonth());
console.log(maDate.getHours());
console.log(maDate.getTime());
// Nombre de ms depuis le 01/01/1970
```

55

Les chaînes de caractères

```
var chaine = ‘une chaine de caractères’;

console.log(“ma chaine : ”+chaine);
var len = chaine.length;
var elements = chaine.split(‘ ’);
console.log(“elements : ”+elements);
console.log(“Nbre de Mots :
”+elements.length);
console.log(“Mot 1 : ”+elements[0]);
```

56

Les tableaux

- On peut initialiser un tableau en javascript sans spécifier la taille
  - var names = new Array();
  - var colors = new Array(10);
  - var cars = [ ];
  - var students = [“Tom”, “John”, “Chris”];
- On peut initialiser un élément du tableau comme ceci
  - colors[0] = “Rouge”;

57

Les tableaux

- Les tableaux sont basés sur des index alors que les objets en JS sont basés sur des propriétés.
- On peut accéder à un élément du tableau en spécifiant son index
  - `console.log(colors[0]);`

58

Les chaînes de caractères

```
var students = [ “Tom”, “John”, “Chris”];

console.log(“Liste : ”+students);
var len = students.length;
students.push(‘Tim’);
console.log(“New Liste: ”+students);
students.reverse();
console.log(“Liste inversée: ”+students);
```

6

Le DOM (Document Object Model)

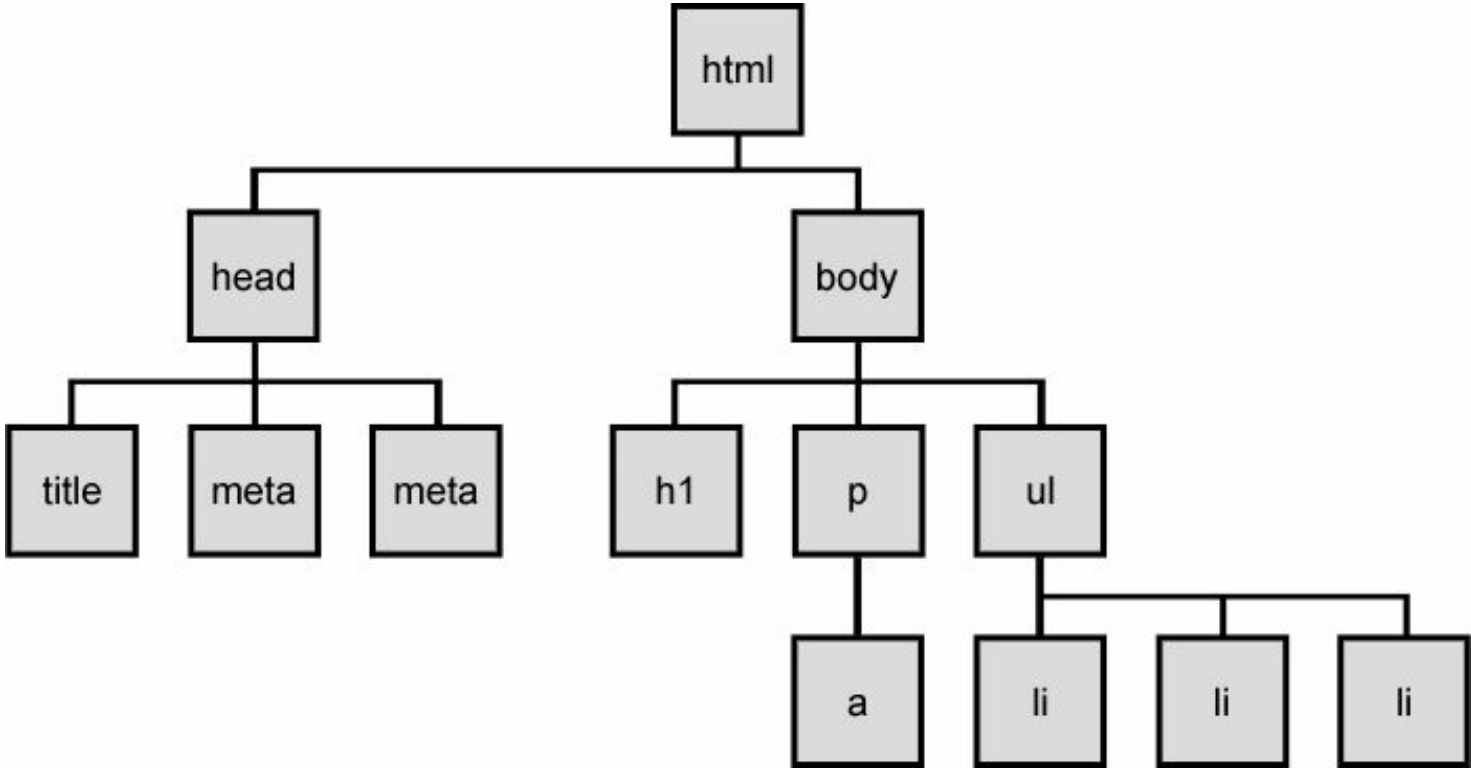
60

Le DOM

- Le DOM (Document Object Model).
  - **Document:** C'est la page Web dans laquelle s'exécute le javascript, on y accède à travers l'objet document.
  - **Object:** Javascript voit la page Web comme une hiérarchie d'objets (<html>, <h1>, etc...)
  - **Model:** Chaque élément de la page Web est un noeud (<html>, <h1>, etc...)

61

Le DOM



62

Le DOM

DOM (Document Object Model)

Une convention

Les élements

Une page Web

- Afin d'accéder aux éléments du DOM, on utilise des fonctions comme:
  - document.getElementById()
  - document.getElementsByTagName()
- On peut utiliser des méthodes afin de modifier une propriété de la page Web
  - element.setAttribute('property', 'value')
  - element.innerHTML = 'New text'

- On peut créer un élément en Javascript en utilisant la fonction:
  - document.createElement(“h1”)
  - document.createElement(“p”)
- Une fois l’élément créé, on peut lui attribuer un contenu via la propriété innerHTML.
- La fonction appendChild permet de rajouter un noeud enfant à un élément.

- On peut utiliser la fonction document.createTextNode() afin de créer un texte, et ensuite ajouter le texte comme un noeud enfant à travers la fonction appendChild().

```
var heading = document.createElement(“h1”);
var paragraph = document.createElement(“p”);

// Option 1…
heading .innerHTML = “Header de mon texte”;
paragraph .innerHTML = “Voici mon texte”;

// Option 2...Noeuds
var h1Text = document.createTextNode(“Header de mon texte”);
var pText = document.createTextNode(“Voici mon texte”);
// On ajoute les noeuds comme des childs
heading.appendChild(h1Text);
paragraph .appendChild(pText);
```

7

La gestion des événements

68

Les événements

- Un événement est une action qui se passe dans votre page Web, par exemple:
  - Cliquer sur un bouton.
  - Survoler un titre.
  - Sélectionner un texte.
  - Etc.
- Si on décide de réagir par rapport à un événement, on doit créer un gestionnaire d’événements.

```
// Option 1...
<input type="button" id = “idButton” onclick="alert(‘salut !’)"
value="Cliquez">

// Option2...
var myButton = document.getElementById(“idButton”);
myButton.onclick = function(){
      alert(‘salut’);
}

// Option3…
var myButton = document.getElementById(“idButton”);
myButton.addEventListener(“click”,
            function(){
                  alert(‘Salut !’);
            },
            false);
```

- **addEventListener** reçoit 3 paramètres, le 3ème paramètre est souvent false.
- Pour une meilleure lisibilité, il vaut mieux externaliser la fonction pour voir les 3 paramètres et avoir un code plus clair.

71

Les événements

- Javascript propose une liste complète d'événements afin de gérer tout type d’actions sur la page HTML. On va se contenter de travailler sur ces events:
  - onClick.
  - onLoad
  - onFocus
  - onBlur

72

Les événements

```
var eMailField = document.getElementByld('emailId');

emailField.onfocus = function(){
      If (emailField.value == "Votre email"){
            emailField.value = "";
      }
};

emailField.onblur = function(){
      If (emailField.value == ""){
            emailField.value = "Votre email";
      }
};

// emailField.style.backgroundColor = "blue";
```

8

Les formulaires

74

Les formulaires

- On peut accéder à un formulaire de 2 manières différentes:
  - Par son id = getElementById(‘formId’).
  - Par son nom = document.forms[‘formName’]
- Chaque champ du formulaire est un noeud et on peut y accéder via l’objet form, par exemple, si un input possède l’attribut name =’myInput’ :
  - var myInput = myForm.myInput;

- On peut valider un formulaire avant envoi, en utilisant l'événement **onsubmit.**
- La syntaxe dans la balise <form> s'écrit donc:
  - <form name="myForm" **onsubmit**="return **validateForm()**">
  - validateForm() est le nom de la fonction qui fait la validation.

9

Les cas pratiques

77

Minifier du Code

- La minification de code est une opération qui consiste à réduire la taille des fichiers JS afin de les rendre plus légers et améliorer la performance du site Web.
- La minification consiste à:
  - Réduire les espaces.
  - Changer les noms des variables.
- https://closure-compiler.appspot.com/home

78

Le mode strict

```
// “use strict”

name = ‘Bonjour tout le monde’;

function addition(a,b,a) {
        var addition = a+b+c;
}

console.log(‘name = ’+name);
```

79

Librairies JS

- https://mootools.net/
- https://dojotoolkit.org/
- https://developers.google.com/closure/library/
- https://script.aculo.us/
- https://jquery.com/