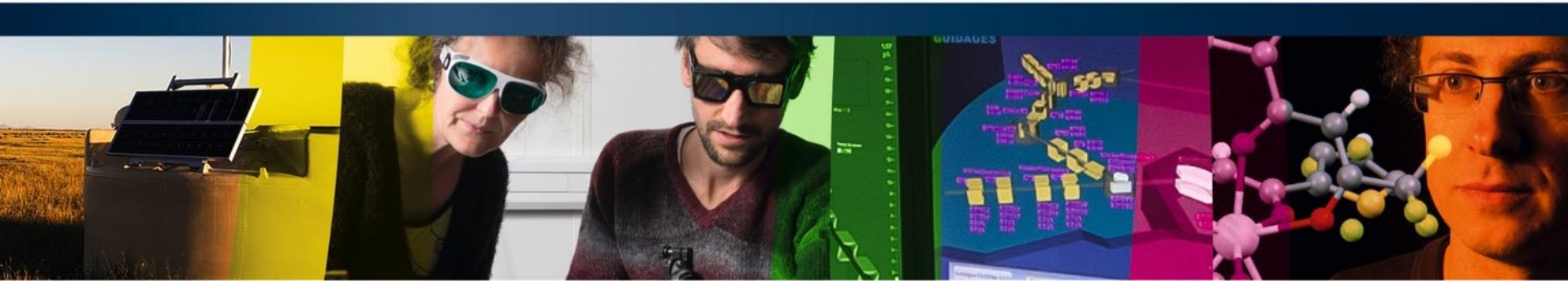
www.cnrs.fr

HTML5 et la sécurité



www.cnrs.fr

Sommaire

1. Qu’est-ce HTML5 ?
2. Failles de sécurité
3. Comment améliorer la sécurité ?

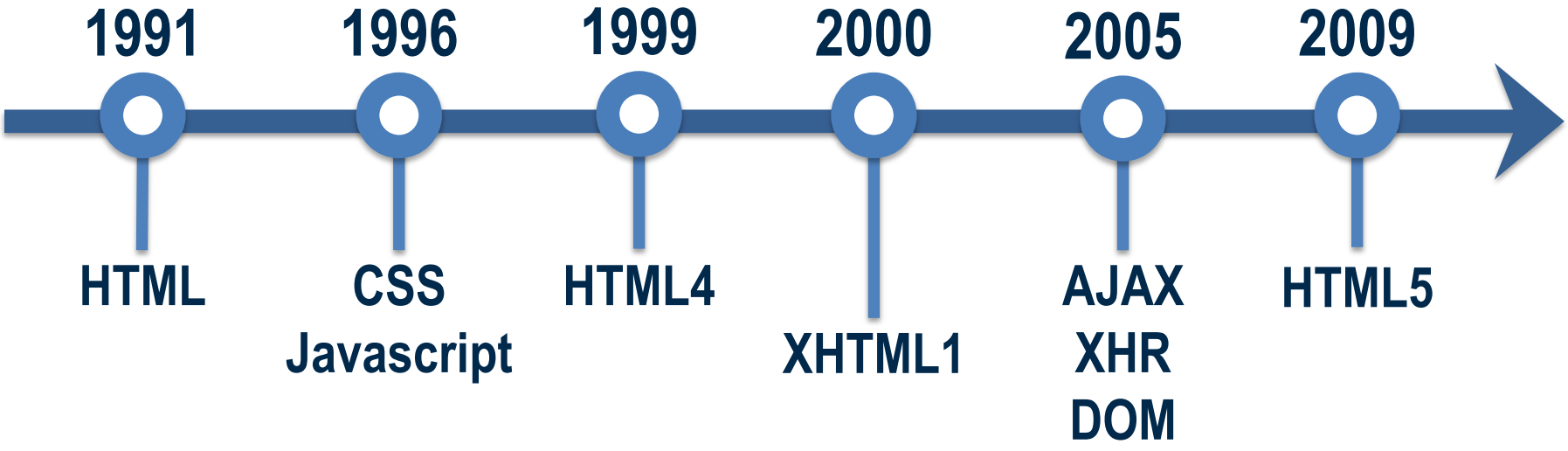
www.cnrs.fr

Qu'est-ce HTML5 ?

1. Un peu d'histoire
2. Description
3. Architecture

Qu’est-ce que HTML5 ?

- Un peu d’histoire



- HTML5 = HTML + CSS3 + EcmaScript

Qu'est-ce que HTML5 ?

- Format de données conçu pour représenter les pages web
- 5ème révision du langage
  - Nouveaux éléments de structures, de formulaires, de sémantique
  - Animations via les Canvas CSS3
  - Prise en charge d'éléments multimédia (audio, vidéo) nativement
  - Géolocalisation
  - Communication
  - Stockage
  - Bases de données

- Vérifier le support de HTML5 dans les navigateurs

  http://html5test.com/

Qu'est-ce que HTML5 ?

Architecture

Media Géo Messages

Présentation

DOM/Parser | Events | Threads

IndexedDB Cache Storage

Logique & Processus

XHR | WebSocket

Service Réseau du navigateur

Réseau

SOP/CORS | Sandbox

Noyau

www.cnrs.fr

Failles de sécurité

1. APIs de communication
2. APIs de stockage
3. Géolocalisation
4. Web workers

Failles de sécurité

- **APIs de communication**
  - **Web messaging**
    - Cross Domain Messaging
    - API de communication entre fenêtres sur des domaines différents
    - Syntaxe : `window.postMessage(message,targetOrigin)`
    - `targetOrigin` peut être positionné sur une source connue et fiable ou ouvert à tout le monde avec « * »
    - Cible d'attaque par des sites malveillants qui envoient des données malicieuses

Failles de sécurité

- **APIs de communication**
  - **Web messaging**
    - À l'envoi du message, spécifier explicitement la cible
    - Sur la page de réception, vérifier l'origine du message
      `if(message.origin.indexOf(".nmsaw.fr")!=-1 { … }`
    - Sur la page de réception, vérifier le format des données
    - Ne pas interpréter les données échangées
    - Pour assigner les données à un élément de la page, ne pas utiliser ~~`element.innerHTML`~~`=data` mais `element.textContent=data`

Failles de sécurité

- **APIs de communication**
  - **Same Origin Policy**
    - Les navigateurs permettent à un objet d'accéder à des objets uniquement s'ils sont du même domaine
    - SOP donne des privilèges
      - Accès total aux fonctionnalités réseau
      - Lecture/Ecriture du DOM
      - API de stockage
    - Mais SOP empêche AJAX de fonctionner facilement et correctement
      Réponse du W3C : Cross Origin Resource Sharing (CORS)

Failles de sécurité

- **APIs de communication**
  - **Cross Origin Resource Sharing**
    - Contrôlé par header HTTP `origin`
    - Permet à une requête d'atteindre sa cible extérieure
    - Cible d'attaque pour exploiter les cookies, déposer des fichiers sans que la victime en ait connaissance, scanner les ports
    - Configurer le header (côté serveur) pour autoriser une liste blanche (plutôt que « * » ou liste noire)
    - RFC recommande l'utilisation d'une requête `OPTIONS` avant de faire une requête `POST` ou `GET`
    - Ne pas autoriser l'envoie de cookies (`xhr.withCredentials=true`)

Failles de sécurité

- **APIs de communication**
  - **WebSockets**
    - Canal de communication bidirectionnel avec connexion permanente
    - Transparent pour les firewalls, proxy, et routeurs.
    - Etend n'importe protocole basé sur TCP/UDP (tunnel VNC, FTP, …)
    - Le protocole ne gère pas les autorisations ou l'authentification
    - Cible d'attaque pour scanner les ports, accéder aux services tunnelés

    - Vérifier les données reçues
    - Ne pas interpréter les données échangées
    - Vérifier le header HTTP `origin`

Failles de sécurité

- **APIs de communication**
  - **Événements serveurs**
    - Permet de recevoir des notifications initiées par le serveur Web
    - Vérifier l'URL passée en paramètre du constructeur `EventSource`
    - Vérifier l'origine du message `event.origin`
    - Vérifier les données reçues
    - Ne pas interpréter les données échangées

Failles de sécurité

- **APIs de stockage**
  - **Stockage local**
    - Stockage de données sous forme de paires clé/valeur
    - Isolation des données basée sur SOP
    - Plus sûr que les cookies, plus rapide
    - `LocalStorage` : pas de date d'expiration
    - `SessionStorage` : durée = la session
    - Mécanisme varie entre les navigateurs
    - Cible d'attaque par XSS pour lire, écrire, altérer des données
    - Utiliser `SessionStorage` si la persistance n'est pas nécessaire
    - Ne pas stocker de données sensibles
    - Éviter d'exécuter différents applications sur la même origine

Failles de sécurité

- **APIs de stockage**
  - **IndexedDB**
    - Base NoSQL Document (pas de tables, format JSON)
    - Mécanisme varie entre les navigateurs
    - Cible d'attaque par XSS ou injection pour lire, écrire, altérer des données
    - Ne pas stocker de données sensibles

Failles de sécurité

Géolocalisation

- Utilise plusieurs moyens : GPS, triangulation, GSM/3G, Wifi, @IP
- Supporté par tous les navigateurs récents
- Implémentation varie entre les navigateurs
- Risques liés aux données privées

- Demander l'autorisation à l'utilisateur avant d'utiliser `getCurrentPosition` or `watchPosition`
- Ne pas stocker les positions de l'utilisateur en local

Failles de sécurité

Web Workers

- Moyen d'exécuter des scripts en tâche de fond (multithread)
- Un Web Worker n'a pas accès au DOM de la page
- Un message envoyé à un WebWorker peut être un objet JavaScript
- Cible d'attaque de type Denial of Service à cause d'une consommation CPU excessive

- Vérifier les données échangées
- Ne pas interpréter les données échangées

Comment améliorer la sécurité ?

1. IFrame Sandbox
2. Headers HTTP
3. Bonnes pratiques de développement

Comment améliorer la sécurité ?

IFrame Sandbox

- `<iframe sandbox src=”...”></iframe>`
- Permet de restreindre les capacités d’un IFrame
  - Bloque la validation des formulaires
  - Bloque l’exécution de scripts
  - Désactive les APIs
  - Empêche de navigue vers des domaines externes
  - Empêche l’exécution de plugins (`<embed>`, `<object>`, `<applet>`, …)
  - Empêche de changer de niveau de navigation
  - Empêche les mécanismes automatiques (vidéo, autofocus, …)
- Permet de n’activer que les fonctionnalités strictement nécessaires

- Pour les navigateurs anciens, cet attribut est ignoré
- En complément, utiliser le header `X-Frame-Options`

Comment améliorer la sécurité ?

- **Headers HTTP**
  - **X-Frame-Options (obsolète) / frame-ancestors**
    - Permet de demander au navigateur de ne pas exécuter les objets embarqués en fonction de leur provenance ou de leur type
    - Protection contre le clickjacking (cf. Clickjacking Defense Cheat Sheet)
    - Définir le domaine autorisé explicitement

```
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: frame-ancestors nmsaw.fr ;
```

  - **X-XSS-Protection**
    - Protection contre les attaques XSS
    - Exemple

```
X-XSS-Protection: 1; mode=block
```

Comment améliorer la sécurité ?

- **Headers HTTP**
  - **Strict Transport Security (STS)**
    - Force les navigateurs à communiquer avec TLS/SSL

```
Strict-Transport-Security: max-age=8640000;
includeSubDomains
```

  - **Content Security Policy (CSP)**
    - Politique pour restreindre les ressources autorisées

```
Content-Security-Policy: allow 'self'; img-src *;
object-src media.nmsaw.fr; script-src js.nmsaw.fr
```

Comment améliorer la sécurité ?

- Bonnes pratiques de développement
  - Règle n°1

NE JAMAIS FAIRE CONFIANCE !

Comment améliorer la sécurité ?

- **Bonnes pratiques de développement**
  1. **Ne jamais faire confiance aux données échangées**
     - Vérifier les données
     - Vérifier le format des données
     - Ne pas interpréter les données
  2. **Vérifier l'url d'origine lors d'échanges cross domain**
  3. **Ne pas stocker de données sensibles**

Bibliographie

- **OWASP HTML5 Security Cheat_Sheet**

https://www.owasp.org/index.php/HTML5_Security_Cheat_Sheet

- **HTML5 Top 10 Threats Stealth Attacks and Silent Exploits**

https://media.blackhat.com/bh-eu-12/shah/bh-eu-12-Shah_HTML5_Top_10-WP.pdf

- **HTML5 Security Cheat Sheet**

https://html5sec.org/

- **W3C Recommendation Content Security Policy Level 2**

https://www.w3.org/TR/CSP11/

- **Mozilla Developer NetworkHTTP headers**

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/

Questions ?
Réponses !