Sapienza Università di Roma
corso di laurea in Ingegneria informatica e automatica

Linguaggi e tecnologie per il Web

a.a. 2018/2019

Parte 6
JSON, Web Storage API, funzioni JavaScript e user script

Riccardo Rosati

JSON

- JSON = JavaScript Object Notation
- Formato standard per la serializzazione e de-serializzazione degli oggetti JavaScript
- Serializzazione = trasformazione dell'oggetto JavaScript in stringa
- De-serializzazione = trasformazione della stringa in oggetto JavaScript
- JSON è un formato standard (ECMA 404) per lo scambio dei dati

JSON

- Esempio 1: rappresentazione di un array:

```
["ciao", 45, null, true]
```

- Esempio 2: rappresentazione di un oggetto con due proprietà prop1 (stringa), prop2 (booleano):

```
{prop1: "ciao", prop2: true}
```

- Esempio 3: rappresentazione di un oggetto con due proprietà: p1, il cui valore è l'array dell'Esempio 1, e p2, il cui valore è l'oggetto dell'Esempio 2:

```
{p1: ["ciao", 45, null, true],
 p2: {prop1: "ciao", prop2: true}}
```

JSON: valori

- Un testo JSON è la rappresentazione di un valore
- Un valore può essere:
  - oggetto
  - array
  - numero
  - stringa
  - `true`
  - `false`
  - `null`

```
valore -> oggetto | array | numero | stringa | "true"
          "false" | "null"
```

JSON: oggetti

- Un oggetto è una sequenza di coppie nome-valore (separate da virgole) racchiusa tra parentesi graffe

```
oggetto -> "{" (coppia altreCoppie)? "}"
coppia -> nome ":" valore
altreCoppie -> ("," nome ":" valore )*
```

JSON: array

- Un array è una sequenza di valori separati da virgole e racchiusi tra parentesi quadre

```
array -> "[" (valore altriValori)? "]"
altriValori -> ("," valore )*
```

JSON: numeri e stringhe

- I numeri sono rappresentati nella sintassi usuale dei linguaggi di programmazione (possono esere rappresentati sia numeri interi che in virgola mobile)
- Le stringhe sono sequenze di caratteri delimitate da doppi apici

L’oggetto JSON

- Oggetto JSON contenente due metodi:
  - Metodo per la de-serializzazione: JSON.parse(stringa)
  - Metodo per la serializzazione: JSON.stringify(oggetto)

Memorizzazione persistente

Per la memorizzazione permanente in script lato client:

- Interazione con il web server (l'applicazione lato server memorizza in modo permanente i dati inviati dal client)
- Uso di cookie o file locali (Web Storage API)
- Uso di database locali (IndexedDB API)

Web Storage API

- Nuove API in HTML5: Web Storage API
- Due nuovi oggetti:
  - `localStorage`
  - `sessionStorage`
  - Derivano entrambi dal nuovo oggetto `Storage`
- `localStorage` permette la memorizzazione persistente
- `sessionStorage` permette la memorizzazione a livello di sessione

Memorizzazione persistente

- Entrambi gli oggetti sono utilizzati definendo nuove coppie chiave-valore
- Es. `localStorage.cognome="Rosati"` memorizza una nuova coppia
- I valori assegnabili ad una proprietà dello storage **devono essere di tipo String**
- Per poter memorizzare oggetti arbitrari JavaScript, è necessario serializzarli (ad es. tramite il metodo JSON.stringify)

Esempio

```
localStorage.cognome="Rosati";
localStorage.nome="Riccardo";
var o = { s1: 5, p2: "ciao" };
localStorage.oggetto=JSON.stringify(o);
alert(localStorage.cognome + " " +
      localStorage["nome"]);
```

Altri metodi dello storage

Altri metodi dell'oggetto `Storage` (e quindi degli oggetti `sessionStorage` e `localStorage`):

- `getItem(chiave)`
- `setItem(chiave,valore)`
- `removeItem(chiave)`
- `clear()`

L'esecuzione dei metodi `setItem`, `removeItem` e `clear` genera l'evento `storage` (che può essere gestito come gli tutti altri eventi del DOM)

Gestione dei dati dello storage

Analogie e differenze con i cookie:

- Gli oggetti del Web Storage hanno un limite di dimensione di (almeno) 5 MB (molto maggiore dei cookie)
- I dati contenuti in questi oggetti NON vengono trasmessi automaticamente al server (a differenza dei cookie)
- Ogni dato (proprietà) contenuto negli oggetti storage è visibile solo dagli script provenienti dallo stesso dominio dello script che lo ha creato (same origin policy): questa politica è simile (anche se non esattamente uguale) a quella adottata per i cookie

IndexedDB API

- Le IndexedDB API rispondono a necessità di memorizzazione permanente lato client che localStorage e sessionStorage non possono soddisfare
- Permettono di usare un Object store (database) lato client
- Non è un database relazionale e non si usa il linguaggio SQL
  - una proposta in tal senso, chiamata Web SQL Database API, non è stata adottata come standard W3C
- Proprietà indexedDB dell'oggetto window
- Per i dettagli della API si rimanda alla documentazione ufficiale: https://www.w3.org/TR/IndexedDB/

Funzioni in JavaScript

- Le funzioni in JavaScript sono **oggetti**
- Possono pertanto, ad esempio, essere assegnate a delle proprietà
- Possono anche essere assegnate a delle variabili:

```
var p = function(a,b){
  …
}
```

- Il precedente è un esempio di **funzione anonima** (o funzione lambda)

Funzioni in JavaScript

- La funzione precedente può essere invocata usando il nome della variabile:

```
var x = p(32,5);
```

oppure può essere invocata in questo modo:

```
(function(a,b){
  …
}(32,5))
```

Funzioni in JavaScript

- Le funzioni hanno proprietà e metodi definiti
- Proprietà:
  - `name` (nome della funzione)
  - `length` (numero di argomenti)
  - …
- Metodi (per invocare la funzione):
  - `call`
  - `apply`
  - `bind`

Funzioni di callback

- Come gli altri oggetti, le funzioni possono essere passate come parametri nella chiamata ad un'altra funzione (**callback**)
- Esempio:

```
function f(x,y){…}
function oper(f,a,b){
  return f(a,b); }
```

oppure

```
document.form1.bott.addEventListener(
"click",f);
```

Funzioni restituite da funzioni

- Come gli altri oggetti, le funzioni possono anche essere restituite come risultato di una funzione
- Esempio:

```
var x = function(y){
  return function (z){
  return y*z; };
};
```

Metodi = funzioni = proprietà

- I metodi degli oggetti sono funzioni
- Ma le funzioni sono oggetti!
- In realtà quindi anche i metodi sono oggetti
- Più precisamente, ogni metodo è una proprietà che ha come valore una funzione
- In questo senso non c'è più distinzione tra metodi e proprietà in JavaScript
- Ogni oggetto è un contenitore di proprietà; i metodi dell'oggetto sono quelle proprietà valorizzate ad oggetti di tipo funzione

User script

- JavaScript può essere usato per definire i cosiddetti **user script** dei browser
- Gli user script sono script che alcuni browser (tramite opportune estensioni) eseguono automaticamente in corrispondenza al caricamento di certe pagine web
- Ogni user script è definito per una classe di pagine (ad esempio, tutte le pagine provenienti da un dominio o da una porzione di un dominio)

Userscript manager

Le estensioni/plugin dei browser che permettono l'esecuzione di user script sono chiamate **userscript manager**

Principali estensioni di questo tipo:

- Greasemonkey (Firefox)
- Tampermonkey (Chrome, Firefox, Safari, Edge)
- Ace Script (Firefox)
- ….

Esempio di user script

Vogliamo scrivere uno user script che fa aggiungere automaticamente al browser la visualizzazione (come primo elemento della pagina) dell'immagine del logo Sapienza.

Vogliamo applicare tale script a tutte le pagine web la cui URL inizia con il seguente prefisso:

`http://www.dis.uniroma1.it/~rosati/`

Tuttavia (come eccezione alla precedente regola) non vogliamo applicare tale script alle pagine web la cui URL inizia con il seguente prefisso:

`http://www.dis.uniroma1.it/~rosati/dmds/`

Esempio di user script (Greasemonkey)

```
// ==UserScript==
// @version          1.0
// @name             Esempio2
// @description      Esempio 2 di userscript per LTW
// @match            http://www.dis.uniroma1.it/~rosati/*
// @exclude          http://www.dis.uniroma1.it/~rosati/dmds/*
// ==/UserScript==
function aggiungiLogoSapienza(){
    var body1 = document.getElementsByTagName("body");
    var body = body1[0];
    var logoSapienza = document.createElement("img");
    logoSapienza.setAttribute("src","https://www.uniroma1.it/sites/
default/files/images/logo/sapienza-big.png");
    body.insertBefore(logoSapienza,body.firstChild);
    alert("Fatto!");
}
aggiungiLogoSapienza();
```

Link

Web storage API:

- https://www.w3.org/TR/webstorage/

Indexed Database API:

- https://www.w3.org/TR/IndexedDB/

Greasemonkey:

- http://www.greasespot.net/

Tampermonkey:

- https://tampermonkey.net/